Каипова А.Д., Казагачев В.Н.

ПРОГРАММИРОВАНИЕ НА PYTHON

Учебне пособие

ЭВЕРО
Алматы, 2022

УДК 004 (075.8)
ББК 32.973.202 я 73
К 15

Рецензенты:
Н.В. Иваницкая – к. ф.-м.н., доцент, директор департамента по международному сотрудничеству с вузами России и СНГ;
А.К. Кайракбаев – доктор PhD, зав.лабораторией Технопарка «Zerek» Актюбинского университета им. С. Баишева.

К 15 Каипова А.Д., Казагачев В.Н.
Программирование на Python: учебное пособие / А.Д. Каипова, В.Н. Казагачев.– Алматы: Эверо, 2022.– 288 с.

ISBN 978-601-352-615-7

Python является простым и, в то же время, мощным интерпретируемым объектно-ориентированным языком программирования. Пособие предназначено для изучение основных конструкций языка Python, которые пригодятся при решении широкого круга задач – от анализа данных до разработки новых программных продуктов.

В результате освоения – научатся обрабатывать и хранить числа, тексты и их наборы, освоят стандартную библиотеку языка Python и смогут автоматизировать задания по сбору и обработке данных.

Учебное пособие предназначено для студентов и преподавателей инженерных и естественно-научных специальностей вузов, школьников старших классов и учителям информатики.

УДК 004 (075.8)
ББК 32.973.202 я 73

ISBN 978-601-352-615-7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная дисциплина предназначена для формирования у студентов умений разрабатывать приложения и анализировать данные посредством современного языка программирования Python. При изучении дисциплины рассматриваются структура языка, основные элементы языка, типы данных простые и составные, особенности работы с типами данных, присущих этому языку программирования. Закрепляются навыки разработки всех типов алгоритмов и их реализация на изучаемом языке. Курс разрабатывает и расширяет основные навыки вычислительного мышления и решения проблем, используя программные средства. Успешное завершение этого курса подготовит студентов к использованию базовой лексики вычислений техники и созданию программ, написанных на языке программировании Python.

Компетенции дисциплины:

A) обладать способностью и постоянной готовностью совершенствовать и углублять свои знания, быстро адаптироваться к любым ситуациям;

C) обладать навыками самостоятельного построения алгоритма и его анализа;

C) разрабатывает компоненты компьютерных систем обработки информации и управления, модулей программ и программных комплексов заданного качества;

D) обладать умением проанализировать результат и скорректировать математическую модель, лежащую в основе задачи;

D) способность к обобщению и анализу, постановке целей и выбору путей ее достижения при проектировании и разработке прикладного программного обеспечения.

Курс дает необходимую базу для освоения более специализированных областей применения языка Python, таких как машинное обучение, статистическая обработка данных, визуализация данных и многих других.

Результаты обучения дисциплины:

Уровни таксономии Блума	Знание и понимание	Применение	Анализ	Синтез	Оценивание
способность к обобщению и анализу, постановке целей и выбору путей ее достижения при проектировании и разработке программного обеспечения; иметь навыки разработки и реализации процессов жизненного цикла информационных систем, программного обеспечения, сервисов систем информационных технологий, а также знать и уметь применять методы и механизмы оценки и анализа функционирования средств и систем информационных технологий; разрабатывать компоненты компьютерных систем обработки информации и управления, модулей программ и программных комплексов заданного качества;	понимать концепцию и идеи языка программирования; знать и уметь использовать структурные элементы языка при разработке алгоритмов и их реализации; знать основные возможности интерпретатора Python	иметь навыки: составления программ на языке программирования Python, использовать среду разработки IDLE для разработки и отладки программ на языке Python.	Уметь анализировать данные, эффективность алгоритмов; проводить последующий анализ полученных результатов выполнения программ	Уметь конструировать программы из отдельных разработанных модулей и структур	Уметь оценивать функциональность программы

ЛЕКЦИОННЫЙ КОМПЛЕКС

Лекция 1.

Введение в программирование на языке Python. Основы языка. Переменные, значения и их типы. Присваивание значения. Ввод значений с клавиатуры. Встроенные операции и функции.

План:

Назначение языка. Основные особенности языка программирования Python. Синтаксис и основные функции языка. Переменные и типы данных

Python это мощный и высокоуровневый объектно-ориентированный язык программирования общего назначения, созданный Гвидо ван Россумом.

Синтаксис

Программа на языке Python состоит из набора инструкций. Каждая инструкция помещается на новую строку.

Python не содержит операторных скобок (begin..end в pascal или {..}в Си), вместо этого блоки выделяются отступами: пробелами или табуляцией, а вход в блок из операторов осуществляется двоеточием. Большую роль в Python играют отступы. Неправильно поставленный отступ фактически является ошибкой.

Основные функции

Основной функцией для вывода информации на консоль является функция print(). В качестве аргумента в эту функцию передается строка, которую надо вывести: print("Лекция")

Функция input() отвечает за ввод информации. В качестве необязательного параметра эта функция принимает приглашение к вводу и возвращает введенную строку, которую мы можем сохранить в переменную: name = input("Введите имя: ")

Переменные и типы данных

Переменная хранит определенные данные. Название переменной в Python должно начинаться с алфавитного символа или со знака подчеркивания и может содержать алфавитно-цифровые символы и знак подчеркивания.

В Python применяется два типа наименования переменных: *camel case и underscore notation. Camel case* подразумевает, что каждое новое подслово в наименовании переменной начинается с большой буквы.

Underscore notation подразумевает, что подслова в наименовании переменной разделяются знаком подчеркивания.

Типы данных, подразделяются на категории: числа, последовательности, словари, наборы:

- boolean - логическое значение True или False
- int - представляет целое число, например, 1, 4, 8, 50.
- float - представляет число с плавающей точкой, например, 1.2 или 34.76
- complex - комплексные числа
- bytes - последовательность чисел в диапазоне 0...255
- byte array - массив байтов, аналогичен bytes с тем отличием, что может изменяться
- str - строки, например "hello". В Python 3.x строки представляют набор символов в кодировке Unicode
- list – список. Списки - похожи на одномерные массивы (Список включающий списки - многомерный массив)
- tuple – кортеж - неизменяемый список
- set - неупорядоченная коллекция уникальных объектов
- frozen set - то же самое, что и set, только не может изменяться (immutable)
- dict (dictionaries)- словарь, где каждый элемент имеет ключ и значение

Словари - тоже списки, но индексы могут быть любого типа, а не только числовыми.

"Массивы" в Python могут содержать данные любого типа, то есть в одном массиве могут находиться числовые, строковые и другие типы данных. Массивы начинаются с индекса 0, а последний элемент можно получить по индексу -1

Python является языком с динамической типизацией. Он определяет тип данных переменной исходя из значения, которое ей присвоено.

Python поддерживает все основные арифметические операторы (см. таблицу. 1.1).

Таблица 1.1. Арифметические операции с числами и встроенные функции с одним и двумя аргументами.

X + y	Сложение - сумма X и Y (2 + 3 = 5)
X - y	Вычитание раность X и Y (5 - 2 = 3)
X * y	Умножения-произведение X и Y (2 * 3 = 6)
X / y	деление x на y: 4 / 1.6 = 2.5, 10/5 = 2.0, результат всегда типа float
X // y	Деление x на y нацело: 11 / / 4 = 2, 11.8//4 = 2.0, результат целый, только если оба целые аргумента
X % y	Остаток от подразделения: 11% 4 = 3, 11,8% 4 = 3.8000000000000007 (существует ошибка, соединенная с погрешностью данных представления в компьютере),
x ** y	Возведение x в степень y: (2 ** 3 = 8)
abs(x)	Модуль числа x
round(x)	Округление (round (11.3) = 11)
round (x, n)	Округление числа x до n знаков после запятой: round(12.34567, 3) = 12.346
pow(x, y)	полный аналог записи x ** y
divmod(x, y)	выдает два числа: частный и остаток, обращатся так: q, r = divmod (x, y)

Отметим, что операции сложения, вычитания, умножения и возведения в степень выдают ответ типа int только если оба аргмента целые, типа float, если один из аргументов действительный, а другой _ целый или действительный, и типа complex, если хотя бы один аргумент комплексный. Операция возведения в степень также может выдать комплексный результат при возведении отрицательных чисел в степень кроме случая, когда эта степень целая и нечётная. То сть, эти опраторы в Python подчиняются общеупотребительным правилам преобразования типов.

```
>>>
>>> 7 * 3.
21.0
>>> 2**10
1024
>>> 8 % 3
2
```

Преобразование типов (кастинг):

```
>>>
>>> float(1)
1.0
```

Целочисленное деление в Python 2:

```
>>>
>>> 3 / 2
1
```

в Python 3:

```
>>>
>>> 3 / 2
1.5
```

Для безопаности:

```
>>>
>>> 3 / 2.
1.5

>>> a = 3
>>> b = 2
>>> a / b # In Python 2
1
>>> a / float(b)
1.5
```

В Python3

```
>>>
 ·>
 ·>> 3 / 2
```

1.5

Если Вы явно хотите использование целочисленного деления //:

```
>>>
>>> 3.0 // 2
1.0
```

Поведение оператора деления изменилось в Python 3.

Вопросы для закрепления:

1. Что представляет собой тип данных?
2. Какие типы данных изучаемого языка вы знаете?
3. Операторы ввода и вывода информации

Литература

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017.— 624с.: ил.—(Мировой компьютерный бестселлер).
2. Гвидо Ванн Россум и др. Язык программирования Python. / 2001 — 454 с.
3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.
4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.
5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.
6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лекция 2
Операторы языка. Основные алгоритмические конструкции. Условный оператор. Множественное ветвление

План:

Арифметические операции. Функции преобразования чисел. Операции сравнения. Логические операции. Условная конструкция if

Арифметические операции: + , -, *, /, // (целочисленное деление), ** (возведение в степень), %

Приоритеты операций:

Операции	Направление
**	Слева направо
* / // %	Слева направо
+ -	Слева направо

Функции преобразования чисел

Функции int() и float() позволяют привести значение к типу int и float соответственно.

Для округления результата можно использовать функцию round().

Представление числа

При обычном определении числовой переменной она получает значение в десятичной системе.

Для определения числа в двоичной системе перед его значением ставится 0 и префикс b:

x = 0b101 # 101 в двоичной системе равно 5

Для определения числа в восьмеричной системе перед его значением ставится 0 и префикс o:

a = 0o11 # 11 в восьмеричной системе равно 9

Для определения числа в шестнадцатеричной системе перед его значением ставится 0 и префикс x:

y = 0x0a # a в шестнадцатеричной системе равно 10

Для вывода числа в различных системах исчисления используются функция format, которая вызывается у строки. В эту строку передаются различные форматы. Для двоичной системы "{0:08b}", где число 8 указывает, сколько знаков должно быть в записи числа.

Условные выражения

Операции сравнения: = =, !=, >, <, >=, <=.

Логические операции: and, or, not

Сначала выполняется оператор not, затем оператор and, а в конце оператор or.

Условная конструкция if:

if логическое_выражение:
инструкции
[elif логическое выражение:
инструкции]
[else:
инструкции]

Вложенные конструкции if

Конструкция if в свою очередь сама может иметь вложенные конструкции if, которые также должны начинаться с отступов, а инструкции во вложенных конструкциях также должны иметь отступы.

Вопросы для закрепления:

1. Какие арифметические операции вы знаете?

2. Полная форма if. Объясните назначение команды. Сравните данный операторв в языках C++ и Python.

3. Перечислите функции преобразования типов

Литература

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017.— 624с.: ил.—(Мировой компьютерный бестселлер).

2. Гвидо ванн Россум и др. Язык программирования Python. / 2001 — 454 с.

3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лекция 3
Циклы и счетчики. Операторы while и for

План:

Циклы While, for. Функция range(). Оператор else в цикле for. Операторы break, continue

Цикл while:

while условное_выражение:

инструкции

После ключевого слова while указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций, который идет далее. Все инструкции, которые относятся к циклу while, располагаются на последующих строках и должны иметь отступ (табулятор или 4 пробела) от начала строки.

Цикл for

Цикл for предназначен для перебора элементов структур данных и некоторых других объектов.

```
spisok  [5,7,1,13,9]
for a in spisok:
print(a*2)
print(spisok)
```

Функция range()

"Range" переводится как "диапазон". Она может принимать один, два или три аргумента. Если задан только один, то генерируются числа от 0 до указанного числа, не включая его. Если заданы два, то числа генерируются от первого до второго, не включая его. Если заданы три, то третье число – это шаг.

Оператор else в цикле for

В языке Python можно использовать оператор else вместе с циклами. Если оператор else используется с циклом for - он будет выполнен, когда список для цикла будет завершен; при использовании else вместе с циклом while - он будет использован, когда условие примет значение ложь (false).

Для управления циклом можно использовать специальные операторы break и continue. Оператор break осуществляет выход из цикла. Оператор continue выполняет переход к следующей итерации цикла. Оператор break может использоваться, если в цикле

образуются условия, которые несовместимы с его дальнейшим выполнением.

Вопросы для закрепления:

1. Каково назначение range()?

2. Чем отличаются операторы while и for?

3. Объясните различие continue и break.

Литература

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017.— 624с.: ил.—(Мировой компьютерный бестселлер).

2. Гвидо ванн Россум и др. Язык программирования Python. / 2001 — 454 с.

3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лекция 4
Последовательности в Python. Строки. Срезы. Операторы общие для всех типов последовательностей

План:

Операции со строками. Срезы (SLICING). Форматирование. Методы.

Строки в Python _ упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

При этом отдельного символьного типа в Python нет, символ _ это строка длины 1. Более того, символы как элементы строки тоже являются строками.

Существуют несколько различных подходов к операциям над строками.

• . Арифметические операции. Для строк подобно числам определены операторы сложения + и умножения *. В результате сложения содержимое двух строк записывается подряд в новую строку, например:

```
>>> S1 = 'Py '
>>> S2 = 'thon '
>>> S3 = S1 + S2
>>> print(S3)
Python
```

Можно складывать несколько строк подряд.

Умножение определено для строки и целого положительного числа, в результате получается новая строка, повторяющая исходную столько раз, каково было значение числа (возьмём строку S3 из прошлого примера):

```
>>> S3 * 4
' PythonPythonPythonPython '
>>> 2 * S3
' PythonPython'
```

65624

• Функция `len()` вычислят длину строки, результат имеет целочисленный тип. Например, `len('Python')` выдаст 6.

• Доступ по индексу. Можно обратиться к любому элементу (символу) строки по его номеру, нумерация начинается с 0 (первый элемент строки S имеет номер 0, последний _ len(S)-1. Разрешается использовать отрицательные индексы, в этом случае нумерация происходит с конца, что можно также интерпретировать как правило: к отрицательным индексам всегда добавляется длина строки, например последний элемент строки чаще всего обозначают как -1):

```
>>> S = 'Python '
>>> S [0]
'P'
>>> S[ -1]
'n'
```

Обращение к символу с несуществующим номером порождает ошибку:

```
«IndexError: string index out of range».
```

При использовании индексов необходимо помнить, что строки в Python относятся к категории неизменяемых последовательностей: нельзя поменять значение того или иного символа, а можно лишь создать новую строку.

```
>>> S = 'Ура'
>>> S[1] = 'x'
Traceback ( most recent call last ):
File "<pyshell #68 >", line 1, in <module >
S[1] = 'x'
TypeError : 'str ' object does not support
item assignment
>>> S = S [0]+ 'x'+S[2]
>>> S
'Ухa'
```

• Срезы позволяют скопировать или использовать в выражениях часть строки. Оператор извлечения среза из строки выглядит так: S[n1:n2]. N1 – это индекс начала среза, а n2 – его окончания, причем символ с номером n2 в срез уже не входит! Если указан отрицательный индекс, это значит, что любой индекс –n аналогичен записи len(s)-n. Если отсутствует первый индекс, то срез берётся от начала до второго индекса; при отсутствии второго индекса срез берётся от первого индекса до конца строки:

```
··· Day = 'morning ,_ afternoon ,_ evening '
··· Day [0:7]
'morning '
··· Day [9: -9]
'afternoon '
··· Day [ -7:]
'evening '
```

• Оператор in позволяет узнать, принадлежит ли подстрока в строке. Оператор возвращает логическое значение: True, если элемент в составе строки встречается и False, если нет:

```
··> S = 'Python '
··> SubS = 'th '
··> SubS in S
True
```

• Функции min и max применимы также и к строкам: max(s) определяет и выводит (возвращает) символ с наименьшим кодом – номером в кодовой таблице. Например:

```
·>> S = 'Python '
·>> min(S)
'P'
```

Возвращает символ с наибольшим значением (кодом).
Например:

```
·>> S = 'Python '
·>> max(S)
'y'
```

Таблица 2. Базовые операции над строками

Операция	Описание
`S1 + S2`	Объединение двух или более строк в новую строку.
`S * n`	Умножение строки на целое число n _ многократное повторение строки.
`Len(S)`	Функция, вычисляющая длину строки S.
`S [n]`	Доступ по индексу (номеру) к любому символу строки.
`S[n1:n2:n3]`	Срез _ новая строка, являющаяся частью исходной и содержащая символы с номерами от n1 включительно до n2 невключительно, если n3 присутствует (может не быть), то берутся не все символы, а с шагом n3.
`S2 in S1`	Логический оператор, проверяющий, является ли строка S2 частью строки S1.
`Min(S)`	Функция, вычисляющая символ строки S с наименьшим кодом.
`Max(S)`	Функция, вычисляющая символ строки S с наибольшим кодом.

Бывают методы, как описанный выше, не требующие вовсе никаких аргументов, бывают с одни аргументом, например метод S1.endswith(S2) требует 1 аргумент _ строку _ и проверяет, заканчивается ли строка S1 строкою S2. Бывают методы с двумя аргументами, например S1.replace(S2, S3), который заменяет в исходной строке S1 содержащуюся в ней подстроку S2 новою подстрокою S3 и выдаёт новую строку, при этом S1 остаётся неизменною. Более полную информацию о строковых методах можно получить, введя в интерактивном режиме команду help(str).

```
>>>
```

```
>>> colors = ['red', 'blue', 'green', 'black', 'white']
>>> colors.append('pink')
>>> colors
['red', 'blue', 'green', 'black', 'white', 'pink']
>>> colors.pop() # удаляет и возвращает последний объект
'pink'
>>> colors
['red', 'blue', 'green', 'black', 'white']
>>> colors.extend(['pink', 'purple']) # расширьте цвета, оперативные
>>> colors
```

```
['red', 'blue', 'green', 'black', 'white', 'pink', 'purple']
>>> colors = colors[:-2]
>>> colors
['red', 'blue', 'green', 'black', 'white']
```

Вперед:

```
>>> colors = colors[::-1]
>>> colors
['white', 'black', 'green', 'blue', 'red']
>>> colors2 = list(colors)
>>> colors2
['red', 'blue', 'green', 'black', 'white']
>>> colors2.reverse() # in-place
>>> colors2
['white', 'black', 'green', 'blue', 'red']
```

Скопируйте и повторите списки:

```
>>> colors = colors
['white', 'black', 'green', 'blue', 'red', 'red', 'blue', 'green', 'black',
'white']
>>> colors * 2
['white', 'black', 'green', 'blue', 'red', 'white', 'black', 'green', 'blue',
'red']
```

Или

```
>>> sorted(colors) # новый объект
['black', 'blue', 'green', 'red', 'white']
>>> colors
['white', 'black', 'green', 'blue', 'red']
>>> colors.sort() # оперативный
>>> colors
['black', 'blue', 'green', 'red', 'white']
```

Методы и объектно-ориентированное программирование

Нотация rcolors.method () (например, rcolors.append (3) и colors.pop ()) является нашим первым Пример объектно-ориентированного программирования (ООР). Будучи списком, объект rcolors владеет функцией метода, которая вызвана с помощью нотации. Никакое изучение ООП, без необходимого понимания нотации.

Обнаружение методов:

Reminder: in Ipython: tab-completion (press tab)

```
In [28]: rcolors.<TAB>
rcolors.append  rcolors.index   rcolors.remove
rcolors.count   rcolors.insert  rcolors.reverse
rcolors.extend  rcolors.pop     rcolors.sort
```

Строки

Различные строковые синтаксисы (простые, удваиваются или тройные кавычки):

```
s = 'Hello, how are you?'
s = "Hi, what's up"
s = '''Hello, how are you'''        # утроение кавычек позволяет
                                    # представьте в виде строки для охвата
больше чем одной строки
s = """Hi,
what's up?"""
In [1]: 'Hi, what's up?'
```

```
  File "<ipython console>", line 1
    'Hi, what's up?'
         ^
SyntaxError: invalid syntax
```

Этой синтаксической ошибки можно избежать путем включения строки в двойных кавычках вместо одинарных кавычек. С другой стороны, можно предварительно ожидать обратную косую черту к второй одинарной кавычке. Другое использование обратной косой черты, например, символ новой строки \n и символ табуляции \t.

Строки являются наборами как списки. Следовательно они могут быть индексированы и нарезаны, с помощью того же синтаксиса и правил.

Индексация:

```
>>>
... a = "hello"
... a[0]
'h'
... a[1]
'e'
... a[-1]
'o'
```

(Помните, что отрицательные индексы соответствуют подсчету с другого конца.)

Разрезание:

```
>>>
... a = "hello, world!"
... a[3:6] # 3-й к 6-м (исключенным) элементам: элементы 3, 4, 5
'lo,'
... a[2:10:2] # Синтаксис: a[start:stop:step]
'lo o'
... a[::3] # каждые три символа, с начала до конца
'hl r!'
```

Диакритические знаки и специальные символы могут также быть обработаны в строках Unicode (см. https://docs.python.org/tutorial/introduction.html #unicode-strings).

Строка является неизменным объектом, и не возможно изменить свое содержание. Можно однако создать новые строки из исходной.

```
In [83]: a = "hello, world!"
In [84]: a[2] = 'z'
```

```
Traceback (новый вызов в последний раз):
  File «<stdin>», line 1, in <module>
    TypeError: 'str' объект не поддерживает присвоение объекта

In [55]: a.replace('l', 'z', 1)
Out[55]: 'hezlo, world!'
In [56]: a.replace('l', 'z')
Out[56]: 'hezzo, worzd!'
```

```
>>> 'An integer: %i; a float: %f; another string: %s' % (1, 0.1, 'string')
'An integer: 1; a float: 0.100000; another string: string'

>>> i = 102
>>> filename = 'processing_of_dataset_%d.txt' % i
>>> filename
'processing_of_dataset_102.txt'
```

Словари

Словарь является в основном эффективной таблицей, которая отображает ключи к значениям. Это – незаказанный контейнер

```
>>>
>>> tel = {'emmanuelle': 5752, 'sebastian': 5578}
>>> tel['francis'] = 5915
>>> tel
{'sebastian': 5578, 'francis': 5915, 'emmanuelle': 5752}
>>> tel['sebastian']
5578
>>> tel.keys()
['sebastian', 'francis', 'emmanuelle']
>>> tel.values()
[5578, 5915, 5752]
>>> 'francis' in tel
True
```

Строка представляет последовательность символов в кодировке Unicode, заключенных в кавычки (одинарные или двойные кавычки, причем кавычки одного типа могут быть произвольно вложены в кавычки другого типа) с произвольным доступом. Строки в языке Python невозможно изменить – в этом случае говорят, что это immutable тип. Попытка изменить символ в определенной позиции или подстроке вызовет ошибку.

Операции:

1. Объединение или конкатенация (+).
2. Преобразование типа: число в строку (функция str())
3. Умножение строки (*).

Пример: str='группа' * 3
print(str)
Ответ: группагруппагруппа

4. Эскейп-последовательности

Управляющие эскейп-последовательности: *\n* представляет перевод строки, *\t* табляция, *кавычки,* чтобы отобразить кавычки (как двойные, так и одинарные) внутри строки, перед ними ставится слеш:

Пример: print("Кафе \"Aqtobe\"")
Ответ: Кафе "Aqtobe"

5. Сравнение строк (<, <=, ==, !=, >, >=).

При сравнении строк принимается во внимание символы и их регистр. Так, цифровой символ условно меньше, чем любой алфавитный символ. Алфавитный символ в верхнем регистре условно меньше, чем алфавитные символы в нижнем регистре.

Функция lower() приводит строку к нижнему регистру, а функция upper() - к верхнему.

Срезы (SLICING).

Срез – это механизм гибкого управления строкой на основе индексации. Можно получить любой символ строки по его индексу. Первый символ имеет индекс 0. Подстрока может быть определена с помощью среза – двух индексов, разделенных двоеточием.

Форматирование

Форматирование в Python – мощный инструмент управления строками. Для форматирования в Python строках используется стандартный оператор – символ %. Слева от процента указывают строку, справа – значение или список значений.

Методы

Строки обладают большим набором разнообразных методов.

1. find – находит подстроку в строке – возвращает позицию вхождения строки, либо -1

2. join – объединяет через разделитель набор строк

3. split – это обратная функция для join, разбивает строку на последовательность.

4. replace – заменяет в строке одну подстроку на другую.

5. strip – удаляет пробелы слева и справа

6. Для конверсии различных типов в строковый тип используются функции str, int, ord, chr:

str – конвертирует число в строку;

int – конвертирует строку в число;

ord – возвращает значение байта;

chr – конвертирует число в символ.

Вопросы для закрепления:

1. Что представляют собой строки в изучаемом языке программирования?

2. Какие операции над строками вы знаете?

3. Для чего предназначены срезы?

Литература

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017.— 624с.: ил.—(Мировой компьютерный бестселлер).

2. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лекция 5
Кортежи. Методы кортежей

План:
1.Понятие кортежа. Методы кортежей
2. Оператор присваивания

Кортежи - это неизменяемые (или иммутабельные) двоюродные братья списков.

Практически все, что можно делать со списком, не внося в него изменения, можно делать и с кортежем. Вместо квадратных скобок кортеж оформляют круглыми скобками, или вообще обходятся без них:

```
my_tuple    (1, 2) # задать кортеж
other_tuple = 3, 4 # еще один кортеж
try:
my_tuple [1] = 3
except TypeError :
print ( "кортеж изменять нель зя" )
```

Кортежи обеспечивают удобный способ для возвращения из функций нескольких значений:

```
# функция возвращает сумму и произведение двух параметров
def sum_and_product (x, y ) :
return (x + y) , (x * y)
sp = sum_and_product (2, 3) # = (5, 6)
s, p = sum_and_product (S, 10) # s = 15, p = 50
```

Базовые операторы кортежей

Кортежи работают с операторами + и * так же, как и строки - они используются для конкатенации и повторения.

Кортежи в Python имеют только два метода: index() и count().

Функции:

1 len(tuple) - количество элементов в кортеже;
2 max(tuple) - получить наибольший элемент кортежа;
3 min(tuple) - получить наименьший элемент кортежа;
4 tuple(seq) - превратить список в кортеж;

Сложные кортежи

Кортеж может содержать в себе другой кортеж или любой сложный элемент.

```
>>> nested = (1, "do", ["param", 10, 20])
```

Можно изменить список внутри кортежа:

```
>>> nested[2][1] = 15
>>> nested
(1, 'do', ['param', 15, 20])
```

Элементы кортежа записаны между круглыми скобками, или просто разделены запятыми:

Список так же может быть неизменяемым, как и строка, в этом случае он называется кортеж (tuple). Кортеж использует меньше памяти, чем список. При задании кортежа вместо квадратных скобок используются круглые (хотя можно и совсем без скобок). Кортеж не допускает изменений, в него нельзя добавить новый элемент, удалить или заменить существующие элементы, но он может содержать изменяемые объекты, например, списки:

```
>>>
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> u = (0, 2)
```

Наборы: незаказанный, уникальные объекты:

```
>>>
>>> s = set(('a', 'b', 'c', 'a'))
>>> s
set(['a', 'c', 'b'])
>>> s.difference(('a', 'b'))
set(['c'])
```

Оператор присваивания

Основное различие между кортежами и списками состоит в том, что кортежи не могут быть изменены. На практике это означает, что у них нет методов, которые бы позволили их изменить: append(), extend(), insert(), remove(), pop().

Но можно взять срез от кортежа, так как при этом создастся новый кортеж.

Кортежи в некоторых случаях быстрее, чем списки. Но такие оптимизации в каждом конкретном случае требуют дополнительных исследований. Кортежи делают код безопаснее в том случае, если у вас есть .защищенные от записи.

Данные, которые не должны изменяться. Некоторые кортежи могут использоваться в качестве элементов множества и ключей словаря (конкретно, кортежи, содержащие неизменяемые значения, например, строки, числа и другие кортежи).

Списки никогда не могут использоваться в качестве ключей словаря потому что списки изменяемые объекты.

В справочном руководстве по библиотеке Python говорится:

Операторы присваивания привыкли к (ре), связывают имена к значениям и изменить атрибуты или объекты изменяемых объектов.

Короче говоря, это работает следующим образом (простое присвоение):

1. выражение на правой стороне оценено, соответствующий объект создается/получается

2. имя на левой стороне присвоено или связано к объекту r.h.s.

Вещи отметить:

- a single object can have several names bound to it:
- In [1]: a = [1, 2, 3]
- In [2]: b = a
- In [3]: a
- Out[3]: [1, 2, 3]
- In [4]: b
- Out[4]: [1, 2, 3]
- In [5]: a is b
- Out[5]: True
- In [6]: b[1] = 'hi!'
- In [7]: a
- Out[7]: [1, 'hi!', 3]

- to change a list *in place*, use indexing/slices:

- In [1]: a = [1, 2, 3]
- In [3]: a
- Out[3]: [1, 2, 3]
- In [4]: a = ['a', 'b', 'c'] # *Creates another object.*
- In [5]: a
- Out[5]: ['a', 'b', 'c']
- In [6]: id(a)
- Out[6]: 138641676
- In [7]: a[:] = [1, 2, 3] # *Modifies object in place.*
- In [8]: a
- Out[8]: [1, 2, 3]
- In [9]: id(a)
- Out[9]: 138641676 # *Same as in Out[6], yours will differ...*

• ключевое понятие здесь изменяемо по сравнению с неизменным

• изменяемые объекты могут быть изменены на месте

• неизменные объекты не могут быть изменены когда-то созданные

Управляет порядком, в котором выполнен код.

If/elif/else

```
>>>
```

```
>>> if 2**2 == 4:
...     print('Obvious!')
...
Obvious!
```

Блоки разграничены добавлением отступа

Введите следующие строки в своем интерпретаторе Python и старайтесь уважать глубину добавления отступа. Оболочка Ipython автоматически увеличивает глубину добавления отступа после двоеточия: знак; для уменьшения глубины добавления отступа пойдите четыре пробелов налево с клавишей Backspace. Нажмите клавишу Enter дважды для отступа логического блока.

```
>>> n = 10

>>> if n == 1:
...     print(1)
... elif n == 2:
...     print(2)
... else:
...     print('A lot')
A lot
```

Добавление отступа обязательно в сценариях также. Как осуществление, перепечатайте предыдущие строки с тем же добавлением отступа в сценарии condition.py и выполните сценарий с выполненным condition.py в Ipython.

1.2.3.2. for/range

Iterating with an index:

```
>>>
... for i in range(4):
...     print(i)
0
1
2
3
```

Но чаще всего, это более читаемо для итерации по значениям:

```
>>>
... for word in ('cool', 'powerful', 'readable'):
...     print('Python is %s' % word)
Python is cool
Python is powerful
Python is readable
```

while/break/continue

Типичный цикл с условием продолжения С-стиля (проблема Mandelbrot):

```
>>>
>>> z = 1 + 1j
>>> while abs(z) < 100:
...     z = z**2 + 1
>>> z
(-134+352j)
```

Больше расширенных функций
Break изключите из циклов for/while loop:

```
>>>
>>> z = 1 + 1j

>>> while abs(z) < 100:
...     if z.imag == 0:
...         break
...     z = z**2 + 1
```

Продолжите следующее повторение цикла:

```
>>>
>>> a = [1, 0, 2, 4]
>>> for element in a:
...     if element == 0:
...         continue
...     print(1. / element)
1.0
0.5
0.25
```

Вопросы для закрепления:

1. Что такое кортеж?
2. Чем кортеж отличается от строки?
3. Какие методы, применимые к кортежам, Вы знаете?

Литература

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017. 624с.: ил.—(Мировой компьютерный бестселлер).
2. Гвидо ванн Россум и др. Язык программирования Python. / 2001 454 с.
3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.
4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.
5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.
6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека АЛТ).

Лекция 6
Списки. Специальные операторы и функции для работы со списками. Вложенные списки

План: Понятие списка. Способы задания списка. Методы списков

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

Чтобы использовать списки, их нужно создать. Создать список можно несколькими способами:

1) с помощью встроенной функцией list.

2) при помощи литерала.

3) с помощью генераторов списков. Генератор списков - способ построить новый список, применяя выражение к каждому элементу последовательности.

Таблица "методы списков"

Метод	Что делает
list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет на i-ый элемент значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
list.count(x)	Возвращает количество элементов со значением x
list.sort([key=функция])	Сортирует список на основе функции
list.reverse()	Разворачивает список
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

Методы списков, в отличие от строковых методов, изменяют сам список, и потому результат выполнения не нужно записывать в эту переменную.

Вопросы для закрепления:

1. Что такое список?
2. Объясните назначение каждого метода, применимого к списку
3. В чем заключаются особенности работы с вложенными списками?

Литература

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017. — 624с.: ил.—(Мировой компьютерный бестселлер).
2. Гвидо ван Россум и др. Язык программирования Python. / 2001. — 454 с.
3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.
4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.
5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.
6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО, МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека АЛТ)

Лекция 7.
Определение функций

План:

Понятие функции. Параметры и аргументы. Вызовы функций. Оператор возврата return. Лямбда-функции. Рекурсия.

Функция в Python - объект, принимающий аргументы и возвращающий значение. Обычно функция определяется с помощью инструкции def:

```
def add(x, y):
return x + y
```

Инструкция return возвращает значение. В примере функция возвращает сумму x и y.

Вызов:

```
>>>
>>> add(1, 10)
11
>>> add('abc', 'def')
'abcdef'
```

Функция может быть любой сложности и возвращать любые объекты (списки, кортежи, и даже функции). Функция может и не заканчиваться инструкцией return, при этом функция вернет значение None.

Аргументы функции

Функция может принимать произвольное количество аргументов или не принимать их вовсе. Также распространены функции с произвольным числом аргументов, функции с позиционными и именованными аргументами, обязательными и необязательными.

Функция также может принимать переменное количество позиционных аргументов, тогда перед именем ставится *.

Функция может принимать и произвольное число именованных аргументов, тогда перед именем ставится **.

Python поддерживает так называемые анонимные, или лямбда-функции. По существу, это простые однострочные функции, возвращающие значение. Определяются они с помощью ключевого

слова lambda, которое означает «мы определяем анонимную функцию» и ничего более.

```
def short_function(x):
return x * 2
equiv_anon = lambda x: x * 2
```

Они особенно удобны в ходе анализа данных, потому что во многих случаях функции преобразования данных принимают другие функции в качестве аргументов. Часто быстрее (и чище) передать лямбда-функцию, чем писать полноценное объявление функции или даже присваивать лямбда-функцию локальной переменной.

Рекурсия – определение объекта через самого себя. При создании рекурсивных функций нужно сначала определить условие выхода из рекурсии.

Вопросы для закрепления:

1. Каковы отличительные особенности рекурсии?
2. Как создать и вызвать функцию пользователя?
3. Для чего необходимы функции в Python? Лямбда-функции?

Литература

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017.— 624с.: ил.—(Мировой компьютерный бестселлер).
2. Гвидо ванн Россум и др. Язык программирования Python. / 2001 454 с.
3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.
4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.
5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.
6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лекция 8
Операции над множествами.
Работа со словарями. Методы словарей

План: Множество. Словари.

Структура данных set или множество представляет собой совокупность неупорядоченных элементов без повторов:

```
s = set ()     # задать пустое множество
s.add (1)      # теперь s = { 1 }
s.add (2)      # теперь s = { 1, 2 }
s.add (2)      # s как и прежде = { 1, 2 }
x = len (s)    # = 2
y = 2 in s     # = True
z = 3 in s     # = False
```

Словарь или ассоциативный список - это еще одна основная структура данных. В нем значения связаны с ключами, что позволяет быстро извлекать значение, соответствующее конкретному ключу:

```
empty_dict = {}                    # задать словарь по-питоновски
empty_dict2 = dict ()              # не совсем по-питоновски
grades = { "Grigoriy" : 80, "Tim" : 95 } # литерал словаря (оценки за экзамены)
```

Доступ к значению по ключу можно получить при помощи квадратных скобок:

```
rigory_aleksee = grades[ "Grigoriy"]          # = 80
```

При попытке запросить значение, которое в словаре отсутствует, будет выдано сообщение об ошибке KeyError.

Словари имеют метод get(), который при поиске отсутствующего ключа вместо вызова исключения возвращает значение по умолчанию.

Присваивание значения по ключу выполняется при помощи квадратных скобок.

Словари часто используются в качестве простого способа представить структурные данные. Ключи должны быть неизменяемыми; в частности, в качестве ключей нельзя использовать списки. Если нужен составной ключ, то лучше воспользоваться кортежем или же найти способ, как преобразовать ключ в строку.

Словарь defaultdict (словарь с значением по умолчанию). Он похож на обычный словарь за исключением одной особенности - при попытке обратиться к ключу, которого в нем нет, он сперва добавляет для него значение, используя функцию без аргументов, которая предоставляется при его создании. Чтобы воспользоваться словарями defaultdict, их необходимо импортировать из модуля collections: from collections import defaultdict

Словарь Counter. Подкласс словарей counter трансформирует последовательность значений в похожий на словарь defaultdict(int) объект, где ключам поставлены в соответствие частотности или, выражаясь более точно, ключи отображаются (map) в частотности. Словарь counter располагает методом most_common( ), который нередко бывает полезен:

```
# напечатать 10 наиболее встречаемых слов и их частотность (встречаемость)
for word, count in word_counts.most_common(10) :
print (word, count )
```

Вопросы для закрепления:

1. Чем множество отличается от словаря?
2. Перечислите операции над множествами.
3. Что такое словарь?
4. В чем заключается особенность словарей со значением по умолчанию?

Литература

1. Гвидо ванн Россум и др. Язык программирования Python. / 2001 – 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лекция 9
Файлы. Операции с файлами

План: Открытие файла. Чтение/запись в файл. Произвольный доступ к компоненте файла. .csv файлы.

Прежде, чем работать с файлом, его надо открыть:

```
f = open('text.txt', 'r')
```

У функции open много параметров, важны 3 аргумента: первый, это имя файла. Путь к файлу может быть относительным или абсолютным. Второй аргумент, это режим, в котором будет открыт файл

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'x'	открытие на запись, если файла не существует, иначе исключение.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.
't'	открытие в текстовом режиме (является значением по умолчанию).
'+'	открытие на чтение и запись

Режимы могут быть объединены, то есть, к примеру, 'rb' - чтение в двоичном режиме. По умолчанию режим равен 'rt'.

И последний аргумент, encoding, нужен только в текстовом режиме чтения файла. Этот аргумент задает кодировку.

Чтение из файла

У метод read, читающий весь файл целиком, если был вызван без аргументов, и n символов, если был вызван с аргументом целым числом n).

```
>>> f = open('text.txt')
>>> f.read(1)
'H'
>>> f.read()
'ello world!\nThe end.\n\n'
```

2. прочитать файл построчно, воспользовавшись циклом for: for line in f:

Существует много способов чтения входного файла:

file.read() - возвращает строку, содержащую все символы (или байты), хранящиеся в файле.

file.read(N) - возвращает строку, содержащую очередные N символов (или байтов), из файла.

file.readline() - читает содержимое файла до ближайшего символа \n и возвращает строку.

file.readlines() - читает файл целиком и возвращает список строк

Вызов метода seek(0) перед каждой попыткой чтения переустанавливает текущую позицию чтения в начало файла.

Запись в файл

Открыть файл на запись: f = open('text.txt', 'w')

Если полный путь к файлу не указан, то файл будет создан в текущем рабочем каталоге. Если в этом каталоге уже существует такой файл, то он будет удален и создан новый. С режимом "w" нужно быть осторожнее, можно потерять содержимое старого файла.

Запись в файл осуществляется с помощью метода write:

```
>>>
>>> for index in l:
...     f.write(index + '\n')
...
```

Метод write возвращает число записанных символов. Метод write объекта файл записывает в точности то, что ему передано, без дополнительного форматирования. Поэтому мы явно добавляем символ конца строки.

Для записи в файлы можно также использовать метод writelines, который просто записывает все строки из списка без дополнительного форматирования.

```
    myfile = open("C:\\projects\\data.txt", "w")
myfile.writelines(["# Name Email Phone\n",
                   "Larry larry@example.com 111-1111\n",
                   "Curly curly@example.com 222-2222\n",
                   "Moe moe@example.com 333-3333\n"])
myfile.close()
```

Произвольный доступ к данным в файлах

При открытии файлов текущая позиция обычно устанавливается в смещение 0 от начала файла и перемещается вперед по мере чтения/записи данных.

Метод seek позволяет переместить текущую позицию для следующей операции чтения/записи в другое место, для чего ему достаточно передать величину смещения в байтах.

seek(n)
seek(n, mode)
mode 0 абсолютное смещение на n байтов (по умолчанию)
mode 1 смещение относительно текущей позиции на n байтов
mode 2 смещение относительно конца файла на n байтов

Закрытие файла: f.close()

Файлы CSV (Comma Separated Values)

Файл CSV – это особый вид файла, который позволяет структурировать большие объемы данных.

По сути, он является обычным текстовым файлом, однако каждый новый элемент отделен от предыдущего запятой или другим разделителем.

Обычно каждая запись начинается с новой строки. Данные CSV можно легко экспортировать в электронные таблицы или базы данных.

Библиотека CSV - основная библиотека для работы с CSV файлами в Python.

Она является встроенной: import csv

Чтение из файлов (парсинг)

Для того чтобы прочитать данные из файла, нужно создать объект reader:

reader_object = csv.reader(file, delimiter = ",")

reader имеет метод __next__(), то есть является итерируемым объектом.

Запись в файлы

Для записи информации в CSV файл необходимо создать объект writer:

file_writer = csv.writer(w_file, delimiter = "\t")

Для записи в файл данных используется метод writerow(), который имеет следующий синтаксис:

writerow("Имя", "Фамилия", "Отчество")

Вопросы для закрепления:

1. Что мы понимаем под файлом?
2. Каков алгоритм работы при создании нового файла?
3. Всегда ли нужно закрывать открытый файл?
4. Чем отличается текстовый файл от csv-файла?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ
2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018
3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).
4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лекция 10
Модули и пакеты в Python. Основные стандартные модули и пакеты в Python. Импортирование модулей. Создание собственных модулей и их импортирование

План: Понятие модуля. Создание и использование. Пространство имен.

Модуль в языке Python представляет отдельный файл с кодом, который можно повторно использовать в других программах.

Для создания модуля необходимо создать файл с расширением py, который будет представлять модуль. Название файла будет представлять название модуля. Затем в этом файле надо определить одну или несколько функций.

Для использования модуля его надо импортировать с помощью оператора import, после которого указывается имя модуля: import MyModul. Можно подключить несколько модулей, подключаемые модули можно указывать через запятую или же каждый модуль в отдельной строке.

Чтобы обращаться к функциональности модуля, нужно получить его пространство имен. По умолчанию оно будет совпадать с именем модуля.

Получив пространство имен модуля, можно обратиться к его функциям по схеме пространство_имен.функция.

Настройка пространства имен

По умолчанию при импорте модуля он доступен через одноименное пространство имен. Можно переопределить это поведение с помощью ключевого слова as. Например:

```
import account as acc
.....
result = acc.calculate_income(rate, money, period)
```

В данном случае пространство имен будет называться acc.

Другой вариант настройки предполагает импорт функциональности модуля в глобальное пространство имен текущего модуля с помощью ключевого слова from:

```
from account import calculate_income
.....
 result = calculate_income(rate, money, period)
```

В данном случае мы импортируем из модуля account в глобальное пространство имен функцию calculate_income. Поэтому мы сможем ее использовать без указания пространства имен модуля, как если бы она была определена в этом же файле.

Если бы в модуле account было бы несколько функций, то могли бы их импортировать в глобальное пространство имен одним выражением:

```
from account import *
 #.....
 result = calculate_income(rate, money, period)
```

Извлечение определённой функции

В случае когда не нужен весь модуль, а лишь нужны отдельные функции модуля, нужно извлечь эти функции. Для этого нужно прописать служебное слово from, затем указать имя модуля, потом ввести import, а после, через запятую, прописать все необходимые функции модуля:

```
from math import pi   #извлечение функции выводящей число Пи
print(pi)             #на экран будет выведено число Пи
from math import pi as var_pi   извлечение   функции   выводящей
число Пи
print(var_pi)                   #на экран будет выведено число Пи
```

Местонахождение модулей в Python.

Когда импортируется модуль, интерпретатор Python ищет этот модуль в следующих местах:

- Директория, в которой находится файл, в котором вызывается команда импорта
- Если модуль не найден, Python ищет в каждой директории, определенной в консольной переменной PYTHONPATH.
- Если и там модуль не найден, Python проверяет путь, заданный по умолчанию

Путь поиска модулей сохранен в системном модуле sys в переменной path. Переменная sys.path содержит все три вышеописанных места поиска модулей.

Многократное использование кода: сценарии и модули

План:

1. Усовершенствованное повторение
2. Понимания списка

Тезис

Усовершенствованное повторение

Выполните итерации по любой последовательности

Можно выполнить итерации по любой последовательности (строка, список, вводит словарь, строки в файле, …):

```
>>>
... vowels = 'aeiouy'

... for i in 'powerful':
...     if i in vowels:
...         print(i)
o
e
u
>>>
... message = "Hello how are you?"
... message.split() # returns a list
['Hello', 'how', 'are', 'you?']
... for word in message.split():
...     print(word)
...
Hello
how
are
you?
```

Немного языков (в частности, языков для научных вычислений) позволяют циклично выполняться по чему-либо кроме целых чисел/индексов. С Python возможно циклично выполниться точно по предметам интереса, не беспокоясь индексами, о которых Вы часто не заботитесь. Эта функция может часто использоваться для создания кода более читаемым.

• Отслеживание числа перечисления

• Общая задача состоит в том, чтобы выполнить итерации по последовательности при отслеживании номер изделия.

• Мог использовать цикл с условием продолжения со счетчиком как выше. Или для цикла:

```
>>>
>>> words = ('cool', 'powerful', 'readable')
>>> for i in range(0, len(words)):
...     print((i, words[i]))
(0, 'cool')
(1, 'powerful')
(2, 'readable')
```

•• Но, Python обеспечивает, встроенная функция – перечисляют – для этого:

```
>>>
>>> for index, item in enumerate(words):
...     print((index, item))
(0, 'cool')
(1, 'powerful')
(2, 'readable')
```

Цикличное выполнение по словарю

Используйте объекты:

```
>>>
>>> d = {'a': 1, 'b':1.2, 'c':1j}
```

```
... for key, val in sorted(d.items()):
...     print('Key: %s has value: %s' % (key, val))
Key: a has value: 1
Key: b has value: 1.2
Key: c has value: 1j
```

Примечание

Упорядочивание словаря случайно, таким образом мы используем отсортированный (), который отсортирует на ключах.

2. Понимания списка

```
>>>
>>> [i**2 for i in range(4)]
[0, 1, 4, 9]
```

Когда Вы выходите из интерпретатора и приезжаете снова, все определения, сделанные Вами (функции, переменная), потеряны навсегда. Поэтому, если Вы хотите ввести немного длинной программы, для подготовки входа лучше использовать текстовый редактор. В процессе увеличения программы Вы, возможно, хотите повредить его в несколько файлов, что они были легче поддерживаться. Можно хотеть использовать полезные функции, написанные Вами в нескольких программах, не копируя их определение в каждой из программ.

Python позволяет помещать определения в файл и использовать их в программах и интерактивном режиме. Такой файл называют модулем. Определения от модуля могут быть импортированы в другие модули и в основной модуль (набор переменных, доступных в программе и в интерактивном режиме).

Создание и использование модулей

Модуль — файл, содержащий определения и другие инструкции языка Python. Имя файла формируется путем добавления к названию модуля суффикса (расширение) на.r'. В модуле, его зовут доступном в глобальном peremenno __, называют . Для Пример, с помощью любимого текстового редактора, создают в текущем каталоге файл с именем 'fibo.py' следующего содержания:

```
''\
Generation and output of numbers of Fibonacci
'''
def fib(n):
'''Displays the sequence of the numbers of
Fibonacci which are not exceeding n '''
a, b = 0, 1
while b <n:
print b,
 a, b = b, a +b
def fib2 (n):
''' returns the list containing the numbers of a
number of Fibonacci which are not exceeding n '''
result = []
a, b = 0, 1
while b < n:
result.append(b)
a, b = b, a+b
return result
```

Теперь запустите интерпретатор и импортируйте просто созданный модуль:

```
>>> import fibo
```

Эта инструкция не вводит имена функций, определяемых в fibo непосредственно в текущем пространстве имен, это только вводит fibo имя модуля. Используя имя модуля, можно получить доступ к функциям:

```
>>> fibo.fib (1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2 (100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo. Name__
'fibo'
```

Если Вы собираетесь использовать функцию часто, может импортировать его локальной переменной:

```
>>> fib = fibo.fib
>>> fib (500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Модуль может содержать любые инструкции, предназначенные для его инициализации, и не только определения функций. Они выполняются, только когда модуль импортируется в первый раз.

Каждый модуль имеет собственное пространство имен, которое является глобальной областью видимости для всех функций, определяемых в нем. Таким образом автор модуля может использовать глобальные переменные, не волнуясь о возможных конфликтах с глобальными переменными пользователя. С другой стороны, если Вы знаете, что делаете, можно получить доступ к глобальным переменным модуля таким же образом относительно его функций (на самом деле, функции являются переменными также), modname. Itemname.

Модули могут импортировать другие модули. Обычно инструкция импорта имеет в начале модуля или программы. Названия импортированных модулей расположены в текущем пространстве имен модуля импорта или программы.

Другая версия инструкции импортирует имена импорта из модуля непосредственно к текущему пространству имен:

```
>>> from fibo import fib, fib2 >>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

В этом случае название модуля не будет представлено в текущей области видимости (в данном Пример, имя fibo не определяется).

Еще одна версия импорта инструкции позволяет импортировать все имена oprekdelenny в модуле, кроме имен, начинающихся с символа подчеркивания (* _'):

```
>>> from fibo import *
>>> fib (500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Часто существует потребность импортировать модуль или объект модуля, использующего для него локальное имя кроме начальной буквы. Для Пример следующий код позволяет заменять строку имени _string (который не будет импортирован инструкцией ‘из импорта my_module *’) при записи модуля:

```
import string
_string = string
del string
```

Устанавливает еще один Пример, поскольку возможно избежать конфликта имени, определенного в модуле со встроенным именем:

```
import anydbm
dbopen = anydbm.open
```

При запуске с версии 2.0 подобные операции могут быть сделаны очень проще (и bezopaskny) благодаря расширению синтаксиса инструкции `import`:

```
import string as _string
from anydbm import open as dbopen
```

Необходимо заметить, что как не зарезервированное слово, и можно все еще определить переменные с таким именем.

Если название модуля, который должен быть импортирован, становится известным только во времени выполнения программа, можно использовать инструкцию exec (‘должностное лицо 'импорт' + module_name’) или встроенная функция __ import __ ()

Поиск модулей

Для ускорения запуска программ с помощью большого количества модулей, если уже существует файл с именем 'spam.pyc' в том же каталоге, где ‘spam.py’ найден, считается, что это содержит «байт – скомпилированный» модуль спама. Если нет такого файла, то это создается, и время последнего изменения

регистров 'spam.py' в созданном 'spam.pyc' (при последующем использовании, '.pyc '-the файл проигнорирован, если начальная буква к.г' файл была изменена).

Обычно Вы ничего не должны делать для создания 'spam.pyc'. Как только 'spam.py' успешно компилируется, интерпретатор пытается записать скомпилированную версию в 'spam.pyc'. Если интерпретатор по каким-либо причинам не управляет им (для Пример, недостаточного количества пользовательских полномочий), ошибка не возникает. Если файл записан не полностью, далее он распознан, как отключено и проигнорировано. Содержание байтов – скомпилированные файлы platformno-независимы (но может быть разное для различных версий интерпретатора), таким образом, каталог с модулями может быть совместно использован машинами с различной архитектурой.

Если интерпретатор вызывается с опцией – Об или переменная среды, PYTHONOPTIMIZE имеет непустое значение, интерпретатор генерирует оптимизированный байт-код и сохраняет его в '.pyo '-файлах. В данный момент оптимизация дает очень мало: одновременно инструкции утверждают, удалены, инструкции 'тем, если _debug ___ являются ignored:..., информация не сохранена о нумерациях строк в начальной букве к.г '-файлы. В этом случае все используемые модули оптимизированы, '.pyc' – файлы проигнорированы.

Опция 00 приводит к тому, что интерпретатор выполняет оптимизацию, которая, в некоторых (редких) случаях, может привести к отказам во времени выполнения программ. В данный момент, в дополнение к операциям, выполненным с опцией – О, строки документации удалены, дав более компактные '.pyo '-файлы. Поскольку некоторые программы могут ожидать существование строк документации, использовать эту опцию с осторожностью.

Для программы, запущенной из командной строки, код байта никогда не регистрируется в '.pyc '-or '.pyo '-the файл. Поэтому, если Вы хотите уменьшить время, я требую щееся для загрузки, помещаю большую часть кода в модуле, оставив только часть начальной загрузки, которая импортирует этот модуль в программе.

Использование модуля (или запускаются программы) в присутствии '.pyc' – файл (или '.pyo '-the файл, если одна из опций О или 00 используется) возможен, даже если отсутствует к.г' –

файл. Таким образом можно расширить библиотеки и программы во взгляде, от которого довольно трудно получить информацию об используемых алгоритмах.

Compileall модуль позволяет создавать '.pyc '-(или \rua '-) файлы для всех модулей в каталоге. Может быть особенно полезно, если Вы хотите ограничить доступ к каталогу, в котором существует библиотека. Давайте заметим, что интерпретатор не будет использовать '.pyc' – файлы, если это будет запущено с включенной оптимизации и '.pyo' файлов, если оптимизация выключена (если отсутствует к.г' – файл, модуль будет недоступен).

Вопросы для закрепления:

1. Что такое модуль и каково его назначение?
2. Можно ли создать собственный модуль? Если можно, то как им пользоваться?
3. Какие стандартные модули языка Вы знаете?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ
2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018
3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / . Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).
4. Уэс Маккинли. Python и анализ данных / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лекция 11
Матрицы. Модуль Numpy.
Обработка данных с применением массивов

План: Модуль Numpy. Массивы в Python. Объект ndarray и его атрибуты. Операции над массивами.

Numerical Python, или сокращенно NumPy – краеугольный пакет для высокопроизводительных научных расчетов и анализа данных.

В большинстве приложений для анализа данных основной интерес представляет следующая функциональность:

- быстрые векторные операции для переформатирования и очистки данных, выборки подмножеств и фильтрации, преобразований и других видов вычислений;
- стандартные алгоритмы работы с массивами, например: фильтрация, удаление дубликатов и теоретико-множественные операции;
- эффективная описательная статистика, агрегирование и обобщение данных;
- выравнивание данных и реляционные операции объединения и соединения разнородных наборов данных;
- описание условной логики в виде выражений-массивов вместо циклов с ветвлением if-elif-else;
- групповые операции с данными (агрегирование, преобразование, применение функции).

Одна из ключевых особенностей NumPy – объект ndarray для представления N-мерного массива; это быстрый и гибкий контейнер для хранения больших наборов данных в Python. Массивы позволяют выполнять математические операции над целыми блоками данных, применяя такой же синтаксис, как для соответствующих операций над скалярами. ndarray – это обобщенный многомерный контейнер для однородных данных, т. е. в нем могут храниться только элементы одного типа. У любого массива есть атрибут shape – кортеж, описывающий размер по каждому измерению, и атрибут dtype – объект, описывающий тип данных в массиве.

Создать массив можно с помощью функции array. Помимо array, существует еще ряд функций для создания массивов. Например, zeros и ones создают массивы заданной длины, состоящие из нулей и единиц соответственно, а shape.empty создает массив, не инициализируя его элементы.

Наиболее важные атрибуты объектов ndarray:

ndarray.ndim - число измерений (чаще их называют "оси") массива.

ndarray.shape - размеры массива, его форма. Это кортеж натуральных чисел, показывающий длину массива по каждой оси. Для матрицы из n строк и m столбов, shape будет (n,m). Число элементов кортежа shape равно ndim.

ndarray.size - количество элементов массива. Очевидно, равно произведению всех элементов атрибута shape.

ndarray.dtype - объект, описывающий тип элементов массива. Можно определить dtype, используя стандартные типы данных Python. NumPy здесь предоставляет целый букет возможностей, как встроенных, например: bool_, character, int8, int16, int32, int64, float8, float16, float32, float64, complex64, object_, так и возможность определить собственные типы данных, в том числе и составные.

ndarray.itemsize - размер каждого элемента массива в байтах.

ndarray.data - буфер, содержащий фактические элементы массива. Обычно не нужно использовать этот атрибут, так как обращаться к элементам массива проще всего с помощью индексов.

Базовые операции

Операции между массивами и скалярами. Массивы важны, потому что позволяют выразить операции над совокупностями данных без выписывания циклов for. Обычно это называется векторизацией.

Математические операции над массивами выполняются поэлементно. Создается новый массив, который заполняется результатами действия оператора. Для этого массивы должны быть одинаковых размеров.

Одномерные массивы осуществляют операции индексирования, срезов и итераций очень схожим образом с обычными списками и другими последовательностями Python (удалять с помощью срезов нельзя).

У многомерных массивов на каждую ось приходится один индекс. Индексы передаются в виде последовательности чисел, разделенных запятыми (кортежами).

Манипуляции с формой – shape().

Объединение массивов - hstack(), vstack(), column_stack(), row_stack().

Разбиение массива - hsplit(), vsplit(),array_split().

Поверхностная копия - view().

Глубокая копия - copy().

Определение класса и создание копии класса

Класс описан посредством ключевого слова класса согласно следующей схеме:

```
class <Название класса>[(<Класс1>[,  <
КлассN>])]:
[«»» Строка документирования «»»]
<Описание атрибутов и методов>
```

Инструкция создает новый объект и адаптирует ссылку на нее к идентификатору, определенному после ключевого слова класса. Это означает, что название класса должно соответствовать правилам именования переменных полностью. После названия класса возможно определить один или несколько базовых классов через запятую в круглых скобках. Если класс не наследовал базовые классы, то круглые скобки не могут быть определены. Необходимо заметить, что все выражения в инструкции класса выполняются во время создания класса, но не его копии. Для Пример мы создадим класс, в котором сообщение (просто отображен список 1.).

Листинг 1. Создание определенного класса

```
# coding: utf-8 class MyClass:
"""Это строка документирования «»»
print(«Инструкции выполняются сразу»)
input()
```

Этот Пример содержит только определение класса MyClass и не создает копию класса. Как только поток выполнения достигает

инструкций класса, сообщение, определенное как печать (), сразу будет отображен.

Создание атрибута класса подобно созданию нормальной переменной. Метод в классе создается, а также обычная функция — посредством инструкции определения. Ссылка на копию класса автоматически передается методам класса в первом параметре, который, конечно, должен быть определен, очевидно. Это стандартно этот параметр для называния именем сам, хотя это и не обязательно. Доступ к атрибутам и методам класса в определенном методе сделан через сам переменная посредством нотации точки — для приписывания x из метода класса, к которому возможно обратиться так: сам. X.

Для использования атрибутов и методов класса необходимо создать копию класса согласно следующему синтаксису:

< Class copy > = < Name of a class >([< Parameters >]>

At the appeal to methods of a class such format is used:

< Class copy >.< Method name > ([<Parameters >])

Обратите внимание, что в вызове метода не необходимо передать ссылку на копию класса как параметр, поскольку это становится в определении метода в классе. Интерпретатор передает ссылку на копию класса автоматически.

Обращение к атрибутам класса выполняется так же:

< Class copy >.< Attribute name >

Давайте определим класс MyClass с атрибутом x и печатью x метод (), отображая значение этого атрибута, и затем мы создадим копию класса, и мы назовем метод (список 2.).

Список 2. Создание атрибута и метода

```
   class MyClass:
def     init (self): # Designer
    self.x = 10    #    Attribute of a copy of a
class
```

```
def print_x(self):#    self is a ssypka on a
class copy
print(self.x)      #    We display value of
attribute
c = MyClass ()     #    Creation of a copy of a
class
                   #    We call a method print_x()
c.print_x()        #    self it is not specified
at a method call
print(c.x)         #    It is possible to address
attribute directly
```

Для доступа к атрибутам и методам возможно использовать также следующие функции:

♦ getattr() – значение возвратов атрибута его именем установлено в форме строки. Посредством этой функции возможно создать название атрибута динамично во времени выполнения программа. Функциональный формат:

```
getattr(<Object>, <Attribute>[, <Value by
default>])
```

Если указанный атрибут не найден, исключение AttributeError возбуждено. Для предотвращения вывода сообщения об ошибке возможно определить значение, которое возвратится, если атрибут не будет существовать в третьем параметре;

♦ setattr() – устанавливает значение атрибута. Название атрибута определяется в форме строки.

Формат функции:

```
setattr(<Объ Objectект>, <Attribute>, <Value>)
```

Это – возможный setattr () для передачи вторым параметром метода названия несуществующего атрибута — в этом случае, атрибут с введенным именем будет создан;

- delattr (<Objectъект>, <Attribute>) — удаляет указанный атрибут. Название атрибута определяется в форме строки;
- hasattr (<Object>, <Attribute>) — существование проверок указанного атрибута. Если атрибут существует, функциональное верное значение возвратов.

Давайте покажем работу функций на примере (листинг 3.).

Functions getattr(),setattr() и hasattr()

```
class MyClass:
def  init  (self) :
self.x = 10
def get_x(self):
return self.x
c = MyClass()              # We create a class copy
print(getattr(c,"x"))      # Will display: 10
print(getattr(c,"get_      # Will display: 10
x") ())
print(getattr(c,"y",       # Will display: 0, since the
0))                        attribute is not found
setattrtc, "y",20)         # We create attribute y
print(getattr(c,"y",       # Will display: 20
0))
delattr(c, "y")            # We delete attribute y
print(getattr(c,"y",       # Will display: 0, since the
0))                        attribute is not found
print(hasattr(c,"x"))      # Will display: True
print(hasattr(c,"y")
)                          # Will display: False
```

Все атрибуты класса на языке Python открыты (общественность), т.е. доступны прямому изменению и от класса, и от других классов и от основного кода программы.

Кроме того, атрибутам позволяют быть созданными динамично после создания класса — возможно создать и атрибут объекта класса и атрибут копии класса.

Вот маленький пример: мы создаем Студенческий класс, который является объектом, собирающим несколько пользовательских функций (методы) и переменные (атрибуты), мы будем в состоянии использовать:

```
>>>
>>> class Student(object):
...     def __init__(self, name):
...         self.name = name
...     def set_age(self, age):
...         self.age = age
...     def set_major(self, major):
...         self.major = major
...
>>> anna = Student('anna')
>>> anna.set_age(21)
>>> anna.set_major('physics')
```

В предыдущем Пример Студенческий класс имеет __ init __, set_age и set_major методы. Его атрибуты являются именем, возрастом и майором. Мы можем назвать эти методы и атрибуты со следующей нотацией: classinstance.method или classinstance.attribute. __ init __ конструктор является специальным методом, с которым мы звоним: MyClass (init параметры если таковые имеются).

Теперь, предположите, что мы хотим создать новый класс MasterStudent с теми же методами и атрибутами как предыдущий, но с дополнительным атрибутом стажировки. Мы не скопируем предыдущий класс, но inheritfrom это:

```
>>>
>>> class MasterStudent(Student):
...     internship = 'mandatory, from March to June'
...
>>> james = MasterStudent('james')
>>> james.internship
'mandatory, from March to June'
>>> james.set_age(23)
>>> james.age
23
```

Класс MasterStudent наследован от Студенческих атрибутов и методов.

Благодаря классам и объектно-ориентированному программированию, мы можем организовать код с различными классами, соответствующими различным объектам, с которыми мы встречаемся (класс Эксперимента, класс Изображения, класс Потока, и т.д.), с их собственными методами и атрибутами. Тогда мы можем использовать наследование для рассмотрения изменений вокруг кода повторного использования и базового класса. Исключая: от базового класса Потока мы можем создать, получил StokesFlow, TurbulentFlow, PotentialFlow, и т.д..

Что является массивами NumPy и NumPy?

Массивы NumPy

Объекты Python:	• • высокоуровневые объекты числа: целые числа, плавающая точка • • контейнеры: списки (бесплатная вставка и добавляют), словари (быстрый поиск) • • пакет расширения к Python для многомерных массивов
NumPy обеспечивает:	• • ближе к аппаратным средствам (эффективность) • • разработанный для научных расчетов (удобство) • • Также известный, поскольку массив ориентировал вычисление

```
>>>
```

```
>>> import numpy as np
>>> a = np.array([0, 1, 2, 3])
>>> a
array([0, 1, 2, 3])
```

Для примера, массив, содержащий:

• • значения эксперимента/моделирования на шагах дискретного времени

• • сигнал, зарегистрированный устройством измерения, например, звуковой волной

• • пиксели изображения, уровня серого или цвета

• • 3-D данные, измеренные в различных X-Y-Z положениях, например, сканировании МРТ

Почему это полезно: эффективный памятью контейнер, который обеспечивает быстро числовые операции.

```
In [1]: L = range(1000)

In [2]: %timeit [i**2 for i in L]
1000 loops, best of 3: 403 us per loop

In [3]: a = np.arange(1000)

In [4]: %timeit a**2
100000 loops, best of 3: 12.7 us per loop
```

NumPy Reference documentation

- On the web: http://docs.scipy.org/
- Interactive help:

```
- In [5]: np.array?
- String Form:<built-in function array>
- Docstring:
- array(object, dtype=None, copy=True, order=None, subok=False, ndmin=0, ...
```

- Looking for something:

```
>>>
>>> np.lookfor('create array')
Search results for 'create array'
```

numpy.array

Create an array.

Numpy.memmap

Create a memory-map to an array stored in a *binary* file on disk.

```
In [6]: np.con*?
np.concatenate
```

```
np.conj
np.conjugate
np.convolve
```

Соглашения импорта

Рекомендуемое соглашение импортировать numpy:

```
>>>
```

```
>>> import numpy as np
```

1.Creating arrays

Manual construction of arrays

- 1-D:

```
>>>
```

```
>>> a = np.array([0, 1, 2, 3])
>>> a
array([0, 1, 2, 3])
>>> a.ndim
1
>>> a.shape
(4,)
>>> len(a)
4
```

- 2-D, 3-D, …:

```
>>>
```

```
>>> b = np.array([[0, 1, 2], [3, 4, 5]])    # 2 x 3 array
>>> b
array([[0, 1, 2],
       [3, 4, 5]])
>>> b.ndim
2
>>> b.shape
(2, 3)
>>> len(b)     # returns the size of the first dimension
2

>>> c = np.array([[[1], [2]], [[3], [4]]])
>>> c
array([[[1],
```

```
       [2]],

       [[3],
        [4]]])
... c.shape
(2, 2, 1)
```

• Осуществление: Простые массивы

•• Создайте простую двумерную матрицу. Во-первых, восстановите Примерs сверху. И затем создайте свое собственное: как насчет нечетных чисел, считающих в обратном направлении на первой строке и четных числах на втором?

•• Используйте функции len (), numpy.shape () на этих массивах. Как они касаются друг друга? И к ndim атрибуту массивов?

2. Функции для создания массивов

• На практике мы редко вводим объекты один за другим ...

•• Равномерно расположенный с интервалами:

```
>>>
>>> a = np.arange(10) # 0 .. n-1  (!)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> b = np.arange(1, 9, 2) # start, end (exclusive), step
>>> b
array([1, 3, 5, 7])
```

• or by number of points:

```
>>>
>>> c = np.linspace(0, 1, 6)   # start, end, num-points
>>> c
array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ])
>>> d = np.linspace(0, 1, 5, endpoint=False)
>>> d
array([ 0. ,  0.2,  0.4,  0.6,  0.8])
```

• Common arrays:

```
>>>
```

```
>>> a = np.ones((3, 3))  # reminder: (3, 3) is a tuple
>>> a
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
>>> b = np.zeros((2, 2))
>>> b
array([[ 0.,  0.],
       [ 0.,  0.]])
>>> c = np.eye(3)
>>> c
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
>>> d = np.diag(np.array([1, 2, 3, 4]))
>>> d
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

- np.random: random numbers (Mersenne Twister PRNG):

```
>>>
>>> a = np.random.rand(4)       # uniform in [0, 1]
>>> a
array([ 0.95799151,  0.14222247,  0.08777354,  0.51887998])

>>> b = np.random.randn(4)      # Gaussian
>>> b
array([ 0.37544699, -0.11425369, -0.47616538,  1.79664113])

>>> np.random.seed(1234)        # Setting the random seed
```

Осуществление: Создание массивов с помощью функций

• Эксперимент с расположением, linspace, нулями, глазом и диагональю.

• Создайте различные виды массивов со случайными числами.

• Попытайтесь установить семя прежде, чем создать массив со случайными значениями.

• Посмотрите на функцию np.empty. Что это делает? Когда это могло бы быть полезно?

3. Типы основных данных

Вы, возможно, заметили, что в некоторых случаях элементы массива отображены с запаздывающей точкой (например, 2. по сравнению с 2). Это происходит из-за различия в используемом типе данных:

```
>>>
... a = np.array([1, 2, 3])
... a.dtype
dtype('int64')

... b = np.array([1., 2., 3.])
... b.dtype
dtype('float64')
```

Различные типы данных позволяют нам хранить данные более сжато в памяти, но большую часть времени мы просто работаем с числами с плавающей точкой. Обратите внимание, что, в Пример выше, NumPy автоматически обнаруживает тип данных от входа.

Можно явно определить, какой тип данных Вы хотите:

```
>>>
... c = np.array([1, 2, 3], dtype=float)
... c.dtype
dtype('float64')
```

The default data type is floating point:

```
>>>
... a = np.ones((3, 3))
... a.dtype
dtype('float64')
```

There are also other types:

Комплексные:

```
>>>
>>> d = np.array([1+2j, 3+4j, 5+6*1j])
>>> d.dtype
dtype('complex128')
```

Булевые:

```
>>>
>>> e = np.array([True, False, False, True])
>>> e.dtype
dtype('bool')
```

Строковые:

```
>>>
>>> f = np.array(['Bonjour', 'Hello', 'Hallo'])
>>> f.dtype    # ←- strings containing max. 7 letters
dtype('S7')
```

Намного больше:

- int32
- int64
- uint32
- uint64

4. Основная визуализация

Теперь, когда у нас есть наши массивы First Data, мы собираемся визуализировать их.

Запустите путем запуска Ipython:

$ ipython

Или ноутбук:

$ ipython ноутбук

Как только Ipython запустился, включите интерактивные графики:

```
>>>
>>> %matplotlib
```

Или от ноутбука включите графики в ноутбуке:

```
>>>
>>> %matplotlib inline
```

Встроенное важно для ноутбука, так, чтобы графики были отображены в ноутбуке а не в новом окне.

Matplotlib является 2D пакетом графического изображения. Мы можем импортировать его функции как ниже:

```
>>>
... import matplotlib.pyplot as plt  # the tidy way
```

И затем используйте (обратите внимание, что необходимо использовать шоу явно, если Вы не включили интерактивные графики с %matplotlib):

```
>>>
... plt.plot(x, y)      # line plot
... plt.show()          # ← shows the plot (not needed with interactive plots)
```

Вопросы для закрепления:

1. Дайте определение понятию массив. Особенности массива в Python?
2. Перечислите атрибуты объекта ndarray?
3. Что мы понимаем под векторизацией?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ
2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018
3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).
4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лекция 12
Модуль Numpy. Файловый ввод-вывод массивов

План: Работа с файлами в Python. Особенности файлов в Numpy.

NumPy умеет сохранять на диске и загружать с диска данные в текстовом или двоичном формате.

Хранение массивов на диске в двоичном формате

np.save и np.load – основные функции для эффективного сохранения и загрузки данных с диска. По умолчанию массивы хранятся в несжатом двоичном формате в файле с расширением .npy.

Можно сохранить несколько массивов в zip-архиве с помощью функции np.savez, которой массивы передаются в виде именованных аргументов:

np.savez('array_archive.npz', a=arr, b=arr)

При считывании npz-файла мы получаем похожий на словарь объект, который отложенно загружает отдельные массивы.

Загрузка текста из файлов – вполне стандартная задача. Иногда бывает полезно загружать данные в массивы NumPy с помощью функции np.loadtxt или более специализированной np.genfromtxt.

У этих функций много параметров, которые позволяют задавать разные разделители и функции-конвертеры для столбцов, пропускать строки и делать много других вещей.

Функция np.savetxt выполняет обратную операцию: записывает массив в текстовый файл с разделителями. Функция genfromtxt аналогична loadtxt, но ориентирована на структурные массивы и обработку отсутствующих данных.

Ввод/вывод

Существует несколько способов представления вывода программы: данные могут быть распечатаны во взгляде, удобном для восприятия, или записаны в файле для дальнейшего использования. Отформатированный вывод

Тем не менее мы использовали два способа вывода: вывод значений выражений в интерактивном режиме и посредством инструкции печати (третий путь — метода объектов – файлы записи ()).

Часто существует требование иметь больший контроль над форматированием вывода, чем только для отображения значений, разделенных на пространство. Конечно, Вы можете производственные линейки: посредством операций сокращения и ассоциации возможно создать любое расположение, что только Вы будете в состоянии представить. Строки имеют методы, позволяющие дополнять их пробелами до необходимой ширины kolonki1. — для использования другого пути оператор % со строкой как левый аргумент. Оператор % интерпретирует строку как строку формата в стиле sprintf О функция справа в, С которыми потребностями быть примененным к правильному аргументу, и возвращает строку с результатом форматирования.

Конечно, существует еще один вопрос: как получить впечатление строки для значений другого типа? К счастью, Python дает возможность преобразовать значение любого типа к строке: посредством функции str (). На самом деле язык обеспечивает две функции для получения представления строки — repr О (тот же эффект может быть получен просто завершавший выражение в кавычках возврата: 'expr') и str (). Первый путь, для Пример, используется интерпретатором для вывода значений выражений в интерактивном режиме, втором — для вывода аргументов инструкция печати. Функция str () каждый раз, когда возможные возвраты представление, самое подходящее для вывода и функция repr () — для входа выражения в интерактивном режиме. Давайте дадим несколько примеров:

```
>>> x = 10 * 3.14
```

Номер 31.4 не может быть точно представлен в двоичной форме поэтому:

```
>>> x
31.399999999999999
```

Однако функция str () отобразит число с разумной точностью:

```
>>> y = 200*200
>>> s = 'Значение x равно ' + str (x) + \
```

```
…', значение y равно ' + str(y) + '…'
>>> print s
Значение x равно 31.4, значение y равно 40000…
```

Длинные целые регистра числа на языке Python с суффиксом 'L'. Начиная версию 1.6, функция str () не приносит ему:

```
>>> repr (1000L)
' 1000L'
>>> str(1000L)
'1000'
```

Впечатление строки может быть получено также для других типов:

```
>>> p = [x, y]
>>> ps = repr(p)
>>> ps
'[31.399999999999999, 40000]'
>>> 'x, y, ('spam', 'eggs')'
«(31.399999999999999, 40000, ('spam', 'eggs'))»
```

Функция repr () (или'') добавляет кавычку и пишет специальные символы посредством управляющих последовательностей:

```
>>> hello = 'hello, world\n'
>>> print hello hello, world
>>> hellos = 'hellos
>>> print hellos 'hello, world\012'
```

Давайте произведем таблицу квадратов и кубов двумя описанными способами:

```
>>> for x in ranged, 11):
...     print str(x) .rjust(2) , str (x*x) .rjust
(3),
…       # Обратите внимание на завершающую запятую
…       print str(x*x*x) .rjust (4)
```

```
1  1   1
2  4   8
3  9   27
4  16  64
5  25  125
6  36  216
7  49  343
8  64  512
9  81  729
       100
10 100 0
>>> for x in range(1,11):
...     print '%2d %3d %4d* % (x, x*x, x*x*x)
1  1   1
2  4   8
3  9   27
4  16  64
5  25  125
6  36  216
7  49  343
8  64  512
9  81  729
       100
10 100 0
```

(Обратите внимание, что одно пространство между столбцами было добавлено инструкцией печати.)

Этот Пример показывает использование метода крика строк (), который выравнивает строку в области ширины набора направо, дополняя его с пробелами слева. Так же ljust работа методов () и центр (). Они не отображаются, что-либо — просто возвращает новую строку. Если начальная строка слишком долго, это не отключено и возвращается в постоянном взгляде: обычно ввести разупорядочение в расположение столбцов лучше, чем отобразить неправильное значение. (Если Вы действительно хотите отключить его, используйте операцию сокращения: 's.ljust (n) [0: n]'.)

Также функция zfill может быть полезной (), определенный в строковом модуле который дополнения с нулем строка с числом слева, правильно обработав знаки «плюс» и минус:

```
>>> import string
>>> string.zfill ('12 ', 5)
'00012'

>>> string.zfill('-3.14', 7) '-003.14'
»> string.zfill('3.14159265359', 5)
'3.14159265359'
```

Использование оператора % смотрит приблизительно так:

```
>>> import math
>>> print 'The PI value is approximate equally
in %5.3f.' % \
.           math.pi
The PI value is approximate equally in 3.142.
```

Если в строке необходимо отобразить несколько значений как правильный операнд, поезд используется:

```
>>> table = {'Sjoerd      ': 4127,
          'Jack'          : 4098,
          'Dcab'          : 7678}
»> for    name, phone     in table.items():
          print '%-10s    %10d' % (name,
Sjoerd                    4127
Dcab                      7678
Jack                      4098
```

Большинство форматов работает таким же образом в S. Odnako, если типы аргументов не соответствуют формату, интерпретатор, приводит их к необходимому типу (для Пример, выражение любого типа может быть преобразовано к строке посредством встроенной функции str ()), или если это невозможно, генерирует исключение. Можно использовать * для передачи отдельной шириной поля параметра или точностью.

Давайте заметим это, если правильный аргумент поезд, это всегда считают списком аргументов:

```
··· def f(x):
...     print 'The value is transferred to function
"%s"' % x
»· f (1)
The value is transferred to function "1"
»· f ([ 1, 2])
The value is transferred to function "[1, 2]"
·»> # it is interpreted not as you expected. . .
f((1, ) )
The value is transferred to function "1"

·»> # ошибка ... f ( (1, 2))
Traceback (most recent call last):
File "<stdin>", line 1, in ?
File "<stdin>", line 2, in f TypeError: not all
arguments converted
```

В этом случае это будет более надежно для использования поезда, состоящего из одного элемента как правильный операнд:

```
>>> def f(x):
...     print 'The value is transferred to function
"%s"' % (x,)
>>> # now everything is correct... f(<1,))
The value is transferred to function " (1,)"
»» f < <1, 2))
The value is transferred to function "(1, 2)"
```

В случае длинных линий формата можно хотеть обратиться к переменным по имени вместо их ситуации. Это может быть сделано, с помощью расширенной записи в форме % (название) формата, для примера:

```
>>> table = {'Sjoerd': 4127, 'Jack': 4098,
'Dcab': 8637678}
>>> print 'Jack: %(Jack)d; Sjoerd: %(Sjoerd)d; \
... Dcab: %(Dcab)d' % table
Jack: 4098; Sjoerd: 4127; Dcab: 8637678
```

Такая запись особенно полезна в комбинации со встроенной функцией Вара (), который возвращает словарь с переменными в текущей области видимости.

Читайте и запись файлов

Встроенная функция открытых () возвращает объектный файл (файл) и обычно используется двумя аргументами: 'открытый (имя файла, режим)'.

```
>>> f=open ('/tmp/workfile ', 'wb')
>>> print f
<open file '/tmp/workfile', mode 'wb' at 80a0960>
```

Первый аргумент — строка, содержащая имя файла, второй аргумент — строка, soderkzhashchy несколько символов, описывая режим использования файла. Режим может быть, " если файл откроется только для чтения, 'то w' — только для записи (существующий файл будет пфезаписан), и 'и' — для dopisyvaniye в конце файла. В режиме '+' файл открывается для чтения и записи сразу. Аргумент режима не обязателен: если это понижено, это предназначено ".

В Windows (и в определенных случаях и в Macintosh) файлы по умолчанию открываются в текстовом режиме — для открытия файла в режиме двоичного счета, необходимо добавить к строке режима 'Ь'. Необходимо помнить, что будут повреждены двоичные данные, такие как изображения в формате JPEG и даже текст в UNICODE, при чтении из файла или записи файла, открытой в текстовом режиме! Лучший способ защитить себя от проблем — всегда для открытия файлов в режиме двоичного счета, даже на тех платформах, где режим двоичного счета используется по умолчанию (возможно, у Вас когда-то будет требование запустить программу на другой платформе).

Обязательные параметры (позиционные параметры)

```
In [81]: def double_it(x):
   ....:     return x * 2
   ....:

In [82]: double_it(3)
```

```
Out[82]: 6

In [83]: double_it()

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: double_it() takes exactly 1 argument (0 given)
```

Дополнительные параметры (ключевое слово или параметры, передаваемые по имени)

```
In [84]: def double_it(x=2):
  ....:     return x * 2
  ....:

In [85]: double_it()
Out[85]: 4

In [86]: double_it(3)
Out[86]: 6
In [124]: bigx = 10

In [125]: def double_it(x=bigx):
  .....:     return x * 2
  .....:

In [126]: bigx = 1e9  # Now really big

In [128]: double_it()
Out[128]: 20
```

Используя изменяемый тип в аргументе ключевого слова (и изменение его в теле функции):

```
In [2]: def add_to_dict(args={'a': 1, 'b': 2}):
  ...:     for i in args.keys():
  ...:         args[i] += 1
  ...:     print args
```

```
  ...:

In [3]: add_to_dict
Out[3]: <function __main__.add_to_dict>

In [4]: add_to_dict()
{'a': 2, 'b': 3}

In [5]: add_to_dict()
{'a': 3, 'b': 4}

In [6]: add_to_dict()
{'a': 4, 'b': 5}
```

Более включенный пример, реализовывая разрезание Python:

```
In [98]: def slicer(seq, start=None, stop=None, step=None):
  ....:     """Implement basic python slicing."""
  ....:     return seq[start:stop:step]
  ....:

In [101]: rhyme = 'one fish, two fish, red fish, blue fish'.split()

In [102]: rhyme
Out[102]: ['one', 'fish,', 'two', 'fish,', 'red', 'fish,', 'blue', 'fish']

In [103]: slicer(rhyme)
Out[103]: ['one', 'fish,', 'two', 'fish,', 'red', 'fish,', 'blue', 'fish']

In [104]: slicer(rhyme, step=2)
Out[104]: ['one', 'two', 'red', 'blue']

In [105]: slicer(rhyme, 1, step=2)
Out[105]: ['fish,', 'fish,', 'fish,', 'fish']

In [106]: slicer(rhyme, start=1, stop=4, step=2)
Out[106]: ['fish,', 'fish,']
```

Порядок аргументов ключевого слова не имеет значения:

```
In [107]: slicer(rhyme, step=2, start=1, stop=4)
Out[107]: ['fish,', 'fish,']
```

но это – хорошая практика для использования того же упорядочивания в качестве определения функции.

Аргументами ключевого слова является очень удобная функция определения функций с переменным количеством аргументов, особенно когда значения по умолчанию должны привыкнуть в большинстве вызовов к функции.

2. Передача значением

Можно ли изменить значение переменной в функции? Большинство языков (C, Java, ...) отличает "передачу значением" и "передачу ссылкой". В Python такое различие несколько искусственно, и это немного тонко, будут ли Ваши переменные измененными или нет. К счастью, там существуйте четкие правила.

Параметры к функциям являются Литература к объектам, которые передаются значением. При передаче переменной функции Python передает ссылку на объект, к которому переменная относится (значение). Не сама переменная.

Если значение, переданное в функции, неизменно, функция не изменяет переменную вызывающей стороны. Если значение изменяемо, функция может изменить оперативную переменную вызывающей стороны:

```
>>>
>>> def try_to_modify(x, y, z):
...     x = 23
...     y.append(42)
...     z = [99] # new reference
...     print(x)
...     print(y)
...     print(z)
...
>>> a = 77    # immutable variable
>>> b = [99]  # mutable variable
```

```
>>> c = [28]
>>> try_to_modify(a, b, c)
23
[99, 42]
[99]
>>> print(a)
77
>>> print(b)
[99, 42]
>>> printI
[28]
```

Функции имеют таблицу локальных переменных, названную локальным пространством имен.

Переменная x только существует в функции try_to_modify.

3. Глобальные переменные

На переменные, объявленные вне функции, можно сослаться в функции:

```
In [114]: x = 5

In [115]: def addx(y):
  .....:     return x + y
  .....:

In [116]: addx(10)
Out[116]: 15
```

Но эти "глобальные" переменные не могут быть изменены в функции, если не объявлено глобальной в функции.

Это не работает:

```
In [117]: def setx(y):
  .....:     x = y
  .....:     print('x is %d' % x)
  .....:
  .....:
In [118]: setx(10)
```

```
In [120]: x
Out[120]: 5
```

This works:

```
In [121]: def setx(y):
   .....:     global x
   .....:     x = y
   .....:     print('x is %d' % x)
   .....:
   .....:

In [122]: setx(10)
x is 10

In [123]: x
Out[123]: 10
```

Вопросы для закрепления:

1. Что мы понимаем под файлом? Чем отличаются текстовые файлы от бинарных?
2. Расскажите об особенностях работы с csv-файлами.
3. Какие функции и методы для работы с файлами в Numpy Вы знаете?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ
2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018
3. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).
4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лекция 13
Модуль Numpy. Линейная алгебра

План: Модуль numpy.linalg. Характеристики матриц. Системы уравнений.

Модуль numpy.linalg позволяет выполнить многие операции из линейной алгебры.

Возведение в степень

linalg.matrix_power(M, n) - возводит матрицу в степень n.

Разложения

linalg.cholesky(a) - разложение Холецкого.

linalg.qr(a[, mode]) - QR разложение.

linalg.svd(a[, full_matrices, compute_uv]) - сингулярное разложение.

Некоторые характеристики матриц

linalg.eig(a) - собственные значения и собственные векторы.

linalg.norm(x[, ord, axis]) - норма вектора или оператора.

linalg.cond(x[, p]) - число обусловленности.

linalg.det(a) - определитель.

linalg.slogdet(a) - знак и логарифм определителя (для избежания переполнения, если сам определитель очень маленький).

Системы уравнений

linalg.solve(a, b) - решает систему линейных уравнений Ax = b.

linalg.tensorsolve(a, b[, axes]) - решает тензорную систему линейных уравнений Ax = b.

linalg.lstsq(a, b[, rcond]) - метод наименьших квадратов.

linalg.inv(a) - обратная матрица.

linalg.LinAlgError - исключение, вызываемое данными функциями в случае неудачи (например, при попытке взять обратную матрицу от вырожденной).

Массивы большей размерности в большинстве функций linalg интерпретируются как набор из нескольких массивов нужной размерности. Таким образом, можно одним вызовом функции проделывать операции над несколькими объектами.

Функция	Описание
diag	Возвращает диагональные элементы квадратной матрицы в виде одномерного массива или преобразует одномерный массив в квадратную матрицу, в которой все элементы, кроме находящихся на главной диагонали, равны нулю
dot	Вычисляет произведение матриц
trace	Вычисляет след матрицы – сумму диагональных элементов
det	Вычисляет определитель матрицы
eig	Вычисляет собственные значения и собственные векторы квадратной матрицы
inv	Вычисляет обратную матрицу
pinv	Вычисляет псевдообратную матрицу Мура-Пенроуза для квадратной матрицы

Вопросы для закрепления:

1. Какие функции модуля numpy.linalg для работы с матрицами Вы знаете?

2. Перечислите функции, используемые для решения системы линейных алгебраических уравнений.

3. Какие исключения могут возникнуть при работе с модулем numpy.linalg?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лекция 14.
Графический интерфейс пользователя (GUI). Стандартные виджеты Tkinter. События и методы. Диалоговые окна

План: О графическом интерфейсе. Основы Tkinter. Классы виджетов. Конфигурирование виджетов. Понятие события. Содержание события. Диалоговые окна.

Почти все современные графические интерфейсы общего назначения строятся по модели WIMP — Window, Icon, Menu, Pointer (окно, иконка, меню, указатель). Основной задачей графического интерфейса является упрощение коммуникации между пользователем и компьютером. Основная черта любой программы с графическим интерфейсом — интерактивность. Программа не просто что–то считает (в пакетном режиме) от начала своего запуска до конца: ее действия зависят от вмешательства пользователя. Фактически, графическое приложение выполняет бесконечный цикл обработки событий. Программа, реализующая графический интерфейс, событийно–ориентирована. Она ждет от интерфейса событий, которые и обрабатывает сообразно своему внутреннему состоянию.

Эти события возникают в элементах графического интерфейса (виджетах) и обрабатываются прикрепленными к этим виджетам обработчиками. Сами виджеты имеют многочисленные свойства (цвет, размер, расположение), выстраиваются в иерархию принадлежности (один виджет может быть хозяином другого), имеют методы для доступа к своему состоянию.

Расположением виджетов (внутри других виджетов) ведают так называемые менеджеры расположения. Виджет устанавливается на место по правилам менеджера расположения. Эти правила могут определять не только координаты виджета, но и его размеры. В Tk имеются три типа менеджеров расположения: простой упаковщик (pack), сетка (grid) и произвольное расположение (place).

Классы виджетов

Для построения графического интерфейса в библиотеке Tk отобраны следующие классы виджетов (в алфавитном порядке):

• Button (Кнопка) Простая кнопка для вызова некоторых действий (выполнения определенной команды).

• Canvas (Рисунок) Основа для вывода графических примитивов.

• Checkbutton (Флажок) Кнопка, которая умеет переключаться между двумя состояниями при нажатии на нее.

• Entry (Поле ввода) Горизонтальное поле, в которое можно ввести строку текста.

• Frame (Рамка) Виджет, который содержит в себе другие визуальные компоненты.

• Label (Надпись) Виджет может показывать текст или графическое изображение.

• Listbox (Список) Прямоугольная рамка со списком, из которого пользователь может выделить один или несколько элементов.

• Menu (Меню) Элемент, с помощью которого можно создавать всплывающие (popup) и ниспадающие (pulldown) меню.

• Menubutton (Кнопка–меню) Кнопка с ниспадающим меню.

• Message (Сообщение) Аналогично надписи, но позволяет заворачивать длинные строки и менять размер по требованию менеджера расположения.

• Radiobutton (Селекторная кнопка) Кнопка для представления одного из альтернативных значений. Такие кнопки, как правило, действует в группе. При нажатии на одну из них кнопка группы, выбранная ранее, «отскакивает».

• Scale (Шкала) Служит для задания числового значения путем перемещения движка в определенном диапазоне.

• Scrollbar (Полоса прокрутки) Полоса прокрутки служит для отображения величины прокрутки в других виджетах. Может быть как вертикальной, так и горизонтальной.

• Text (Форматированный текст) Этот прямоугольный виджет позволяет редактировать и форматировать текст с использованием различных стилей, внедрять в текст рисунки и даже окна.

• Toplevel (Окно верхнего уровня) Показывается как отдельное окно и содержит внутри другие виджеты.

Создание виджета происходит вызовом конструктора соответствующего класса. Вызов конструктора имеет следующий синтаксис:

```
Widget([master[, option=value, ...]])
```

Здесь Widget - класс виджета, master - виджет–хозяин, option и value - конфигурационная опция и ее значение (таких пар может быть несколько).

Каждый виджет имеет свойства, которые можно устанавливать (конфигурировать) с помощью методов config() (или configure()) и читать с помощью методов, подобных методам работы со словарями.

В системе современного графического интерфейса имеется возможность отслеживать различные события, связанные с клавиатурой и мышью, и происходящие на «территории» того или иного виджета. В Tk события описываются в виде текстовой строки — шаблона события, состоящего из трех элементов (модификаторы, тип события и детализация события).

Тип события	Содержание события
Activate	Активизация окна
ButtonPress	Нажатие кнопки мыши
ButtonRelease	Отжатие кнопки мыши
Deactivate	Деактивация окна
Destroy	Закрытие окна
Enter	Вхождение курсора в пределы виджета
FocusIn	Получение фокуса окном
FocusOut	Потеря фокуса окном
KeyPress	Нажатие клавиши на клавиатуре
KeyRelease	Отжатие клавиши на клавиатуре
Leave	Выход курсора за пределы виджета
Motion	Движение мыши в пределах виджета
MouseWheel	Прокрутка колесика мыши
Reparent	Изменение родителя окна
Visibility	Изменение видимости окна

Диалоговые окна, как элементы графического интерфейса, предназначены для вывода сообщений пользователю, получения от него какой-либо информации, а также управления.

Диалоговые окна весьма разнообразны. Для того чтобы запрограммировать с помощью Tkinter вызов диалоговых окон открытия и сохранения файлов и работу с ними, требуется дополнительно импортировать "подмодуль" Tkinter - tkinter.filedialog, в котором описаны классы для окон данного типа.

Еще одна группа диалоговых окон описана в модуле tkinter.messagebox. Это достаточно простые диалоговые окна для вывода сообщений, предупреждений, получения от пользователя ответа "да" или "нет" и т. п.

Вопросы для закрепления:

1. Для чего необходим графический модуль?
2. Что такое виджет?
3. Что такое событие и как оно связано с виджетами?
4. Когда может возникнуть событие в системе? Какие события могут возникнуть?
5. Какие виды диалоговых окон вы знаете?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ
2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018
3. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров.- 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).
4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лекция 15.
Стандартная библиотека

План:

1. Переменное количество параметров
2. *Строки документа*
3. Функции являются объектами

Библиотека стандартных модулей

Описание языка Python было бы неполным без описания библиотеки — огромный набор модулей. Некоторые модули записаны на C и создаются в интерпретаторе, другие записаны на языке Python и доступны в начальном взгляде. Некоторые модули обеспечивают сервисную характеристику языка Python (для Пример, вывода сообщений об ошибках) или определенная операционная система (для Пример, доступа к определенным аппаратным средствам), другие определяют характеристику интерфейсов некоторого объема (для Пример, WWW).

Модули конфигурации

сайт — общая конфигурация

Этот модуль автоматически импортируется в каждом, запускаются интерпретатора (за исключением случаев, когда опция командной строки используется – S), также это предназначается, в первую очередь, для добавления способов поиска характеристики модулей этой машины. Каталогами по умолчанию с именами 'sys.prefix + '/lib/site-packages" 'и' sys.prefix + '/site-python1 '(UNIX) или sys.prefix (другие платформы) добавляются к способам поиска. Кроме того, модуль обрабатывает конфигурационные файлы способов взгляда‘ package.pxW во всех указанных каталогах. Эти конфигурационные файлы должны содержать дополнительные пути (согласно одному каталогу в строке), который будет включен в sys.path. Пустые строки и строки, начинающиеся “#’, проигнорированы.

Для Пример стандартные библиотеки, которым позволяют, установлены in/usr/lib/python1.5 каталог’, в котором существует подкаталог ‘пакеты сайта. Позвольте the/usr/lib/python1.5/site-packages каталогу в свою очередь содержать вложенное 'также',

'панель' и каталоги 'спама' и конфигурационные файлы 'foo.pth' и 'bar.pth'. Давайте примем, foo.pth файл содержит следующие строки:

```
# foo package configuration
foo
bar
bletch
```

and 'bar.pth' contains:

```
# bar package configuration
Bar
```

Тогда the/usr/lib/python1.5/site-packages/bar каталоги и '/usr/lib/python1.5/site-packages/foo будут добавлены к sys.path!.

Обратите внимание, что 'bletch' и каталоги 'спама' не добавляются, поскольку нет никакого файла или каталога с именем 'bletch', и каталог спама не упоминается в одном из конфигурационных файлов.

После выполнения перечисленных манипуляций с путями предпринята попытка импортировать sitecustomize модуль, в котором может выполнить дополнительные настройки. Если импортом исключение ImportError сгенерировано, это тихо проигнорировано.

Пользователь — конфигурация пользователя

Для Цель безопасности интерпретатор не выполняет автоматически конфигурационный файл пользователя. Только в интерактивном режиме интерпретатор выполняет файл, определенный в переменной среды PYTHONSTARTUP. Однако некоторые программы в состоянии позволить себе загрузить стандартный конфигурационный файл пользователя. Программа, желающая использовать такой механизм, должна выполнить инструкцию 'пользователя импорта'.

Пользовательский модуль ищет файл с именем '.pythonrc.py' в каталоге дома пользователя и выполняет его посредством встроенной функции execfileO в собственном (пользовательский модуль) глобальное пространство имен. Ошибки, возникающие при выполнении '.pythonrc.py' пользовательский модуль, не

обрабатываются. Если каталог дома не может быть определен, .pythonrc.py файл' в текущем каталоге выполняется.

Будьте ограничены, что Вы помещаете в .pythonrc.py файле'. Поскольку Вы не знаете, какие программы будут использовать его, изменение поведения стандартных модулей и функций едва будет хорошей идеей.

Рекомендация для программистов, желающих использовать механизм, описанный здесь: самый простой способ установить параметры — для определения переменных в .pythonrc.py файле'. Для Пример модуль спама может определить, как подробный пользователь хочет видеть его сообщения посредством следующих инструкций:

```
import user try:
#     значение, установленное пользователем
verbose = user.spam_verbose
except AttributeError:
#     значение по умолчанию verbose = 0
```

Программы, имеющие большое количество пользовательских параметров, должны использовать отдельный конфигурационный файл. Программы относительно безопасности и также модулей общего Цель, не должен импортировать модуль `user`.

Модули Office

Модули, являются офисом относительно интерпретатора языка Python и его взаимодействия со средой.

`Sys`	Доступ к параметрам и функциям, характеристике системы.
`Atexit`	Регистры функционируют и выполняют их при завершении работы программы
`types`	Названия всех встроенных типов.
`Operator`	Операторы языка Python в форме функций.
`Traceback`	Модуль для работы с объектами traceback.
`Imp`	Доступ к операциям сделан импортом инструкции.
`Pprint`	Представление и вывод данных в более привлекательном взгляде.

Repr	Альтернативная реализация функции repr () с ограничением размера.

Работа со строками

Модули, предоставляют широкий диапазон операций над строками.

String	Наиболее крупные операции по строкам.
Re	Операции с регулярными выражениями.
StringIO	Работа со строками как с объектами файла.
cStringI	Более быстрая версия модуля StringIO.
Codecs	Регистрация кодеров и работа с ними.

Средства интернационализации

Средства, позвольте писать программы, рассмотрев национальные установки пользователя.

Locale	Использование национальных особенностей
gettext	Выпуск сообщений на родном языке .

Математический аппарат

Модули, позвольте проводить основные математические операции. Если Вы интересуетесь многомерными массивами и функциями линейной алгебры, смотрите на пакет модулей «Числовой Python» (NumPy).

Math	Математические функции для работы с вещественными числами.
Cmath	Математические функции для работы с комплексными числами.
Random	Поколение псевдослучайных чисел с различными дистрибутивами.
Whrandom	Генератор псевдослучайного числа.
Bisect	Обслуживание последовательностей в отсортированном состоянии. Реализует алгоритм поиска путем сокращения вдвое.
Array	Эффективные массивы чисел идентичного типа.

Math — математические функции для работы с вещественными числами

Математический модуль всегда доступен и обеспечивает доступ к фиксированным математическим функциям. Эти функции не работают с комплексными числами — для этого Цель, необходимо использовать функции того же имени от cmath модуля.

`Acos (x)` Возвраты arccosine x.
`Asin (x)` Возвраты arcsine x.
`Atan (x)` Возвраты арктангенс x.
`atan2 (x, s)` эквивалентный atan (x/y). Аргумент в может быть равен 0 — в этом пи/2 возвратов случая.
`Ceil(x)` Возвращает самое маленькое вещественное число с нулевой больше дробной частью, чем x.
`Cos (x)` Возвращает cosin x.
`Cosh (x)` Возвращает гиперболический cosin x.
`Exp (x)` Возвращает e ** x.
`Fabs (x)` Возвращает абсолютное x.
`Floor (x)` Возвращает самое большое вещественное число с нулевой меньшей дробной частью, чем x.
`Fmod (x, s)` результат этой функции зависит от реализации функции того же названия библиотеки языка C. Обычно приводит к тому же результату как x % at.
`Frexp (x)` Возвраты связываются‘ (m, e)’ где m — мантисса (вещественное число) и e — экспоненциальная часть (целое число). Для чисел t и e всегда удовлетворяется условие ‘x == m * 2 ** e. Если аргумент x равен нулю, возвраты (0.0, 0)’. Иначе это всегда выполняется ‘0.5 <= by abs (t) <1’.
`Hypot (x, s)` возвращает Евклидово кодовое расстояние, ‘к sqrt (x*x + to an u*)
`ldexp (m, e)` функция, возвратиться frexp (|).возвращает 'm * (2 ** e)’.
`Log(x)` возвращает натуральный логарифм x.
`Log10 (x)` возвращает десятичный логарифм x.
`Modf (x)` Возвращает поезд из | пары вещественных чисел — дробная и целая часть x. Оба возвращенных числа имеют тот же знак, как в числе x.
`Pow(x, s)` Возвращает x **s.

```
Sinh (x) Возвраты гиперболический синус x.
Sqrt (x) Возвраты квадратный корень от x.
Xarap (x) Возвраты касательная x.
Tanh (x) Возвраты гиперболическая касательная x.
```

Обратите внимание, что frexpO и функции modf () имеют другой интерфейс, чем их эквиваленты на языке С: они возвращают несколько значений вместо того, чтобы возвратить второе значение через аргумент.

Модуль также определяет две константы:

```
Pi - число π.
E - число e.
```

cmath — mathematical functions for work with complex numbers

The cmath module is always available and provides access to mathematical functions, working with complex numbers.

```
Acos (x) Возврат arccosin x.
acosh (x) Возврат hyperbolic arccosin x.
asin (x) Возврат arcsin x.
Asinh (x) Возврат гиперболический arcsin x.
Atan (x) Возврат arctangent x.
atanh(x) Возврат гиперболический arctangent x.
Cos (x) Возврат cosin x.
Cosh (x) Возврат гиперболический cosin x.
Exp (x) Возврат e ** x.
Log (x) Возврат натуральный логарифм x.
log10 (x) Returns десятичный логарифм x.
Sin (x) Возврат sin x.
Sinh (x) Возврат гиперболический sin x.
Sqrt (x) Возврат из квадрата x.
Tan (x) Возврат tangent x.
tanh (x) Возврат гиперболический tangent x.
```

Обратите внимание, что frexpO и функции modf () имеют другой интерфейс, чем их эквиваленты на языке С: они возвращают несколько значений вместо того, чтобы возвратить второе значение через аргумент.

Модуль также определяет две константы:

`Pi` - число (вещественный) π.
`E` - число (вещественный) `e`.

Давайте заметим, что эти функции подобны, но не идентичны функциям того же имени в математическом модуле. Даже когда различия показывают с помощью материальных аргументов. Для Пример математика, sqrt (-1) генерирует исключение в то время как cmath. Sqrt (-1) возвраты lj. Кроме того, функции cmath модуля всегда возвращают комплексное число, даже если он может быть выражен в материале (в этом случае, число имеет нулевую мнимую часть).

Текущий каталог:

```
In [17]: os.getcwd()
Out[17]: '/Users/cburns/src/scipy2009/scipy_2009_tutorial/source'
```

List a directory:

```
In [31]: os.listdir(os.curdir)
Out[31]:
['.index.rst.swo',
 '.python_language.rst.swp',
 '.view_array.py.swp',
 '_static',
 '_templates',
 'basic_types.rst',
 'conf.py',
 'control_flow.rst',
 'debugging.rst',
 ...
```

Сделайте каталог:

```
In [32]: os.mkdir('junkdir')

In [33]: 'junkdir' in os.listdir(os.curdir)
Out[33]: True
```

Переименовать каталог:

```
In [36]: os.rename('junkdir', 'foodir')

In [37]: 'junkdir' in os.listdir(os.curdir)
Out[37]: False

In [38]: 'foodir' in os.listdir(os.curdir)
Out[38]: True

In [41]: os.rmdir('foodir')

In [42]: 'foodir' in os.listdir(os.curdir)
Out[42]: False
```

Delete a file:

```
In [44]: fp = open('junk.txt', 'w')

In [45]: fp.close()

In [46]: 'junk.txt' in os.listdir(os.curdir)
Out[46]: True

In [47]: os.remove('junk.txt')

In [48]: 'junk.txt' in os.listdir(os.curdir)
Out[48]: False
```

os.path: path manipulations

os.path provides common operations on pathnames.

```
In [70]: fp = open('junk.txt', 'w')

In [71]: fp.close()

In [72]: a = os.path.abspath('junk.txt')

In [73]: a
Out[73]:
'/Users/cburns/src/scipy2009/scipy_2009_tutorial/source/junk.txt'
```

```
In [74]: os.path.split(a)
Out[74]: ('/Users/cburns/src/scipy2009/scipy_2009_tutorial/source',
          'junk.txt')

In [78]: os.path.dirname(a)
Out[78]: '/Users/cburns/src/scipy2009/scipy_2009_tutorial/source'

In [79]: os.path.basename(a)
Out[79]: 'junk.txt'

In [80]: os.path.splitext(os.path.basename(a))
Out[80]: ('junk', '.txt')

In [84]: os.path.exists('junk.txt')
Out[84]: True

In [86]: os.path.isfile('junk.txt')
Out[86]: True

In [87]: os.path.isdir('junk.txt')
Out[87]: False

In [88]: os.path.expanduser('~/local')
Out[88]: '/Users/cburns/local'

In [92]: os.path.join(os.path.expanduser('~'), 'local', 'bin')
Out[92]: '/Users/cburns/local/bin'
```

Running an external command

```
In [8]: os.system('ls')
basic_types.rst  demo.py        functions.rst  python_language.rst
standard_library.rst
control_flow.rst exceptions.rst  io.rst        python-logo.png
demo2.py         first_steps.rst oop.rst       reusing_code.rst
In [20]: import sh
In [20]: com = sh.ls()
In [21]: print com
```

```
basic_types.rst   exceptions.rst   oop.rst   standard_library.rst
control_flow.rst  first_steps.rst  python_language.rst
demo2.py          functions.rst    python-logo.png
demo.py           io.rst           reusing_code.rst

In [22]: print com.exit_code
0
In [23]: type(com)
Out[23]: sh.RunningCommand
```

Обход каталога

os.path.walk generates a list of filenames in a directory tree.

```
In [10]: for dirpath, dirnames, filenames in os.walk(os.curdir):
  ....:     for fp in filenames:
  ....:         print os.path.abspath(fp)
  ....:
  ....:
/Users/cburns/src/scipy2009/scipy_2009_tutorial/source/.index.rst.swo
/Users/cburns/src/scipy2009/scipy_2009_tutorial/source/.view_array.py.s
wp
/Users/cburns/src/scipy2009/scipy_2009_tutorial/source/basic_types.rst
/Users/cburns/src/scipy2009/scipy_2009_tutorial/source/conf.py
/Users/cburns/src/scipy2009/scipy_2009_tutorial/source/control_flow.rst
...
```

Переменные среды:

```
In [9]: import os

In [11]: os.environ.keys()
Out[11]:
['_',
 'FSLDIR',
 'TERM_PROGRAM_VERSION',
 'FSLREMOTECALL',
 'USER',
```

'HOME',
'PATH',
'PS1',
'SHELL',
'EDITOR',
'WORKON_HOME',
'PYTHONPATH',
...

```
In [12]: os.environ['PYTHONPATH']
Out[12]: '.:/Users/cburns/src/utils:/Users/cburns/src/nitools:
/Users/cburns/local/lib/python2.5/site-packages/:
/usr/local/lib/python2.5/site-packages/:
/Library/Frameworks/Python.framework/Versions/2.5/lib/python2.5'

In [16]: os.getenv('PYTHONPATH')
Out[16]: '.:/Users/cburns/src/utils:/Users/cburns/src/nitools:
/Users/cburns/local/lib/python2.5/site-packages/:
/usr/local/lib/python2.5/site-packages/:
/Library/Frameworks/Python.framework/Versions/2.5/lib/python2.5'
```

sys module: system-specific information

Определенная для системы информация связана с интерпретатором Python.

• Какую версию Python Вы выполняете и где это установлено:

```
In [117]: sys.platform
Out[117]: 'darwin'

In [118]: sys.version
Out[118]: '2.5.2 (r252:60911, Feb 22 2008, 07:57:53) \n
        [GCC 4.0.1 (Apple Computer, Inc. build 5363)]'

In [119]: sys.prefix
Out[119]: '/Library/Frameworks/Python.framework/Versions/2.5'
```

• List of command line arguments passed to a Python script:

- In [100]: sys.argv
- Out[100]: ['/Users/cburns/local/bin/ipython']

sys.path is a list of strings that specifies the search path for modules. Initialized from PYTHONPATH:

```
In [121]: sys.path
Out[121]:
['',
 '/Users/cburns/local/bin',
 '/Users/cburns/local/lib/python2.5/site-packages/grin-1.1-py2.5.egg',
 '/Users/cburns/local/lib/python2.5/site-packages/argparse-0.8.0-
py2.5.egg',
 '/Users/cburns/local/lib/python2.5/site-packages/urwid-0.9.7.1-
py2.5.egg',
 '/Users/cburns/local/lib/python2.5/site-packages/yolk-0.4.1-py2.5.egg',
 '/Users/cburns/local/lib/python2.5/site-packages/virtualenv-1.2-
py2.5.egg',
 ...
```

Обработка исключений в Python

Исключения подняты различными видами ошибочного возникновения, выполняя код Пайтона. В твоем собственном кодексе ты можешь также зафиксировать ошибки или определить таможенные ошибочные типы. Ты можешь хотеть посмотреть на описания встроенные Исключения, ища правильный тип исключения.

1. Исключения

Синтаксические ошибки, возможно, чаще всего встречаются во время изучения языка:

```
In [1]: 1/0

ZeroDivisionError: integer division or modulo by zero

In [2]: 1 + 'e'
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
In [3]: d = {1:1, 2:2}

In [4]: d[3]

KeyError: 3

In [5]: l = [1, 2, 3]

In [6]: l[4]

IndexError: list index out of range

In [7]: l.foobar

AttributeError: 'list' object has no attribute 'foobar'
```

Как Вы видите, существуют различные типы исключений для различных ошибок.

Синтаксический анализатор отображает строку, содержащую ошибку, и определяет место, где ошибка была найдена. Ошибка обычно вызывается лексемой, предшествующей стрелке: в данном Пример ошибка найдена на сайте ключевое слово печати как прежде чем это, там не двоеточие (*: '). Имя файла и номер строки отображены, чтобы Вы знали, где искать ошибку, если инструкции прочитаны из файла.

Исключения повышены ошибками в Python:

Даже если инструкция или выражение, это синтаксически правильно, ошибка, могут произойти в попытке выполнения. Ошибки, найденные во времени выполнения, не являются, конечно, фатальными, и Вы скоро узнаете, поскольку они могут быть обработаны в программах в языке Python. Большинство исключений, однако, не обрабатывается программами и выводом к сообщениям об ошибках:

2. Ловля исключений

try/except

```
In [10]: while True:
  ....:     try:
  ....:         x = int(raw_input('Please enter a number: '))
  ....:         break
  ....:     except ValueError:
  ....:         print('That was no valid number.  Try again...')
  ....:
Please enter a number: a
That was no valid number.  Try again...
Please enter a number: 1

In [9]: x
Out[9]: 1
```

try/finally

```
In [10]: try:
  ....:     x = int(raw_input('Please enter a number: '))
  ....: finally:
  ....:     print('Thank you for your input')
  ....:
  ....:
Please enter a number: a
Thank you for your input
```

ValueError: invalid literal for int() with base 10: 'a'

Важный для управления ресурсами (e.g. закрытие файла)
Easier to ask for forgiveness than for permission

```
In [11]: def print_sorted(collection):
  ....:     try:
  ....:         collection.sort()
  ....:     except AttributeError:
  ....:         pass
  ....:     print(collection)
```

```
....:
....:
In [12]: print_sorted([1, 3, 2])
[1, 2, 3]

In [13]: print_sorted(set((1, 3, 2)))
set([1, 2, 3])

In [14]: print_sorted('132')
132
```

Прямо после того, как детали типа появления исключения отображены. Предыдущая часть показа сообщения об ошибке | контекст появления эксклюзивной ситуации в форме содержания стека. В то же время строки исходного текста, за исключением строк, считанных из стандартного входа, отображены.

3. Повышение исключений

Можно записать программы, которые обработают определенные исключения. Посмотрите на следующий Пример, в котором пользователю будет выпущено приглашение, пока целый номер не введен, или выполнение не прервано (обычно Ctrl-C). Во втором случае сгенерировано исключение KeyboardInterrupt.

• • Получение и переповышение исключения:

```
In [15]: def filter_name(name):
   ....:     try:
   ....:         name = name.encode('ascii')
   ....:     except UnicodeError as e:
   ....:         if name == 'Gaël':
   ....:             print('OK, Gaël')
   ....:         else:
   ....:             raise e
   ....:     return name
   ....:

In [16]: filter_name('Gaël')
OK, Gaël
```

- Out[16]: 'Ga\xc3\xabl'
- In [17]: filter_name('Stéfan')
- ---
- UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in position 2: ordinal not in range(128)

Используйте исключения, чтобы уведомить, что определенные условия соблюдены (например, StopIteration) или не (например, пользовательская ошибка при повышении)

Инструкция TRY работает следующим образом

При первом ответвлении попытки (инструкция, которые являются между ключевым словом, *TRY* кроме) выполняется.

• Если нет никакой эксклюзивной ситуации, ответвление кроме передается, и выполнение инструкции *TRY* происходит до конца.

• Если во времени выполнения ответвление *TRY* исключение сгенерировано, остальная часть ответвления передается. Далее, если тип (класс) исключения соответствует кроме указанного после ключевого слова, выполняется ве (гвь кроме, и выполнение инструкции *TRY* происходит до конца.

• Если исключение не соответствует кроме указанного после ключевого слова, то оно передается внешнему блоку *TRY* или если процессор не найден, исключение считают не прерванным, выполнение прервано, и сообщение об ошибке отображено.

Инструкция *TRY* может иметь больше чем одно ответвление кроме, определяя процессоры для различных исключений. Только один из них будет выполнен (как самое большее). Только исключения, сгенерированные в соответствующем ответвлении *TRY*, но не в других процессорах инструкции *TRY*, обрабатываются. После того, как ключевое слово кроме нескольких типов исключений в форме поезда может быть определено:

```
... except (RuntimeError, TypeError, NameError):
....pass
```

В последнем ответвлении кроме типа исключения может быть понижен — все исключения будут обработаны в этом случае. Используйте такую запись с дополнительной осторожностью —

таким образом, можно легко замаскировать реальные ошибки! Перехват всех исключений может использоваться для вывода сообщения об ошибке, и затем неоднократно генерировать его (позволяющий обрабатывать исключение в другом месте):

```
import sys try:
f = open('myflie.txt')
s = f.readline()
i = int(s.strip())
except IOErro
r, exc:
print "Ошибка ввода/зывода", exc e
xcept ValueError:
print " He I can transform data to the whole
type,
except:
print " Unexpected error:", sys.exc_info()[0]
raise # repeatedly generates the last
        # the intercepted exception
```

После всех ответвлений кроме инструкция *TRY* еще может содержать ответвление, которое будет выполнено, если во времени выполнения ответвление *TRY* исключение не будет сгенерировано. Пример:

```
for arg in sys.argv[1:]:
try:
f = open(arg, 'r')
except IOError:
print 'I can open He', arg
else:
print arg, 'contains', len(f.readlines()),
\ 'lines'
f.close()
```

Обычно еще для использования ответвления лучше, чем добавить код к основному ответвлению инструкции *TRY*, поскольку это позволяет избегать обработки исключений, сгенерированной кодом, который Вы не собирались защищать вообще.

Исключение может иметь значение связанное с ним — аргумент, переданный классу исключений при инициализации. Существование и тип аргумента зависят от типа исключительной ситуации. Присваиваемое значение используется при получении для исключения значения строки. Для получения значения исключения, в кроме ответвления после класса исключений (или поезд классов) определяют переменную:

```
>>> try:
spam ()
... except NameError, x:
...       print 'Name', x, 'is not defined'
Name spam it is not defined
```

Если исключение не обрабатывается, значение исключения отображено в сообщении об ошибке после имени класса исключений.

Процессор прерывает не только исключения, сгенерированные непосредственно в блоке *TRY*, но также и в функциях, вызванных от него. Пример:

```
>>> def this_fails() :
 x = 1/0
 .....
>>> try:
this_fails()
... except ZeroDivisionError, exc:
...       print 'Runtime error:', exc
Runtime error: integer division or modulo
```

Поколение исключений

Инструкция повышения позволяет программисту генерировать исключение обязательно. Пример:

```
>>> raise NameError ('HiThere')
Traceback (innermost last):
File "<stdin>", line 1 NameError: HiThere
```

Исключения определяются пользователем

Можно использовать собственные исключения, с помощью выражений строки для обозначения его имени (устаревший путь) или создавая новые классы исключения. Пример:

```
>>> class MyError (Exception) : pass

>>> try:
raise MyError(2*2)
... except MyError, e:
...  print 'Исключение MyError, value равно', e

Исключение MyError, value равно 4
>>> raise MyError (1)
Traceback (innermost last):
File "<stdin>", line 1
__main__.MyError: 1
```

Еще одна опция записи инструкции *TRY* — с определением ответвления «безопасности» наконец, которое будет выполнено при любых обстоятельствах. Пример:

```
>>> try:
raise KeyboardInterrupt()
... finally:
print 'Good-bye!'
...
Good-bye
Traceback (innermost last):
File "<stdin>", line 2
KeyboardInterrupt
```

Ответвление наконец выполняется независимо от того, была ли эксклюзивная ситуация во времени выполнения блок TRY или не включая то, если вывод происходит согласно инструкции повреждения или возврата.

Инструкция TRY может иметь или одно или несколько ответвлений кроме или одно ответвление наконец, но не обе опции сразу.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Практическая работа № 1
Переменные, значения и их типы. Присваивание значения. Ввод значений с клавиатуры. Арифметические операции. Возможности интерпретатора

Цель: Рассмотреть основные типы

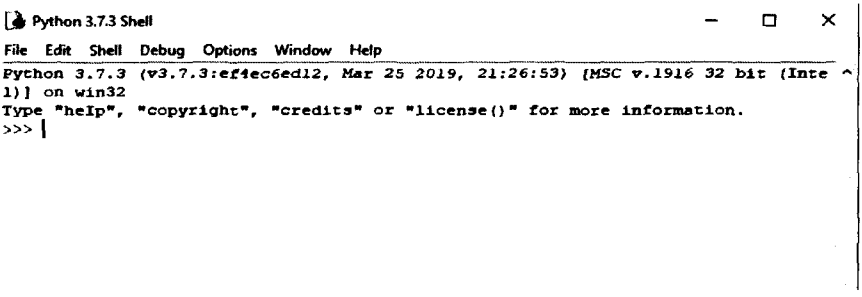
План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.

Скачивать Python необходимо с официального сайта https://python.org/downloads/windows/, выбираем **Python 3**.

В установщик Python для Windows встроена среда разработки IDLE.

После загрузки и установки python открываем IDLE (среда разработки на языке Python, поставляемая вместе с дистрибутивом).



Ввод данных. Функция input()

Функция input() передает введенные данные в программу. Их можно присвоить переменной. В этом случае интерпретатор не выводит строку сразу же.

Функция input() всегда возвращает строку. Т.е. если даже вводится число 5, оно присваивается переменной как строка "5".

Вывод данных

Функция print() - вывод данных, по умолчанию, на экран.

В функцию print() можно передавать как непосредственно литералы, так и переменные, вместо которых будут выведены их

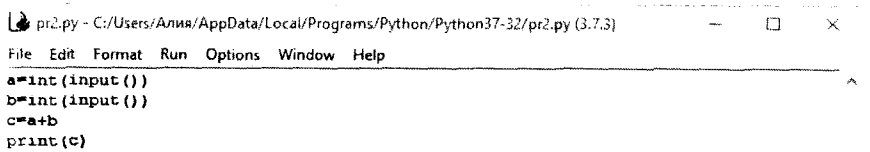
значения. Аргументы функции, разделяются между собой запятыми. В выводе вместо запятых значения разделены пробелом.

Если в скобках стоит выражение, то сначала оно выполняется, после чего print() уже выводит результат данного выражения.

Для работы с текстами программ необходимо создать новое окно: в интерактивном режиме IDLE выберите **File → New File (или нажмите Ctrl + N).**

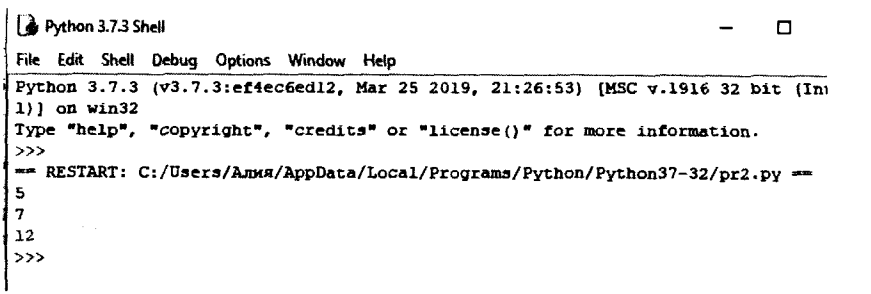
Пример: найти сумму двух чисел

В открывшемся окне введите следующий код:



Выполнение программы - **F5 (или в меню IDLE Run → Run Module).** Перед запуском IDLE предложит сохранить файл, после чего программа запустится.

Результат:



Оператор присваивания

Пример 1: Вещественные числа a и b. Вычислите функцию $c = a^b + \sqrt{a}$

```
a=input();a=int(a);
b=input();b=int(b);
import math
c=math.pow(a,b)+math.sqrt(a);
printI;
```

Пример 2: Запишите программу (необходимые данные вводятся с клавиатуры) для вычисления всех трех сторон прямоугольного треугольника, если задан, один из острых углов и площадь.

Решение: Стороны прямоугольного треугольника обозначим – *a* и *b* и гипотенузу *c*. Площадь треугольника S, один из острых углов α. Используем формулу площади прямоугольного треугольника $S=\frac{ab}{2}$ и формула тангенса $tg=\frac{a}{c}$. Отсюда возможно получить выражение для одной из сторон:

$$a = a = \sqrt{\frac{2S}{tg\alpha}}.$$

Теперь легко вычислить другую сторону, и гипотенузу:

$$b=\frac{2S}{a},\ c=\sqrt{a^2+b^2}.$$

```
From math import *
S = int (input ('Площадь трегольника = '))
alpha = int (input ('Острый угол (в градусах)
a = sqrt(2*S/tan(radians(alpha)))
b = 2*S/a
c = (a**2 + b**2|) ** (1/2)
print (a, b, c)
```

Вывод программы:

```
Площадь трегольника = 8
Острый угол (в градусах) = 45
4.0 4.0 5.656854249492381
```

Задания для студентов:

Задание. Выполнять три задания в зависимости от номера в списке.

Чтобы узнать номера ваших заданий, необходимо решить задачку: требуется сделать задания № m, № m + 5, № m + 10, где m

(n − 1)%5 + 1, n _ порядковый номер студента в списке группы по алфавиту.

Используя арифметические операторы (+, −, *, /, //, %), напишите программу (необходимая информация запрашивается у пользователя с клавиатуры).

1. Составьте арифметическое выражение и вычислите n-е чётное число (первым считается 2, вторым 4 и т.д.).

2. Составьте арифметическое выражение и вычислите n-е нечётное число (первое _ 1, второе _ 3 и т.д.).

3. Сколько человек находится между i-м и k-м в очереди?

4. Сколько нечётных чисел на отрезке [a; b], если a и b _ чётные? A и b _нечётные? A _ чётное, a _ нечётное?

5. Сколько полных часов, минут и секунд содержится в x секундах? Разложите имеющееся количество секунд на сумму из x часов + y минут + z секунд.

6. В доме 9 этажей, на каждом этаже одного подъезда по 4 квартиры. В каком подъезде, и на каком этаже находится n-я квартира?

7. Старинными русскими денежными единицами являются: 1 рубль = 100 копеек, 1 гривна = 10 копеек, 1 алтын = 3 копейки, 1 полушка = 0,25 копейки. Имеется A копеек. Разложите имеющуюся сумму в копейках на сумму из x рублей + y гривен + z алтынов + v полушек.

8. Стрелка прибора вращается с постоянной скоростью, совершая w оборотов в секунду (не обязательно стрелка прибора, может быть это волчок в игре .Что? Где? Когда?. И т.п.) Угол поворота стрелки в нулевой момент времени примем за 0. Каков будет угол поворота через t секунд?

9. Вы стоите на краю дороги и от вас до ближайшего фонарного столба x метров. Расстояние между столбами y метров. На каком расстоянии от вас находится n-й столб?

10. Та же ситуация, что и в предыдущей задаче. Длина вашего шага z метров. Мимо скольких столбов вы пройдете, сделав n шагов?

11. x _ вещественное число. Запишите выражение, позволяющее выделить его дробную часть.

12. x _ вещественное число. Запишите выражение, которое округлит его до сотых долей (останется только два знака после запятой).

13. От бревна длиной L отпиливают куски длиной x. Сколько целых полноразмерных кусков максимально удастся отпилить?

14. Бревно длиной L распилили в n местах. Какова средняя длина получившихся кусков?

15. Резиновое кольцо диаметром d разрезали в n местах. Какова средняя длина получившихся кусков?

Задачи для самостоятельного решения (домашняя работа).

Последовательные действия описаны последовательными строками программы. Стоит добавить, однако, что добавление отступа важно в программах, таким образом, все операторы, включенные в последовательность действий, должны иметь то же добавление отступа:

```
a = 1
b = 2
a = a + b
b = a – b
a = a – b
print (a, b)
```

Что делает этот Пример? Можно протестировать предположение с помощью интерактивного режима интерпретатора Python.

При работе с Python в интерактивном режиме, представлена одна большая программа, состоящая из последовательных действий. В Пример выше, используются операторы присваивания и оператор печати.

2. Вычислите на компьютере значения переменных, определенных в таблице. 1 на съемочной площадке оцененная формула и основные данные. Значения печати вводимых данных и результаты вычислений, сопровождая вывод с названиями отображенных переменных.

Таблице 1.

№	Формулы	Значения
1	$a=\frac{2\cos(x-\pi/6)}{1/2+\sin^2 y}\ b=1+\frac{z^2}{3+z^2/5}$	X=1,426 y=1,220 г Z=3,5
2	$\gamma=\lvert x^2-\sqrt{y/x}\rvert$ $\varphi=(y-x)\frac{y-z/(y-x)}{1+(y-x)^2}$	x= 1,825 y =18,225 z = -3,298
3	$s=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}\ \phi=x(\sin x^3+\cos^2 y)$	x=0,335 y =0,025
4	$y=e^{-bt}\sin(at+b)-\sqrt{\lvert bt+a\rvert}$ $s=b\sin(at^2\cos 2t)-1$	a = -0,5 b=1,7 t=0,44
5	$y=\cos^2 x^3-x/\sqrt{a^2+b^2}$ $w=\sqrt{x^2+b}-b^2\sin^2(x+a)/x$	a=1,5 b=15,5 x= -2.9
6	$s=x^3tg^2(x+b)^2+a/\sqrt{x+b}$ $Q=\frac{bx^2-a}{e^{ax}-1}$	a=16,5 b =3,4 x=0,61
7	$R=x^2(x+1)/b-\sin^2(x+a)$ $s=\sqrt{xb/a}+\cos^2(x+b)^3$	a=0,7 b=0,05 x =0,5
8	$y=\sin^3(x^2+a)^2-\sqrt{x/b}$ $z=\frac{x^2}{a}+\cos(x+b)^3$	a=1,1 b=0,004 x=0,2
9	$f=\sqrt{m*tgt+\lvert c\sin t\rvert}$ $z=m*\cos(bt\sin t)+c$	m=2 c= -1 t=1,2 b=0,7
10	$y=b*tg^2x-\frac{a}{\sin^2(x/a)}$ $d=a*e^{-\sqrt{a}}\cos(b*x/a)$	a=3,2 b=17,5 x= -4,8
11	$f=\ln(a+x^2)+\sin^2(x/b)$ $z=e^{-c*x}\frac{x+\sqrt{x+a}}{x-\sqrt{\lvert x-b\rvert}}$	a=10,2 b=9,2 x=2,2 c=0,5
12	$y=\frac{a^2+b*\cos(a+b)*x}{x+1}\quad R=\sqrt{x^2+b}-b^2*\sin^2(x+a)/x$	a=0,3 b=0,9 x=0,61

№	Формулы	Значения
13	$z = \sqrt{a * x * \sin 2x} + e^{-2x}(x+b)$ $w = \cos^2 x - x/\sqrt{a^2+b^2}$	a=0,5 b=3,1 x=1,4
14	$e = \frac{a^2 x + e^{-x}\cos bx}{bx - e^{-x}\sin bx + 1}$ $f = e^{2x}\ln(a+x) - b^2 \ln(b-x)$	a=0,5 b=2,9 x=0,3
15	$z = \frac{\sin x}{\sqrt{1+m^2 \sin^2 x}} - c * m \ln mx$ $y = e^{-ax}\sqrt{x+1} + e^{-bx}\sqrt{x+1,5}$	*m=0,7* c=2,1 x=1,7 a =0,5 b=1,08

Контрольные вопросы:

1. Объясните назначение input() и print().
2. На что влияет тип данных?
3. Какие типы данных Вы знаете?
4. Перечислите арифметические операции в языке Python.
5. В каком режиме первоначально запущена программа?
6. Для чего используется функция печати?
7. Какой горячей клавишей мы можем запустить программу?

Литература:

1. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 2
Вывод. Форматный вывод. Варианты вывода текста на экран.

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.

g	Общий числовой формат. Для заданной точности p >= 1 это число округляется до p значащих цифр, а затем форматирует результат как в формате с фиксированной точкой или в научной нотации в зависимости от величины.
G	То же самое, что и «g», но буквенные символы выводятся прописными
n	То же самое, что и «d», за исключением того, что используется текущая настройка локали для вставки соответствующих символов разделителей.
o	Число в восьмеричной системе счисления
s	Строковый тип (по умолчанию)
x	Шестнадцатеричное число
X	То же самое, что и «x», только буквенные символы выводятся прописными
%	Значение выводится в процентах (умножается на 100 и добавляется символ «%»)
None	Аналогичен «g», за исключением того, что нотация с фиксированной точкой, используется, тогда, когда число имеет по крайней мере одну цифру после десятичной точки. Точность по умолчанию настолько высока, насколько это необходимо для представления конкретного значения.

Флаги, используемые для выравнивания:

< - Выравнивание объекта по левому краю (по умолчанию)

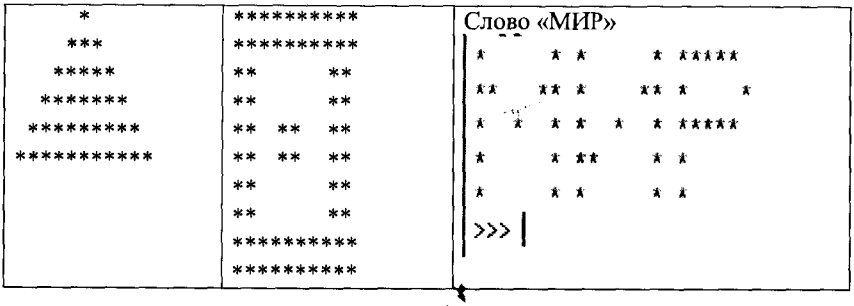
> - Выравнивание объекта по правому краю

^ - Выравнивание по центру

3. Решение задач.

Разобраться со всеми параметрами метода format().

1. С помощью символа "*" вывести следующий рисунок. Символы пробела и табуляции не использовать!



4. Подведение итогов.

Домашнее задание

Выведите следующие рисунки с помощью метода format() и функции print():

```
*
**
***
** **
** **
**  **
**  **
**   **
**    **
**     **
**      **
**********
**********

        *****
       *
      **
     ***
    ****
   *****
  ******
 *******
********
*********
**********
**********
**********
**********
 *********
```

Вопросы для самоконтроля:

1. Какая функция языка выводит информацию на экран?
2. Как вывести информацию с комментариями?
3. Какие эскейп-последовательности Вы знаете?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Практическая работа № 3.
Встроенные операции и функции.
Основные алгоритмические конструкции

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.

```
>>> x=complex(5,7)
>>> print(x)
(5+7j)
>>> y=complex(2,4)
>>> print(y)
(2+4j)
>>> z = x + y
>>> print(z)
(7+11j)
>>>
```

3. Решение задач.

Линейные алгоритмы. Составить программы к следующим задачам

1). Даны два действительных числа a и b. Получить их сумму, разность и произведение.

2). Даны действительные числа x и y. Получить $|x| - |y|+1$.

3). Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.

4). Даны два действительных положительных числа. Найти среднее арифметическое и среднее геометрическое этих чисел.

5). Даны два действительных числа. Найти среднее арифметическое этих чисел и среднее геометрическое их модулей.

4. Подведение итогов.

Вопросы для самоконтроля:

1. Перечислите логические операции языка.
2. Приведите пример решения линейного алгоритма.
3. Как ввести целое число?

Литература:

1. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 4.
Операторы условий. Условия равенства/неравенства

Цель: Рассмотреть код многократного использования: сценарии и модули

План практических занятий:

4.1 Сценарии

4.2. Импорт объектов от модулей

4.3. Создание модулей

• Используйте значимые имена объектов

• Добавление отступа: никакой выбор!

Заказывание обязательно в Пайтоне! Каждый блок команды после двоеточия имеет дополнительный уровень углубления относительно предыдущей линии с двоеточием. Нужно поэтому заказать после deff (): или while. В конце таких логических блоков каждый уменьшает глубину углубления (и повторно увеличивает ее, если новый блок введен, и т.д.).

Строгое уважение углубления – цена, чтобы заплатить за избавление от {или; знаки, которые очерчивают логические блоки на других языках. Неподходящее углубление приводит к ошибкам такой как

IndentationError: unexpected indent (test.py, line 2)

Весь этот порядок добавления отступа может немного сбивать с толку в начале. Однако с четким добавлением отступа, и в отсутствие дополнительных символов, то получается очень хороший читающий код, по сравнению с другими языками.

```
File Edit Format Run O
a=int(input())
b=int(input())
if a>b:
   print(a)
else:
   print(b)
```

```
File Edit Format Run Options Wi
a=int(input())
b=int(input())
c=2*a if a>0 else 2*b
print(c)
```

Задания для студентов:

Задача. (Условный оператор). Выполните три задачи в зависимости от числа в списке. Необходимо сделать задачи № m, № m + 5, № m + 10 где m = (n − 1) %5 + 1, n – число студента в списке группы в алфавитном порядке.

1. Запишите программу, которая запрашивает значение x, и затем отображает значение следующей функции от x (это называют в латинском .signum., который означает знак.):

$$y(x) = \begin{cases} 1, & x > 0, \\ 0, & x = 0, \\ -1, & x < 0 \end{cases}$$

2. Запишите программу, которая запрашивает значение x, и затем отображает значение следующей функции от x:

$$y(x) = \begin{cases} \sin^2(x), & x > 0, \\ 0, & x = 0, \\ 1 + 2\sin(x^2), & x < 0 \end{cases}$$

3. Запишите программу, которая запрашивает значение x, и затем отображает значение следующей функции от x:

$$y(x) = \begin{cases} \cos^2(x), & x > 0, \\ 0, & x = 0, \\ 1 - 2\sin(x^2), & x < 0 \end{cases}$$

4. Запросите два числа от пользователя. Далее:

• если первое является более, чем вторым, то вычислить их различие и распечатать данные;

• если второе является более, чем первым, то вычислить их сумму и распечатать;

• если оба числа равны, то распечатать это значение.

5. Запросите два целых числа m и n от пользователя. Если целое число m разделено полностью в целое число n, то отобразиться частный от подразделения, иначе для отображения сообщения.m полностью не разделено на n.

6. Запишите программу для решения квадратного уравнения ax2 + к основному обмену + c = 0. Значения коэффициентов a, b, c вводятся с клавиатуры. Вычисление квадратного корня может быть организовано или конструкцией в stepen0.5, или посредством функции sqrt от математического модуля. Контрольное число дискриминанта: если это не присутствует меньше, чем нуль, корни, если это равно нулю, так корень 1, если существуют больше, чем нулевые корни два. Для этого Цель возможно еще использовать конструкцию типа если elif.

7. Запишите программу, решив кубическое уравнение типа y3+px+q = 0 C справка формулы Cardano. Значения коэффициентов p и q вводятся с клавиатуры. Найдите корни уравнения. Помните, что Python может работать с комплексными числами, но математический модуль не может использоваться для их возведения в степень. Будьте внимательны с кубическим корнем: кубический корень от отрицательного числа превращается в комплексное число.

8. Запишите программу, которая запрашивает ее возраст (целое число лет) от пользователя и в зависимости от значения вводимых дисплеев числа:

- от 0 до 7. Вам в детском саду.;
- от 7 до 18. Вам в школе;
- от 18 до 25. Вам в профессиональном учебном заведении;
- от 25 до 60. Вам для работы;
- от 60 до 120. Выбор предоставляется Вам;
- меньше, чем 0^{th} и больше чем 120 в пять раз в row:. Ошибка! Это – программа для людей!

9. Запишите программу, которая поможет Вам оптимизировать перемещение на автомобиле. Позвольте программе запросить следующие данные от пользователя:

- Сколько километров Вы хотите передать автомобиль?
- Сколько литров топлива потрачено автомобилем для 100 километров?
- Сколько литров топлива в Вашем корпусе?

Далее в зависимости от вводимых значений программа должна дать вердикт: Вы передадите желательное расстояние или нет;

10. Пользователь вводит три вещественных номера: длины сторон треугольника. Программа должна сказать пользователю:

- является ли треугольник равносторонним;
- является ли треугольник равнобедренным;
- универсален ли треугольник;
- является ли треугольник прямоугольным;
- существует ли в целом такой треугольник (такой треугольник не может быть то, если длина, по крайней мере, является больше чем одной стороной, или это равно сумме длин двух других).

11. Вес боксера-любителя известен. Такой случается так, что боксер может быть отнесен в одну из трех весовых категорий:

- легких до 60 кг;
- первый второй полусредний вес составляет до 64 кг;
- второй полусредний вес составляет до 69 кг;

Определите, в какой категории будет действовать этот боксер.

12. На чемпионате мира команде для приза дают 3 точки, за потерю 0, за ничью 1. Количество очков, полученных командой для игры, известно. Определите словесный результат игры (приз, потеря или ничья).

13. Сделайте программу, которая в зависимости от порядкового номера дня недели (от 1 до 7) отображает ее имя (в понедельник, во вторник…, в воскресенье).

14. Сделайте программу, которая в зависимости от порядкового номера месяца (1, 2…, 12) отображает его имя (январь, февраль…, декабрь).

15. Сделайте программу, которая в зависимости от порядкового номера месяца (1, 2…, 12) отображает сезон, которому в этом месяце принадлежит.

Задачи для самостоятельного решения (домашняя работа).

1) $f(x)=\begin{cases}5x^2+6, \text{если } -2<x<5\\ x^3+7, \text{если } x\geq 5\\ -1, \text{если } x\leq -2\end{cases}$

2) $f(x)=\begin{cases}\sqrt{x^3+5}, \text{если } x\geq 0\\ 3x^4+9, \text{если } -3<x<0.\end{cases}$

3) $f(x)=\begin{cases}\sqrt[3]{x^2+6x}, \text{ если } -4\langle x\le 5\\ x^5+3.5, \text{ если } x\rangle 5\end{cases}$

4) $f(x)=\begin{cases}x\cos x, \text{ если } x\langle 122\\ 5x^2+1.7, \text{ если } x\ge 122\end{cases}$

5) $f(x)=\begin{cases}x^3\cos x, \text{ если } 0\langle x\langle 2\\ 3x^4+7, \text{ если } x\ge 2\\ \sqrt{5x^2+16}, \text{ если } x\le 0.\end{cases}$

6) $f(x)=\begin{cases}xtgx-\sin x, \text{ если } x\langle 1.5\\ x^3+\sin x+7, \text{ если } 1.5\le x\langle 2.5\\ 3x^3+5, \text{ если } x\ge 2.5\end{cases}$

7) $f(x)=\begin{cases}\sqrt{3x^2+4}, \text{ если } 0\langle x\langle 1\\ 5-\sin^2 x, \text{ если } x\ge 1\end{cases}$

8) $f(x)=\begin{cases}\sqrt{x^2+|x|}, \text{ если } -5<x<0\\ 5x^2+\cos x, \text{ если } 0\le x\le 2\end{cases}$

9) $f(x)=\begin{cases}x\cos^2 x+\sin x, \text{ если } 0\le x<1\\ \sqrt[3]{x^2+6\sin x}, \text{ если } 1\le x\le 2\\ 1.7x^3+7, \text{ если } x>2\end{cases}$

10) $f(x)=\begin{cases}\sin x+\sqrt{x^2+1.2}, \text{ если } x<0\\ tg^2 x+\cos x+35, \text{ если } 2<x\le 6\end{cases}$

Задача 3.

3. Вычислите значение функции, установите в таблице 2. (согласно варианту задачи). Выполните вывод значений вводимых основных данных и результата вычисления значения функции, сопровождая вывод с названиями переменных.

Таблица 2.

№	Функция	Условия	Значения	Интервал изменения аргумента
1	$y=\begin{cases}at^2\ln t\\ 1\\ e^{at}\cos bt\end{cases}$	$1\le t\le 2$ $t<1$ $t>2$	a= -0,5 b=2	t∈[0;3] h=0,15
2	$y=\begin{cases}\pi x^2-7/x^2\\ ax^2+7\sqrt{x}\\ \lg(x+7\sqrt{x})\end{cases}$	x<1,3 x=1,3 x>1,3	a=1,5	x∈[0,8;2] h=0,1

№	Функция	Условия	Значения	Интервал изменения аргумента
3	$w=\begin{cases} ax^2+bx+c \\ a/x+\sqrt{x^2+1} \\ (a+bx)/\sqrt{x^2+1} \end{cases}$	x<1,2 x=1,2 x>1,2	a = 2,8 b = -0,3 c= 4	x∈[1;2] h=0,05
4	$Q=\begin{cases} \pi x^2-7/x^2 \\ ax^2+7\sqrt{x} \\ \ln(x+7\sqrt{\lvert x+a\rvert}) \end{cases}$	x<1,4 x=1,4 x>1,4	A = 1,65	x∈[0,7;2] h=0,1
5	$y=\begin{cases} 1{,}5\cos^2 x \\ 1{,}8ax \\ (x-2)^2+6 \\ 3tgx \end{cases}$	x<1 x=1 1< x < 2x > 2	A = 2,3	x∈[0,2;2,8] h=0,2
6	$w=\begin{cases} x\sqrt{x-a} \\ x\sin ax \\ e^{-ax}\cos ax \end{cases}$	x> a x= a x< a	a = 2,5	x∈[1;5] h=0,5
7	$Q=\begin{cases} bx-\lg bx \\ 1 \\ bx+\lg bx \end{cases}$	Bx< 1 Bx= 1 Bx> 1	b =1,5	x∈[0,1;1] h=0,1
8	$y=\begin{cases} \sin xlx \\ \cos^2 x \end{cases}$	x> 3,5 x≤ 3,5	———	x∈[2;5] h=0,25
9	$f=\begin{cases} \lg(x+1) \\ \sin^2\sqrt{\lvert ax\rvert} \end{cases}$	x> 1 x≤ 1	a = 20,3	x∈[0,5;2] h= 0,1
10	$z=\begin{cases} (\ln^2 x+x^2)/\sqrt{x+t} \\ \sqrt{x+t}+1/x \\ \cos x+t\sin^2 x \end{cases}$	x> 0,5 x= 0,5 x< 0,5	t= 2,2	x∈[0,2;2] h= 0,2

№	Функция	Условия	Значения	Интервал изменения аргумента
11	$s=\begin{cases}\dfrac{a+b}{e^x+\cos x}\\ (a+b)/(x+1)\\ e^x+\sin x\end{cases}$	x < 2,8 2,8 ≤ x < 6 x ≥ 6	a =2,6 b = -0,39	x∈[0;7] h= 0,5
12	$y=\begin{cases}a\lg x+\sqrt{\lvert x\rvert}\\ 2a\cos x+3x^2\end{cases}$	x > 1 x ≤ 1	a =0,9	x∈[0,8;2] h= 0,1
13	$w=\begin{cases}\dfrac{a}{i}+bi^2+c\\ i\\ ai+bi^2\end{cases}$	I < 4 4 ≤ i ≤ 6 i >6	a= 2,1 b= 1,8 c = -20,5	I∈[0;12] h= 1
14	$z=\begin{cases}a\sin(\dfrac{i^2+1}{n})\\ \cos(i+\dfrac{1}{n})\end{cases}$	$\sin(\dfrac{i^2+1}{n})>0$ $\sin\dfrac{i^2+1}{n}<0$	A=0,3 n= 10	I∈[1;10] h= 1
15	$w=\begin{cases}\sqrt{at^2+b\sin t+1}\\ at+b\\ \sqrt{at^2+b\cos t+1}\end{cases}$	t > 0,1 t = 0,1 t < 0,1	a = 2,5 b = 0,4	t∈[-1;1] h= 0,2

Контрольные вопросы:

1. В каком случае используются операторы цикла?
2. Что пути состоят в том, чтобы организовать циклы в Python?
3. В этом случае действительно ли удобно использовать оператор цикла с параметром?
4. Синтаксис оператора If.
5. В чем особенность elif?
6. Что выполняется раньше and или or?

Литература:

1. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 5.
Циклы и счетчики. Использование команд break, continue

Цель: Рассмотреть ввод и вывод

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.

```
>>> s=0
>>> i=5
>>> while i<=10:
        s=s+i
        print(i)
        i=i+1
else:
        print(s)


5
6
7
8
9
10
45
```

```
>>> k=0
>>> for i in "hello world":
        if i=="l":
                k+=1
else:
        print(k)


3
>>>
```

```
>>> for i in "hello world":
        if i=="l":
                continue
        print(i*2)


hh
ee
oo

ww
oo
rr
dd
>>> for i in "hello world":
        if i=="l":
                continue
        print(i*2,end='')


hheeoo  wwoorrdd
>>>
```

```
>>> for i in "hello world":
        if i=="l":
                break
        print(i*2,end='')


hhee
>>>
```

Задания для студентов:

Учитывая положительное число a> 0. Найдите самое большое представление $\frac{1}{2^n}, n \geq a$

Задача. (Задачи для цикла с условием). Выполните три задачи в зависимости от числа в списке группы в алфавитном порядке. Необходимо сделать задачи N. M, № m+5, № m+10, m = (n-1) %5+1 где n – число в списке группы.

1. Запишите программу, которая подведет итог номеров, введенных с клавиатуры, пока они не будут положительны.

2. Запишите программу, которая подведет итог номеров, введенных с клавиатуры, пока они не будут отрицательны.

3. Запишите программу, которая подведет итог номеров, введенных с клавиатуры, пока они не будут равны нулю.

4. Запишите программу, которая подведет итог номеров, введенных с клавиатуры, пока они не будут ровны.

5. Номер n дан. Распечатайте те натуральные числа, какой квадрат не превышает n.

6. Номер n дан. Найдите первое натуральное число, какой квадрат является больше, чем n.

7. Номер n дан. Среди номеров 1, 1 + 1/2, 1 + 1/2 + 1/3… находят первые, большие числа n.

8. Число a ($1 \leq a \leq 1.5$) дано. Среди чисел 1 + 1/2, 1 + 1/3, 1 + 1/4… (мы заметим, что каждый после числа в последовательности является менее, чем предыдущим) находят первое, это – меньше, чем a.

9. Запишите программу, которая запрашивает от пользователя числа, пока каждый после числа не является более, чем предыдущим. В конце сообщает программа, сколько номеров было введено.

10. Запишите программу, которая запрашивает от пользователя числа, пока каждый после числа не является менее, чем предыдущим. В конце сообщает программа, сколько номеров было введено.

11. Запишите программу, которая запрашивает от пользователя числа до каждого после целого числа. В конце сообщает программа, сколько номеров было введено.

12. Запишите программу, которая запрашивает от пользователя числа до каждого после числа меньше чем 10. В конце сообщает программа, сколько номеров было введено.

13. Натуральное число, в котором все цифры отличаются, дано. Определите порядковый номер его максимальной цифры, включая числа: от конца числа; с начала числа.

14. Натуральное число, в котором все цифры отличаются, дано. Определите порядковый номер его минимальной цифры, включая числа: от конца числа; с начала числа.

15. Натуральное число дано. Определите, сколько раз в нем встречает максимальную цифру (для Пример, для номера 132233, ответ равен 3, для номера 46336 к два, для номера 12345 к одному).

Задачи для самостоятельного решения (домашняя работа).

Задачи для цикла со счетчиком. Выполните три задачи в зависимости от числа в списке группы в алфавитном порядке. Необходимо сделать задачи N. M, № m+5, № m+10, m = (n-1) %5+1 где n – число в списке группы.

1. Запишите программу, вычисляющую сумму всех четных чисел в диапазоне от 1 до 90 включительно.

2. Запишите программа, вычисляющая сумму всех четных чисел в диапазоне от до b включительно (вводятся с клавиатуры).

3. Запишите программу, вычисляющую сумму всех нечетных чисел в диапазоне от 1 до 90 включительно.

4. Запишите программа, вычисляющая сумму всех нечетных чисел в диапазоне от до b включительно (вводятся с клавиатуры).

5. Распечатайте таблицу умножения на 5, желательно распечатать во взгляде:

$$1 \times 5 = 5$$
$$2 \times 5 = 10$$
$$\ldots$$
$$9 \times 5 = 45$$

Вместо знака умножения × это возможно использовать строчный латинский letter.x.

6. Распечатайте таблицу умножения на 9, желательно распечатать во взгляде:

$$1 \times 9 = 9$$
$$2 \times 9 = 18$$
$$\ldots$$
$$9 \times 9 = 81$$

Вместо знака умножения × это возможно использовать строчный латинский letter.x.

7. Распечатайте таблицу умножения на целом числе n, n вводится с клавиатуры (2 6 n 6 9), желательно распечатать во взгляде:

$$1 \times n = \ldots$$
$$2 \times n = \ldots$$
$$\ldots$$
$$9 \times n = \ldots$$

Вместо знака умножения × это возможно использовать строчный латинский letter.x. Внимание! Не необходимо распечатать n символ, вместо этого необходимо распечатать вводимое значение.

8. Распечатайте таблицу стоимости 50, 100, 150…, 1000 из сыра (стоимость 1 кг сыра вводится с клавиатуры).

9. Распечатайте таблицу стоимости 100, 200, 300…, 2000 леденцов (стоимость 1 кг леденцов вводится с клавиатуры).

10. Найдите сумму всех целых чисел от 10 до 100;

11. Найдите сумму всех целых чисел от до 100 содержащих (значение вводимого с клавиатуры).

12. Найдите сумму всех целых чисел от 10 до b включительно (значение b вводится с клавиатуры).

13. Найдите сумму всех целых чисел от до b включительно (оценивает, и b вводятся с клавиатуры).

14. Найдите работу всех целых чисел от 10 до 100 включительно. Обратите внимание, что Python может работать с целыми числами неограниченного размера!

15. Найдите работу всех целых чисел от до b включительно (оценивает, и b вводятся с клавиатуры).

Учитывая допустимый x. Вычисляют приближенное значение суммы бесконечного ряда:

1) $x + \frac{x^2}{2} + \frac{x^3}{3} + \ldots \quad (|x|<1)$

2) $x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots \quad (|x|<1)$

3) $x+\frac{x^3}{3}+\frac{x^5}{5}+...(|x|<1)$

Контрольные вопросы:

1. В каком случае используются операторы цикла?
2. Что пути состоят в том, чтобы организовать циклы в Python?
3. В этом случае действительно ли удобно использовать оператор цикла с параметром?

4 Синтаксис оператора While.

5 Как досрочно можно выйти из цикла?

6 Приведите пример реализации циклического алгоритма.

Литература:

1. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 6.
Различные типы последовательностей и общие для них операторы. Строки.

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.
3. Решение задач.

Привести примеры работы со строками, содержащие следующие операции, функции, методы:

1) сложение двух строк
2) умножение строки на число
3) определение длины строки
4) доступ к 5 символу строки
5) срезы: первый символ, последний символ, первые пять символов, последние 7 символов строки
6) получить перевернутую строку
7) проверить принадлежность подстроки строке (с использованием in: « s1 in s2»)
8) поиск подстроки в строке (s.find(<x>[, <start>[, <end>]]))
9) поиск подстроки в строке (s.rfind(<x>[, <start>[, <end>]])
10) поиск подстроки в строке (s.rfind(<x>))
11) количество вхождений подстроки в строку (s.count(<x>[, <start>[, <end>]])
12) проверить, заканчивается ли строка заданной подстрокой (s.endswith(<x>[, <start>[, <end>]])
13) количество вхождений подстроки в строку (s.count(<x>[, <start>[, <end>]])
14) поиск в строке подстроки (s.index(<x>[, <start>[, <end>]]))
15) проверить, начинается ли строка с заданной подстроки (s.startswith(<x>[, <start>[, <end>]])
16) Изменение регистра строки
a) s.capitalize()
b) s.lower()
c) s.swapcase()
d) s.title()
e) s.upper()

17) Классификация строк

a) s.isalnum()

b) s.isalpha()

c) s.isdigit()

d) s.isidentifier()

e) iskeyword()

f) string.islower()

g) string.isprintable()

h) string.isspace()

i) string.istitle()

j) string.isupper()

18) Выравнивание строк, отступы

a) string.center(<width>[, <fill>])

b) string.expandtabs(tabsize=8)

c) string.ljust(<width>[, <fill>])

d) string.lstrip([<chars>])

e) string.replace(<old>, <new>[, <count>])

f) string.rjust(<width>[, <fill>])

g) string.rstrip([<chars>])

h) string.strip([<chars>])

19) вывод в формате (format)

a) 'Hello, {language} {version}'.format(language='Python', version=3.6)

b) 'Hello, {language} {version}'.format(version=3.6, language='Python')

20) вывод в формате (f)

```
>>> language = 'Python'
>>> version = 3.6
>>> f'Hello, {language} {version}'
```

4. Подведение итогов.

Задачи для самостоятельного решения (домашняя работа).

Вычислите десятичные числа Pi, использующего формулу Wallis:

$$\pi = 2\prod_{i=1}^{\infty}\frac{4i^2}{4i^2-1}$$

Давайте решим популярную проблему нахождения всех простых чисел к некоторому целому числу n. Классический алгоритм решения этой задачи носит имя – Решето Эратосфена. Для нахождения всех простых чисел больше нет набора номер n, в соответствии с методом Эратосфена, необходимо выполнить эти шаги:

1. Выпишите подряд все целые числа от два до n (2, 3, 4..., n).

2. Позвольте переменной p быть первоначально равной два – первое простое число.

3. Вычеркните в списке числа от 2 пунктов до n, рассмотрев шаги на p (это будут числа несколько p: 2 пункта, 3 пункта, 4 пункта...).

4. Найдите, что первое не вычеркнуло число в списке, больше, чем p, и адаптировать значению переменной p, это – число.

5. Повторите шаги 3 и 4, до сих пор это возможно.

6. Теперь все не вычеркнули числа в списке – все это простые числа от 2 до n.

Решение: На практике алгоритм может быть улучшен следующим образом. На шаге № 3 числа возможно вычеркнуть, начинаясь сразу с p2 числа, потому что все составные числа меньше это будет уже вычеркнуто к этому времени. И, соответственно, возможно остановить алгоритм, когда p2 становится больше, чем n.

```
From math import sqrt
n = int (input ("an output of prime numbers
to..."))
a = list(range(n)) # we create the list from n of
elements
# the Second element is unit which not
# consider a prime number. We kill her with zero:
a[1] = 0
# Search of all elements to the set number:
for p in range (2, int (sqrt (n))+1):
if a[p]! = 0: # if it is not equal to zero, then
** 2 # to double j = p: current element prime
number
while j <n:
= 0 # to replace a[j] with 0
```

```
+ to pass j = j with p # into a position to m it
is more
# Output of prime numbers to the screen:
b = []
for i in a:
if a[i]! = 0:
b. append (a[i])
print(b)
```

Ответ:

```
output of prime numbers to number... 70
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
43, 47, 53, 59, 61, 67]
```

Задания для студентов

value = dict[key]
except:
value = default_value

if dict.has_key(key):
value = dict[key]
else:
value = default_value

Выполните задачи все один за другим. Соедините любую переменную со строкой: «Мы будем, конечно, учиться программировать!» Получите от него следующие сокращения:

1. Отобразите третий символ этой строки;
2. Отобразите предпоследний символ этой строки;
3. Отобразите первые пять символов этой строки;
4. Отобразите всю строку, кроме последних двух символов;
5. Отобразите все символы с даже индексами (включая ту индексацию начинается (0);
6. Отобразите все символы с нечетными индексами, то есть, запускающийся с первого символ строки;
7. Выведите четыре символа из центра строки;

8. Символы дисплея с индексами, несколько мы тремся;

9. Отобразите все символы вверх тормашками;

10. Отобразите все символы приблизительно в одной строке вверх тормашками с тех пор последнее;

11. Удалите второе слово из строки;

12. Замените второе слово строкой «никогда не»;

13. Добавьте к концу строки «на Python»;

14. Произнесите последнее слово к первому в строке;

15. Длина дисплея этой строки.

Контрольные вопросы:

1. Какие последовательности Вы знаете?
2. В чем особенность строки?
3. Что такое срез? Приведите пример работы среза.

Литература:

1. Гвидо ва Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 7.
Кортежи

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.
3. Решение задач.

1. Из элементов исходного кортежа сформируйте другой кортеж, элементы которого по модулю больше некоторого значения.
2. В кортеже, состоящего из заранее заданных целых чисел найдите сумму элементов.
3. В кортеже целых чисел найдите максимальный и минимальный элементы, а также осуществите их обмен.
4. В кортеже целых чисел вычислите произведение отрицательных элементов, имеющих нечетные индексы.
5. В кортеже целых чисел вычислите среднее арифметическое значение квадратов положительных элементов.

4. Подведение итогов.

Задачи для самостоятельного решения (домашняя работа).

1. В кортеже целых чисел определите число инверсий. Инверсией называется пара элементов, в которой большее число расположено слева от меньшего. Например, для данного кортежа: 24 35 29 44 8 22 4 число инверсий 3
2. Разработайте программу, в которой определяются максимальный и минимальный элементы среди положительных нечетных элементов целочисленного кортежа A(10).
3. Разработайте программу, которая выводит на экран два кортежа A(10), содержащих диаметры и веса шин. Следует отобрать две шины, диаметры которых отличаются не более чем на a см, а вес - не более чем на b килограмм.

Контрольные вопросы:

1. Приведите пример кортежа.
2. Чем кортеж отличается от строки?
3. Перечислите методы для работы с кортежами.

Литература:

1. Гвидо ва Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 8.
Работа со списками.
Специальные операторы, функции для них

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.

```
>>> list ('gruppa')
['g', 'r', 'u', 'p', 'p', 'a']
>>>
```

```
>>> a=[] #пустой список
>>> b=['g',['rup'],'p','a']
>>> c=[1,2,3,4,5]
>>> a
[]
>>> b
['g', ['rup'], 'p', 'a']
>>> c
[1, 2, 3, 4, 5]
>>>
```

```
>>> c=[c * 3 for c in [1, 2, 3, 4, 5] if c != 3]
>>> c
[3, 6, 12, 15]
>>>
```

ГЕНЕРАТОРЫ СПИСКОВ

В Python создать список можно при помощи генераторов:
Первый простой способ.
Сложение одинаковых списков заменяется умножением:

```
# список из 10 элементов, заполненный единицами
L= [1]*10
# список L = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Второй способ.

```
L = [i for i in range(10)]
# список L= [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

c = [c * 3 for c in 'МКМРО']
print (c)
# ['MMM', 'KKK', 'MMM', 'PPP', 'OOO']
```

Случайные числа в списке:

```
from random import randint
L = [randint(10,80) for x in range(10)]
# 10 чисел, сгенерированных случайным образом в диапазоне (10,80)
```

Ввод списка (массива):

Простой вариант ввода списка и его вывода:

```
L=[]
L = [ int(input()) for i in range(5) ]      # при вводе 1 2 3 4 5
print (L)          # вывод: 1 2 3 4 5
```

Функция int здесь используется для того, чтобы строка, введенная пользователем, преобразовывалась в целые числа.

Вывод списка:

```
print (L)          # вывод целого списка (массива)
for i in range(5):
print ( L[i], end = " " )       # поэлементный вывод списка (массива)
```

3. Решение задач.

Списки (циклы). Составить программы к следующим задачам

1). Для каждого элемента списка целых чисел найти сумму его цифр.

2). Сформировать возрастающий список из чётных чисел (количество элементов 45). Найти среднее арифметическое этих чисел.

3). Пользователь вводит число. Определить, содержит ли список данное число x. Если содержит, то вывести на экран число 7145, если не содержит, то вывести на экран число 5741.

4). Найдите сумму и произведение элементов списка. Результаты вывести на экран.

5). Найти наибольший элемент списка и вывести его на экран.

4. Подведение итогов.

Задачи для самостоятельного решения (домашняя работа).

1. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от -20 до 40. Выведите на экран компьютера созданный список. В списке положительные элементы уменьшите вдвое, а отрицательные замените на значения их индексов.

2. Из списка произвольных чисел A[10] сформируйте два списка, содержащих номера положительных и отрицательных элементов.

Вопросы для самоконтроля:

1. Как сформировать список целых чисел?
2. Может ли список содержать элементы разных типов?
3. Можно ли обратиться к n-ому элементу списка?

Литература:

1. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 9.
Вложенные списки. Матрицы

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.
3. Решение задач.

Для обработки и вывода списка как правило используется два вложенных цикла. Первый цикл по номеру строки, второй цикл по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

```
for i in range(len(A)):
for j in range(len(A[i]):
print(A[i][j], end = ' ')
print()
```

То же самое, но циклы не по индексу, а по значениям списка:

```
for row in A:
for elem in row:
print(elem, end = ' ')
print()
```

Кроме того, для вывода одной строки можно воспользоваться методом join:

```
for row in A:
print(' '.join(list(map(str, row))))
```

1). Заполните массив целыми числами по образцу в виде таблицы умножения.

```
n=5, m=6
0 0 0 0 0 0
0 1 2 3 4 5
```

0 2 4 6 8 10
0 3 6 9 12 15
0 4 8 12 16 20

2). В заданном списке найти сумму всех элементов.

3). Найти сумму и количество положительных элементов второго и четвертого элемента списка.

4). Транспонировать прямоугольную матрицу.

5). Найти седловые точки списка.

4. Подведение итогов.

ДОМАШНЕЕ ЗАДАНИЕ

1. Найти максимум среди сумм элементов вложенного списка.

2. Дан список mlist = ['Игра престолов', 'Цифровая крепость', 'Практическая статистика для специалистов Data Science']. Замените каждую букву «а» цифрой 1.

Вопросы для самоконтроля:

1. Приведите пример вложенных списков.

2. Чем список отличается от матрицы?

Литература:

1. Гвидо ванн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 10.
Определение функций. Параметры и аргументы. Вызовы функций. Оператор возврата return. Рекурсия.

Цель: Рассмотреть определение функции

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.

```
In [56]: def test():
   ....:     print('in test function')
   ....:
   ....:

In [57]: test()
in test function
```

Основания функций

1. В каком смысле из записи функций?
2. В каком момент Python действительно создает функцию?
3. Что действительно функционирует возврат если в нем нет никакого оператора возврата?
4. Когда код, включенный в оператор определения функции, выполняется?
5. Что не так с проверкой типов объектов, переданных функции?

Ответы

1. Функции являются самым основным способом избежать дублирования кода в Python — заявление кода в функциональном существовании средств только одной копии кода операции, которая, возможно, должна быть обновлена в будущем. Функции являются также основной единицей повторного использования кода в Python – комната кода в функции делает это неоднократно используемым инструментом, позволяющим вызов в различных программах. Наконец, функции позволяют сегментировать сложную систему, признающую управление, каждый из которых может быть разработан отдельно.

2. Функция создается, когда Python достигает оператора определения; этот оператор создает объект функции и адаптирует его имени функции. Обычно это происходит импортом файла модуля, содержащего определение, другой модуль (помните, что импорт приводит к выполнению кода в файле с начала до конца, включая любые операторы определения), но может произойти также, когда определение собрано в интерактивной подсказке или вкладывает капитал в другие операторы, такой как будто.

3. Функцией по умолчанию не возвращает Ни один объект, если поток управления достиг конца тела функции, не встретив какой-либо оператор возврата. Такие функции обычно вызываются посредством операторов выражений как переменная присвоения, их результаты в целом не являются бессмысленными Ни один. Оператор возврата без выражения в нем также не возвращает Ни один.

4. Тело функции (код, включенный в оператор определения функции), выполняется, когда функция вызывается впоследствии посредством выражения вызова. При каждом вызове функции ее тело выполняется снова.

5. Проверка типов объектов, переданных для функционирования на самом деле, уничтожает гибкость кода, ограничивая функцию для работы только с определенными типами. Без таких проверок функция вполне возможно была бы способна для обработки целого диапазона типов объектов — любые объекты, которые поддерживают интерфейс, ожидаемый функцией. (Термин, интерфейс означает ряд методов и операций выражений, которые выполняются функциональным кодом.)

Ответы

1. И лямбда и определение создают объект функции, которая подвергается вызову в будущем. Тем не менее, поскольку лямбда является выражением, она возвращает объект функции, но не адаптирует его имени и может использоваться для вставки определения функции к местам, где определение синтаксически не позволяется. Однако лямбда позволяет возвращать только одно выражение неявно; из-за отсутствия поддержки блока операторов лямбды не подходит для больших функций.

2. Выражения лямбды позволяют «сборку в» небольших единицах исполняемого кода, откладывают их выполнение и предоставляют их состояние в форме стандартных аргументов и переменных от всесторонних областей видимости. Заявление лямбды не рассматривается обязательное; можно всегда писать в обменном определении и относиться для функционирования по имени. Тем не менее, выражения лямбды полезны при встраивании небольших частей кода с отложенным выполнением, которое будет едва использоваться в других местах программы. Они обычно происходят в программах на основе обратных вызовов, таких как графические интерфейсы пользователя и имеют естественное подобие инструментам функционального программирования, это кажется тар и фильтром, которые ожидают функцию обработки.

3. Все три встроенных функции применяют другую функцию к элементам в последовательности (или в другом выполненном с помощью итераций объекте) и накапливают результаты. Тар передает каждый элемент функции и собирает все результаты, фильтр накапливает элементы, для которых функция возвращает истинное значение, и уменьшите, вычисляет уникальное значение приложением функции к диску и элементам после один за другим. В отличие от других уменьшать функция в Python Z.H доступен в functools модуле, но не во встроенном контексте; уменьшите встроенная функция в Python 2. X.

4. Сводки функций, доступных в Python Z.H (в Python 3.0 я буду также следовать за shchy версиями), представлены синтаксическими декораторами аргументов и результатом функции, кто собирается в словаре, адаптированном атрибуту _annotations __ функции. Python не выделяет сводку ни с каким семантическим значением, и просто упаковывает их для потенциального использования другими инструментами.

5. Рекурсивные функции вызывают себя или непосредственно, или косвенно для Цель организации цикла. Они могут быть применены к обходу структур любой формы и также к повторению в целом (хотя в последней роли, частой проще и эффективнее использовать операторы цикла). Рекурсия может довольно часто быть эмулирована или заменена кодом, в котором явные стеки или очереди используются для достижения большего управления обходами.

6. Функции, как правило, должны состоять в том каждый раз, когда возможный маленький и самодостаточный, имейте единственное присвоение и взаимодействуйте с другими компонентами через аргументы входа, и возвратил значения. Они могут также применить к передаче результатов изменяемые аргументы, если изменения ожидаются, и некоторые типы программ содержат другие механизмы передачи данных.

7. Функции могут отступить, отправляют результаты посредством операторов возврата, из-за модификации переданных изменяемых аргументов и установкой глобальных переменных. Глобальные переменные, как правило, не утверждены (кроме особых случаев как многопоточные программы), поскольку они могут сделать код более трудным для понимания и использования. Обычно возвращайтесь, операторы будут наилучшим вариантом, но модификация изменяемых аргументов также приближается (и даже удобно), если это ожидается. Результаты посредством системных механизмов, таких как файлы и сокеты также способны для передачи функций, но это вне продуманных предметов.

Включения и генераторы

1. В какой различие между комнатой включения списка в квадратные скобки и в круглые скобки?
2. Как генераторы и итераторы подключены между собой?
3. Как определить, генерирует ли функция?
4. Что сделано оператором урожая?
5. Как вызовы тар и включения списка соединены между собой? Сравните и выступите против них.

Ответы

1. Включения списка в квадратные скобки входят в полный список в памяти на этот раз. Когда включения списка в обмен расположены в круглых скобках, на самом деле генерируют выражения — они имеют похожий смысл, но не выпускают весь получающийся список сразу. Генерация выражений вместо этого возвращает объект генератора, который выделяет на одном элементе результата за время, когда используются в повторяющемся контексте.

2. Генераторы являются выполненными с помощью итераций объектами, которые автоматически поддерживают протокол повторения — у них есть итератор с методом __ затем _ (затем в Python 2. X), который неоднократно передает следующему элементу в серии результатов и инициирует исключение при достижении конца ряда. В Python мы можем записать производящие функции посредством определения и урожая, генерировав выражения посредством включений в круглые скобки и генерировав объекты с помощью классов, в которых определяется специальный метод по имени __ проходом __.

3. Производящая функция содержит в операторе урожая кода. Всего остальные производящие функции синтаксически идентичны нормальным функциям, но они компилируются Python специальным способом это вызовом для возврата выполненного с помощью итераций объекта генератора. Такой объект сохраняет состояние и местоположение в коде между доставкой значений.

4. Когда оператор урожая присутствует, он вынуждает Python скомпилировать функцию особенно как генерацию; вызовом функция возвращает объект генератора, который поддерживает протокол повторения. Когда оператор урожая выполняется, он отправляет результат поддержать вызывающий код и остановки с сохранением состояния функции; тогда функция может возобновить работу после последнего выполняемого оператора урожая в ответ на вызов встроенной функции следующих или метода __ затем __, инициируемый вызывающим кодом. В более трудных ситуациях отправить метод генерирующегося объекта так же возобновляет эксплуатацию генератора, но также способен для передачи значения, которое является значением урожая выражения. Производящие функции могут содержать также эксплуатацию остановки оператора возврата генератора.

5. Вызов тар подобен для списка включения — они оба делают серию значений, накапливая результаты приложения указанной операции к каждому элементу в последовательности или в другом выполненном с помощью итераций объекте на одном в течение времени. Основное различие – то, что тар применяет вызов функции к каждому элементу, и включения списка применяют любые выражения. Из-за него включения списка более универсальны; они способны для применения выражения вызова функции, это кажется

тар и функцией требований тар для приложения выражений других типов. Включения списка также поддерживают расширенный синтаксис, такой как вложенные циклы для и конструкция если, которые принадлежат категории встроенного функционального фильтра. В Python Z.H встроенная функция тар также отличается по тому, что делает генератор значений; включение списка осуществляет весь получающийся список в памяти сразу.

Задания для студентов:

```
def newfunc(n):
def myfunc(x):
return x + n
return myfunc
new = newfunc(100)  # new - это функция
```

3. Решение задач.

```
function quicksort(array)
  var list less, greater
  if length(array) < 2
    return array
  select and remove a pivot value pivot from array
  for each x in array
    if x < pivot + 1 then append x to less
    else append x to greater
  return concatenate(quicksort(less), pivot, quicksort(greater))
```

Функции. Составить программы к следующим задачам

1). Разработать функцию $\min(a,b)$ для нахождения минимального из двух чисел. Вычислить с помощью него минимальное значение из четырех чисел x, y, z, υ.

2). Разработать функцию $\max(a,b)$ для нахождения максимального из двух чисел. Вычислить с помощью него значение выражения $z = \max(x, 2y - x) + \max(5x + 3y, y)$.

3). Разработать функцию $f(x)$, который вычисляет значение по следующей формуле: $f(x) = x^3 - \sin x$. Определить, в какой из точек a или b, функция принимает наибольшее значение.

4). Разработать функцию $f(x)$, который возвращает младшую цифру натурального числа x. Вычислить с помощью него значение выражения $z = f(a) + f(b)$.

5). Вычислить значение факториала числа n. Реализовать рекурсивную функцию.

4. Подведение итогов.

Задачи для самостоятельного решения (домашняя работа).

Последовательность Fibonacci

Запишите функцию, которая отображает n первые сроки последовательности Fibonacci, определенной:

$$\begin{cases} U_0 = 0 \\ U_1 = 1 \\ U_{n+2} = U_{n+1} + U_n \end{cases}$$

Пример из задачи. *Запишите функцию, вычисляющую экспоненту по формуле*

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Управление реализацией точности вычислений посредством дополнительного ε параметра со значением по умолчанию (необходимо остановить вычисления, когда следующий член отличается от предыдущего меньше, чем на 10^{-10}).

Вызов функции реализации различными способами:

• с одним позиционным параметром (одновременно значение по умолчанию будет использоваться);

• с двумя позиционными параметрами (значение точности будет передано как второй аргумент);

• передав значение как именованный параметр.

Решение для задачи:

```
def EXPONENTA (x, eps =10**( -10)):
ex = 1 # future result
dx = x # increment
```

```
i = 2 # increment number
while abs(dx)>eps :
ex = ex + dx
dx = dx * x / i
i = i + 1
rreturn ex
# Main program
A = float(input('Enter an indicator exhibitors: '))
print( EXPONENTA (A))
print( EXPONENTA (A, 10**( -4)))
print( EXPONENTA (x = A)
```

Пример решение задачи. Сделайте процедуру из функции (вместо того, чтобы возвратить результат посредством оператора возврата, принесите ему в функции посредством функции печати).

Решение для задачи.

```
Def EXPONENTA (x, eps =10**( -10)):
ex = 1 # future result
dx = x # increment
i = 2 # increment number
while abs(dx)>eps :
ex = ex + dx
dx = dx * x / i
i = i + 1
print(ex)
# Main program
A = float(input('Enter an indicator exhibitors: '))
EXPONENTA (A)
```

1. Разработать функцию $f(x)$, который возвращает вторую справа цифру натурального числа x. Вычислить с помощью него значение выражения $z = f(a) + f(b) - f(c)$.

2. Вычислить значение x в степени n. Реализовать рекурсивную функцию.

Вопросы для самоконтроля:

1. Синтаксис функции пользователя.
2. Что мы понимаем под формальными переменными?

3. Как вызвать функцию из основной программы?

4. Для чего нужны функции?

5. Как выражения лямбды и операторов определения соединены друг с другом?

6. Какой смысл применяют лямбду?

7. Сравните и выступите против карты, отфильтруйте и уменьшите.

8. Что такое сводки функций и для использования их?

9. Что такое рекурсивные функции и для применения их?

10. Назовите несколько универсальных принципов проектирования функций.

11. Назовите три или больше способа передачи результатов функций к вызывающему коду.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Практическая работа № 11.
Множества. Работа со словарями. Методы словарей

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.

Разработайте программу, которая осуществляет перевод нескольких слов русского языка на английский.

Сначала оператором slovar=dict() создаем пустой словарь с именем slovar. Далее мы создаем в нашем словаре пары, состоящие из ключей (русские слова) и значений (слова английского языка). Когда пользователь вводит слово, которое он хочет перевести на английский язык, организуется перебор элементов словаря, и если какой-то из них совпал с введенным словом, выводится ответ.

3. Решение задач.

1). Разработайте программу, которая выводит фразу «Привет, программист» на одном из иностранных языков.

Создать словарь из нескольких пар, в которых есть ключ - страна и значение, которым является приветствие. С помощью условного оператора if проверить, соответствует ли введенное название страны ключу в словаре и, если условие оказывается истинным, вывести соответствующее значение.

2). Дано множество имен студентов одной учебной группы. Выяснить, учится ли студент с определенным именем в данной группе.

3). Вывести на экран все элементы множества A, которых нет в множестве B.

4). Даны четыре множества:

A = (0,1,2,3,4,5,6,7,8,9) B = (0,2,4,6,8) C = (1,2,3,4,5) D = (5,6,7,8,9)

Найти элементы, принадлежащие множеству E:

$$E=((A\setminus B)\cap(C\setminus D))\cup((D\setminus A)\cap(B\setminus C))$$

5). Имеется база данных, содержащая информацию о наименовании товара и его стоимости. Определить самый дорогой

товар, самый дешевый товар, обеспечить 10% скидку на весть товар, обеспечьте скидку на дорогой товар. Пусть имеется информация о покупках клиента: наименование товара и его количество. Определить общую стоимость покупки.

4. Подведение итогов.

ДОМАШНЕЕ ЗАДАНИЕ

1. Дан текст. Определите, сколько различных слов содержится в этом тексте.

2. Дан список стран и городов каждой страны. Затем даны названия городов. Для каждого города укажите, в какой стране он находится.

Вопросы для самоконтроля:

1. Назовите служебные слова, используемые для описания множества.

2. Приведите примеры орпераций, используемых над множествами.

3. Что такое словарь?

4. Какие словари Вы знаете?

Литература:

1. Гвидо ванн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Практическая работа № 12
Работа с файлами

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.
3. Решение задач.

Задания для студентов:

```
N=input ( 'Количество элементов :')
N=int(N)
S=0
for i in range (N) :
a=input ( )
a=int(a)
S=S+a
C=S/N
print ('Результат : ' ,C)
```

Задачи для самостоятельного решения (домашняя работа).

Задача 1. Выполните на компьютере программу создания файла согласно варианту задачи. Считайте и распечатайте созданный файл.

А. Вариант 1 Sozdat файл, содержащий данные по ежемесячной зарплате рабочих План. Каждая запись содержит поля – фамилия рабочего, название цеха, сумма зарплаты за месяц. Количество записей – любой.

В. Вычислите общую сумму платежей за месяц для семинара X и также среднего ежемесячного дохода работы этого семинара. Печать для бухгалтера лист для заряда зарплаты рабочим этого семинара.

Вариант. 2 А. Создайте файл, содержащий данные по количеству продуктов, собранных коллекторами за неделю. Каждая запись содержит поля: фамилия коллектора, количество продуктов, собранных им ежедневно в течение шестидневной недели т.е. отдельно – в понедельник, во вторник, и т.д. Количество записей – любой.

В. Запишите программу, выпустив следующую информацию о печати:

фамилия коллектора и общее количество деталей, собранных им за неделю;

фамилия коллектора, кто собрал самое большое количество продуктов, и день, когда он достиг самой высокой производительности труда.

Вариант.3 А. Sozdat файл, содержащий данные по количеству продукт категорий A, B, C собранный рабочим в Структуре месяца записи, имеет поля: фамилия коллектора, название цеха, количество продуктов на категориях, собранных рабочим за месяц. Количество записей – любой.

В. Включая установленные значения цитат sa, сурьмой, a/см для выполненной работы над блоком единицы продукта категорий A, V, S соответственно для издания следующей информации о печати:

— общее количество продуктов категорий A, B, C собранный рабочим семинара x:

— лист зарплаты рабочих семинара x;

— средняя сумма зарплаты сотрудников этого семинара

Вариант. 4 А. Создайте файл, содержащий данные по телефонам подписчиков, Каждая запись имеет поля: фамилия подписчиков, год установки телефона, номера телефона. Любое количество записей.

В. Запишите программу, выпустив информацию следующего взгляда:

— на вводимой фамилии подписчика выпущен номер телефона:

— количество установленных телефонов с XXXX лет определяется. Номер года введен с клавиатурой

Вариант. 5 А. Создайте файл. Содержа данные по диапазону игрушек в магазине. Структура записи: название игрушки, цены. Количество. Возрастные группы, для Пример 2 — 5. т.е. с 2 до 5 лет. Количество записей – любой.

В. Запишите программу, в результате которого выполнения выпущены следующие данные:

— названия игрушек, которые подходят детям с 1 до 3 лет;

— стоимость самой дорогой игрушки и ее имени;

— название игрушки, которую по стоимости Вы не превышаете, трется, и подходит ребенку в возрасте от и до лет. Значения h, а, б для ввода с клавиатуры.

Вариант. 6 А. Создайте файл, содержащий данные по доставке студентами меня курса kompyyuter отдела сессии. Структура записи:

индекс группы, фамилия студента, оценки пяти исследований, знака участия в общественной работе: «Я \— активное участие, «О» – неучастие. Количество записей – 30.

В. Запишите программу передачи студентов группы X на предоставлении. Студент, который получил все оценки «5» и активно участвующий в общественной работе, включается в список на повышенном предоставлении (дополнительный сбор 50%). Не активно участвует — дополнительный сбор 25%. Студенты, которые получили «4» и «5», включаются в список на нормальном предоставлении. Студент, который получил одну оценку «3», но активно участвовал в общественной работе, также включается в список на предоставлении, иначе передайте, не сделан. Индекс группы вводится с Подачей клавиатуры. 7 А. Создайте файл, содержащий данные по доставке студентами сессии.

Структура записи: индекс группы, фамилия студента, оценки пяти исследований и пяти смещений («з» означает смещение. «N» — не передают класс), Количество записей — 25.

В. Запишите программу, выпустив следующую информацию:

— фамилии бедных студентов с признаком индексов групп и количеств долгов:

— GPA, полученный каждым студентом группы x и всей группы в целом

Вариант 8 А. Создайте файл, содержащий данные по персональному набору библиофила. Структура записи: шифр книги, автора, имени, год выпуска, местоположение (количество стойки, корпуса, и т.д.). Любое количество записей.

В. Запишите программу, выпустив следующую информацию:

— местоположение книги автора X имен U.

— список книг автора Z., которые находятся в наборе:

— количество книг выпуска xx годы, доступные в библиотеке

Вариант. 9 А. Создайте файл, содержащий данные по существованию билетов и выполнениям Аэрофлота. Структура

записи: номер рейса, пункт назначения, время отправления, время поступления, количество пустых мест в салоне. Любое количество записей

В. Запишите программу, выпустив информацию следующего взгляда:

- время отправления airПланes в город x;

- существование пустых мест на выполненном в город X со временем отправления B.

Значения x, B войти с клавиатуры;

– A. Sozdat's pitch.10 the file containing data on the range of footwear in shop of firm. Structure of record: article name, quantity, cost of one couple. Any record count. The article begins with letter D for ladies' footwear, M for men's, P for the nursery.

– B. Write the program issuing the following information:

– — about existence and cost of footwear of the article x;

– — the assortment list of ladies' footwear with the indication of the name and the available number of couples of each model.

Вариант 11.

– A. Создать два файла, содержащие данные приблизительно десять нападающих команд хоккея с шайбой Динамо и Спартак соответственно: имена форвардов, количество шайб, брошенных ими, сделанными передачами цели, заработанное уголовное время.

– B. Запишите программу, которая согласно полученным данным из этих файлов создаст новый третий файл, содержащий имя, команду, сумму точек (цели – передачи) для шести лучших плееров и команд к отображаемым именам и индикаторов эффективности хоккеистов.

Вариант 12

– A. Файл содержащий информацию, собирающуюся, что из пяти предлагаемых дисциплин студент для выбора хочет послушать. Структура записи: фамилия студента, индекс группы. 5 дисциплин, GPA прогресса. Выбранная дисциплина отмечена символом 1, по-другому — пространство. Количество записей — 25.

– B. Запишите программу, которая печатает список студентов, желающих слушать дисциплину x. Если число заинтересованных людей превышает 8 человек, то выбрать студентов, имеющих выше GPA прогресса.

– Вариант 13 А. Создайте файл, содержащий данные по отъезду поездов дальнего следования от Казанской станции. Структура записи: количество поезда, станции назначения, время отправления, время в путях, существовании билетов. Любое количество записей.

– В. Запишите, программа, которая позволяет получать следующую справочную информацию:

– время отправления поездов в город X во временном интервале от И до В час.

– существование билетов на поезд с номером XXX.

– Вариант. 14 А. Создайте файл. Содержа информацию о штате института.

– Структура записи: работа фамилии, название отдела, год рождения, продолжительность обслуживания, положения, зарплаты. Любое количество записей.

– В. Запишите программу, которая позволяет получать следующую информацию:

– список сотрудников пенсионного возраста сегодня с признаком продолжительности обслуживания,

– средний опыт, работающий в отделе X.

– Вариант. 15 А. Создайте файл, содержащий информацию о пациентах глазной клиники. Структура записи: фамилия пациента, пола, возраста, место жительства (город), диагноз. Любое количество записей.

– В. Запишите программу, выпустив следующую информацию:

– число нерезидентов, которые прибыли в клинику;

– список пациентов более старше, чем эти X лет с диагнозом В. Введите значения Н и U с клавиатуры.

Контрольные вопросы:

1. Перечислите основные операции над файлами.
2. В каких режимах можно открыть файл?
3. В чем особенность файла, открытого в режиме «w»?
4. Можно ли обратиться к 7 компоненте файла?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Практическая работа № 13
Создание собственных модулей и их импортирование

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.
3. Решение задач.

1. Найдите площадь треугольника с помощью формулы Герона. Стороны задаются с клавиатуры. Реализовать вычисление площади в виде функции, на вход которой подаются три числа, на выходе – площадь. Функция находится в отдельном модуле, где происходит разделение между запуском и импортированием.

2. Составить модуль, содержащий три функции, реализующие следующие задания:

a) Определить, является ли заданное шестизначное число счастливым. (счастливым называют такое шестизначное число, у которого сумма его первых трех цифр равна сумме его последних трех цифр.)

b) Даны два слова. Определить можно или нет из букв слова A составить слово B.

c) Известны данные о сотрудниках фирмы: фамилия и отношение к воинской службе (военнообязанный или нет). Напечатать фамилии всех военнообязанных сотрудников.

Продемонстрировать функциональность модуля.

4. Подведение итогов.

ДОМАШНЕЕ ЗАДАНИЕ

1). Дан прямоугольник (стороны). Создать модуль по работе с прямоугольником. Функционал: нахождение периметра, площади и диагонали прямоугольника. Импортировать модуль, продемонстрировать его функциональность.

Контрольные вопросы:

1. Как создать собственный модуль?
2. Как использовать модуль пользователя?
3. Назовите варианты импорта созданного модуля.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров.- 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Практическая работа № 14
Основные стандартные модули и пакеты в Python. Импортирование модулей. Модуль Numpy.

План практических занятий:

1. Повторение теоретического материала по теме.
2. Разбор готовых программ.
3. Решение задач по теме.

1) Создать векторы (одномерные массивы) размера 10, заполненные нулями, единицами, число 7.

2) Создать вектор размера 10, заполненный нулями, но пятый элемент равен 1.

3) Создать вектор со значениями от 10 до 49 и развернуть его (первый становится последним).

4) Создать матрицу (двумерный массив) 3x3. Найти индексы ненулевых элементов.

5) Создать 3x3 единичную матрицу и массив 3x3x3 со случайными значениями.

6) Создать массив 10x10 со случайными значениями, найти минимум и максимум.

7) Создать случайный вектор размера 30 и найти среднее значение всех элементов

8) Создать матрицу с 0 внутри, и 1 на границах

9) Создать 5x5 матрицу с 1,2,3,4 под диагональю

10) Создать 8x8 матрицу и заполнить её в шахматном порядке

11) Дан массив размерности (6,7,8). Каков индекс (x, y, z) сотого элемента?

12) Перемножить матрицы 5x3 и 3x2

13) Дан массив, поменять знак у элементов, значения которых между 3 и 8

14) Создать 5x5 матрицу со значениями в строках от 0 до 4

15) Создать вектор размера 10 со значениями от 0 до 1, не включая ни то, ни другое. Отсортировать вектор

16) Проверить, одинаковы ли 2 numpy массива

17) Сделать массив неизменяемым

18) Заменить максимальный элемент на ноль

19) Из двух массивов сделать матрицу Коши C (Cij = 1/(xi - yj))

20) Найти ближайшее к заданному значению число в заданном массиве

21) Дан массив (100,2) координат, найти расстояние от каждой точки до каждой

22) Преобразовать массив из float в int

23) Сформировать 2D массив с распределением Гаусса

24) Случайно расположить p элементов в 2D массив

25) Отнять среднее из каждой строки в матрице

26) Отсортировать матрицу по n-ому столбцу

27) Определить, есть ли в 2D массиве нулевые столбцы

28) Дан четырехмерный массив, посчитать сумму по последним двум осям

29) Поменять 2 строки в матрице

30) Посчитать ранг матрицы

4. Подведение итогов.

ДОМАШНЕЕ ЗАДАНИЕ

1) Дан массив 16x16, посчитать сумму по блокам 4x4

2) Нужно посчитать произведение и сумму матриц, скаляров и векторов:

ABC*D + E, где B, C - матрицы 2x2, D, E - вектор-столбцы, A - скаляр (число).

3) На небольшой ярмарке входная плата составляет 1,5 доллара для ребёнка и 4 доллара для взрослого. Однажды за день ярмарку посетило 2200 человек; при этом было собрано 5050 долларов входной платы. Сколько детей и сколько взрослых посетили ярмарку в этот день?

Контрольные вопросы:

1. Что такое Numpy?

2. Какие возможности Numpy для решения систем уравнений Вы знаете?

3. Как можно записать массив в текстовый файл?

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

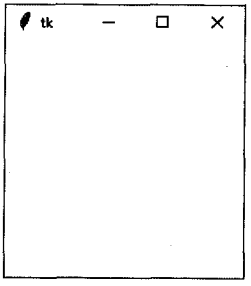
3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

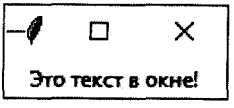
Практическая работа № 15
Работа с виджетами

План практических занятий:
1. Повторение теоретического материала по теме.
2. Разбор готовых программ.

Создание окна
```
import tkinter
window = tkinter.Tk()
window.mainloop()
```

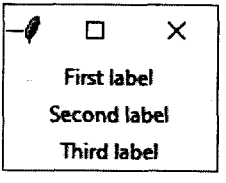


Создание виджета Label
```
import tkinter
window = tkinter.Tk()
label = tkinter.Label(window, text="Это текст в окне!")
label.pack()
window.mainloop()
```

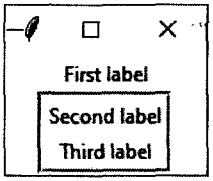


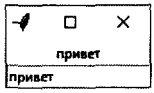
3. Решение задач по теме

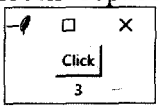
1) Размещение виджетов во фрейме



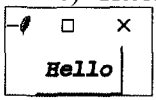
2) Изменение параметров фрейма в момент создания объекта

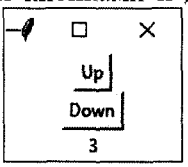


3) Использование виджета (Entry) для ввода данных 

4) Возможности обработки событий при нажатии на кнопку (виджет Button) 

5) Менеджер расположения (геометрии) pack 

6) Изменение параметров виджетов в момент их создания 

7) Работа с двумя кнопками и двумя обработчиками событий (click_up, click_down) 

Используя созданные виджеты, разработать собственное приложение

8). Создайте приложение, состоящее из главного и двух дочерних окон. На каждом из трех окон должны располагаться один или два любых графических объекта.

9). Создайте приложение, в котором меняется размер фрейма в зависимости от того, какая из трех объектов-кнопок была нажата.

10). Напишите скрипт, генерирующий окно с меткой и текстовым полем. После ввода пользователем текста в поле и нажатия Enter, введенный текст должен отображаться в метке.

4. Подведение итогов.

ДОМАШНЕЕ ЗАДАНИЕ

1) Создать программу-калькулятор
2) Создать тестирующую программу

Контрольные вопросы:

1. Что такое виджет?
2. Приведите примеры виджетов.
3. Какими свойствами обладают виджеты.
4. Перечислите классы виджетов, которые Вы знаете.
5. Что такое событие? Для чего оно необходимо?
6. Перечислите события мыши.
7. Приведите пример обработки события клавиатуры.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

ЛАБОРАТОРНЫЕ ЗАНЯТИЯ

Лабораторная работа №1
Введение в среду программирования Python

Цель: Введение в среду программирования Python.

Ключевые слова: Интерактивный режим, основной режим, НЕАКТИВНАЯ среда

Повторение теоретического материала: Запустите НЕАКТИВНЫЙ (первоначально запускается в интерактивном режиме), после которого можно начать писать первую программу. Традиционно, первая программа, которую мы будем иметь, является «Привет мир !». Для записи «Привет мир !» в Python достаточно всего одна строка:

Print ("Привет мир!")

Введите этот код в НЕАКТИВНОМ и нажмите Enter. Результат видим на рис.1:

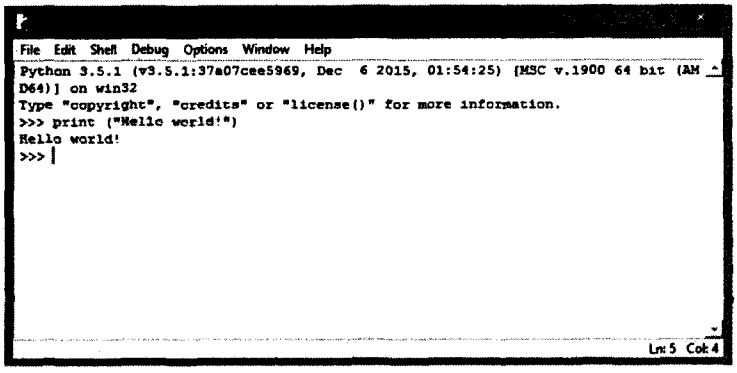


Рис.1.

Мы познакомились с интерактивным режимом, теперь мы будем знакомиться с основным.

Создать новое окно, в НЕАКТИВНОМ интерактивном режиме, select File → New File (or press Ctrl + N).

В окне, которое открывается, введите следующий код:

name = input («Моя первая программа»)

print ("Привет,", name)

В первой строке печатается («Моя первая программа»), ожидает, пока Вы не введете что-нибудь и переходит на следующую строкуи хранит вводимое значение в переменной *name*. Во второй строке мы используем функцию печати, чтобы отобразить текст, в этом случае отобразиться «Привет», и что хранится в переменной «name».

Теперь нажмите F5 (или выберите в меню IDLE Run → Run Module) и удостоверяется это, что мы записали работам. Прежде, чем запуститься НЕАКТИВНЫЙ предложит нам, чтобы сохранить файл. Сохраните туда, где Вы будет удобно, для последующего запуска программы.

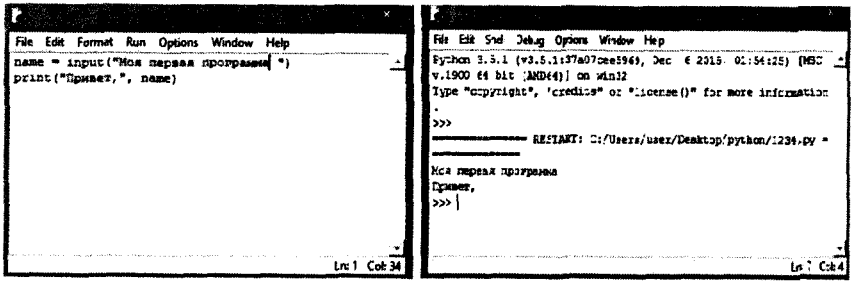


Рис.2.

Пример

Напишите программу, которая запрашивала бы у пользователя:

- ФИО («Ваши фамилия, имя, отчество?»)
- возраст («Сколько Вам лет?»)
- место жительства («Где вы живете?»)

После этого выводила бы три строки:

«Ваше имя»

«Ваш возраст»

«Вы живете в»

Решение

```
a=input('Введите ваши фамилию, имя, отчество ')
b=input('Сколько вам лет? ')
c=input('Где вы живёте? ')
print('Ваше имя ',a)
print('Ваш возраст ',b)
print('Вы живете в ',c)
```

Введите ваши фамилию, имя, отчество **Казагачев Виктор Николаевич**

Сколько вам лет? **45**

Где вы живёте? **Актюбинск**

Ваше имя **Казагачев Виктор Николаевич**

Ваш возраст **45**

Вы живете в **Актюбинск**

Контрольные вопросы:

1. Режим, в который запускается программа?
2. Для чего используется функция печати?
3. Горячая клавиша для запуска программы?

Задания для самостоятельной работы:

Напишите программу, которая запрашивала бы у пользователя:

Вариант 1

Имя, Фамилия, Возраст, Место жительства

- фамилия, имя (“Ваши фамилия, имя?”)
- возраст (“Сколько Вам лет?”)
- место жительства (“Где вы живете?”)

После этого выводила бы три строки:

“Ваши фамилия, имя”

“Ваш возраст”

“Вы живете в”

Вариант 2
Имя, , Дата рождения, Образование
- имя ("Ваше, имя?")
- дата рождения ("Ваша дата рождения?")
- образование ("Где Вы учитесь?")

После этого выводила бы три строки:
"Ваше имя"
"Дата рождения"
"Вы учитесь в "

Вариант 3
Фамилия, Место жительства
- Фамилия("Ваша фамилия?")
- место жительства ("Где Вы живете?")

После этого выводила бы две строки:
"Ваша фамилия"
"Вы живете в"

Вариант 4
Фамилия, Место рождения, любимая музыка
- Фамилия, ("Ваша фамилия?")
- место рождения ("Где Вы родились?")
- музыка("Какая музыка нравится? ")

После этого выводила бы три строки:
"Ваши имя, фамилия"
"Вы родились в"
"Ваша любимая музыка "

Вариант 5
Имя, Фамилия, ФИО мамы, ФИО отца
- ФИО (например, "Ваши фамилия, имя, отчество?")
- возраст ("Сколько Вам лет?")
- место жительства ("Где Вы живете?")

После этого выводила бы три строки:
"Ваши имя, фамилия, отчество"
"Ваш возраст"
"Вы живете в"

Вариант 6

Имя, Любимый предмет в школе, Номер класса

- имя (“Ваше имя?”)
- любимый предмет (“Какой Ваш любимый предмет в школе?”)
- номер класса (“В каком классе Вы учитесь?”)

После этого выводила бы три строки:

“Ваше имя”

“Ваш любимый предмет в школе”

“Вы учитесь в классе номер”

Вариант 7

Имя, Фамилия, Отчество, Хобби

- ФИО (например, “Ваши фамилия, имя, отчество?”)
- хобби (“Чем Вы увлекаетесь?”)

После этого выводила бы две строки:

“Ваши имя, фамилия, отчество”

“Ваше хобби”

Вариант 8

Имя, Фамилия, любимый спорт

- Фамилия, имя (“Ваши фамилия, имя?”)
- образование (“В какой школе Вы учитесь?”)
- ФИО Вашего преподавателя по информатики (“ФИО Вашего преподавателя по информатики?

После этого выводила бы три строки:

“Ваши имя, фамилия”

“Вы учитесь в школе номер: “

“ФИО Вашего руководителя по информатике “

Вариант 9

Имя, Фамилия, Любимый предмет в школе (в институте), ФИО классного руководителя (кура

- Фамилия, имя (“Ваши фамилия, имя?”)
- любимый предмет в школе (“Какой Ваш любимый предмет в школе?”)
- ФИО классного руководителя (“ФИО Вашего классного руководителя?”)

После этого выводила бы три строки:

“Ваши имя, фамилия”
“Ваш любимый предмет в школе “
“ФИО Вашего классного руководителя”

Вариант 10
Имя, Фамилия, Возраст, Дата рождения
• Фамилия, имя (“Ваши фамилия, имя?”)
• возраст (“Сколько Вам лет?”)
-дата рождения (“Когда Вы родились?”)
После этого выводила бы три строки:
“Ваши имя, фамилия”
“Ваш возраст”
“Дата Вашего рождения”

Вариант 11
Имя, Фамилия, Место жительства, Месторождения
• Фамилия, имя (“Ваши фамилия, имя?”)
• место рождения (“Где Вы родились?”)
• место жительства (“Где Вы живете?”)
После этого выводила бы три строки:
“Ваши имя, фамилия”
“Вы родились в”
“Вы живете в”

Вариант 12
Имя, Фамилия, Возраст, Номер телефона
• Фамилия, имя (“Ваши фамилия, имя?”)
• возраст (“Сколько тебе лет?”)
• номер телефона (“Номер Вашего телефона?”)
После этого выводила бы три строки:
“Ваши имя, фамилия”
“Ваш возраст”
“Ваш номер телефона”

Вариант 13
Имя, Фамилия, Страна, Край , Город
• Фамилия, имя (“Ваши фамилия, имя?”)
• страна (“В какой стране Вы живете?”)

• город (“В каком городе Вы живете?”)
После этого выводила бы три строки:
“Ваши имя, фамилия”
“Вы живете в стране”
“Вы живете в крае”
“Вы живете в городе”

Вариант 14
Имя, Фамилия, ФИО Вашего классного руководителя
• Фамилия, имя (“Ваши фамилия, имя?”)
• ФИО Вашего куратора (“ФИО Вашего куратора?”)
После этого выводила бы три строки:
“Ваши имя, фамилия”
“ФИО Вашего руководителя по информатике”
“ФИО Вашего классного руководителя”

Литература:

1. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лабораторная работа № 2
Математические операции в Python

Цель: Познакомиться с алфавитом Python, его структурой, основными математическими операциями.

Ключевые слова: Алфавит, типы данных, структура.

Повторение теоретического материала: Даны вещественные числа a и b. Вычислите функцию

a=input(); a=int(a);
b=input(); b=int(b);
import math
c=math.pow(a,b)+math.sqrt(a);
print(c);

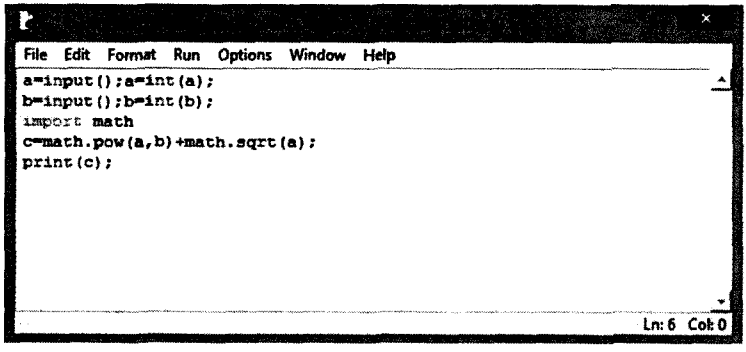


рис. 3

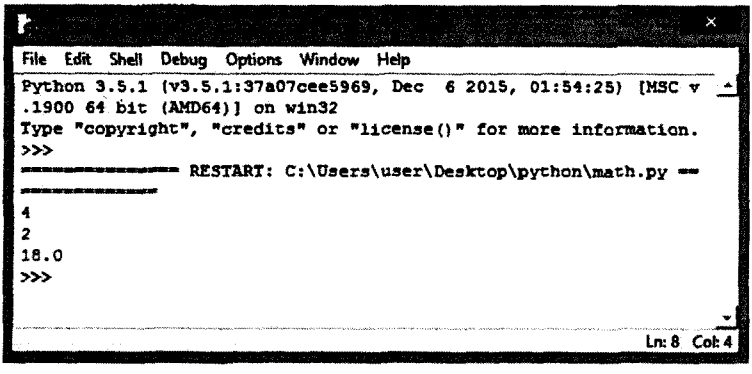


рис. 4

Пример применения операций над целыми числами

```
x = 5
y = 2
z = 3
x+y = 7
x-y = 3
x*y = 10
x/y = 2.5
x//y = 2
x%y = 1
-x= -5
abs(-x) = 5
divmod(x,y) = (2, 1)
x**y = 25
pow(x,y,z) = 1
```

Пример применения операций над вещественными числами

```
x = 5.5
y = 2.3
x+y = 7.8
x-y = 3.2
x*y = 12.649999999999999
x/y = 2.3913043478260874
x//y = 2.0
x%y = 0.9000000000000004
-x= -5.5
abs(-x) = 5.5
divmod(x,y) = (2.0, 0.9000000000000004)
x**y = 50.44686540422945
```

Наиболее часто используемые функции модуля math

math.ceil(x)	Возвращает ближайшее целое число большее, чем x
math.fabs(x)	Возвращает абсолютное значение числа x
math.factorial(x)	Вычисляет факториал x
math.floor(x)	Возвращает ближайшее целое число меньшее, чем x
math.exp(x)	Вычисляет e**x
math.log2(x)	Логарифм по основанию 2
math.loglO(x)	Логарифм по основанию 10
math.log(x[, base])	По умолчанию вычисляет логарифм по основанию e, дополнительно можно указать основание логарифма
math.pow(x, y)	Вычисляет значение x в степени y
math.sqrt(x)	Корень квадратный от x

Тригонометрические функции модуля math

math.cos(x)	Возвращает cos числа X
math.sin(x)	Возвращает sin числа X
math.tan(x)	Возвращает tan числа X
math.acos(x)	Возвращает acos числа X
math.asin(x)	Возвращает asin числа X
math.atan(x)	Возвращает atan числа X

Пример применения функций над числами

В программе определена переменная x, содержащая целое число. Значение переменной выводится командой print() на экран.

В переменную z помещается результат выполнения тригонометрической функции мод

Затем командой print() выводится сообщение в виде используемой функции и её аргумумент – результат её выполнения.

```
import math
x=1
print('x =',x)

z=math.cos(x)
print('math.cos(',x,') =',z)

z=math.sin(x)
print('math.sin(',x,') =',z)

z=math.tan(x)
print('math.tan(',x,') =',z)

z=math.acos(x)
print('math.acos(',x,') =',z)

z=math.asin(x)
print('math.asin(',x,') =',z)

z=math.atan(x)
print('math.atan(',x,') =',z)
```

Пример программы с использованием тригонометрических функций модуля

```
>>>
x = 1
math.cos( 1 ) = 0.5403023058681398
math.sin( 1 ) = 0.8414709848078965
math.tan( 1 ) = 1.5574077246549023
math.acos( 1 ) = 0.0
math.asin( 1 ) = 1.5707963267948966
math.atan( 1 ) = 0.7853981633974483
>>>
```

Константы:
math.pi – число Pi.
Math.e – число e (экспонента).

Пример: Напишите программу, которая бы вычисляла заданное арифметическое выражение заданных переменных. Ввод переменных осуществляется с клавиатуры. Вывести результат с 2-мя знаками после запятой.

$$Z = \frac{9\pi \cdot t + 10\cos(x)}{\sqrt{t} - |\sin(t)|} * e^{x}$$

Решение:

```
x=int(input("Введите переменную x:"))
t=int(input("Введите переменную t:"))
z=((9*math.pi*t+10*math.cos(x))/(math.sqrt(t)-math.fabs(math.sin(t))
print("z = {0:.2f}".format(z))
Введите переменную x:10
Введите переменную t:1
z = 2762685.71
>>>
```

Контрольные вопросы:
1. Что такое алгоритм?
2. Перечислите свойства алгоритма.
3. Это означает сервис для интервала?

Задания для самостоятельной работы:

$$Z = \frac{9\pi \cdot y + 10\cos(x)}{\sqrt{y} - |\sin(y)|} * e^{x}$$

Вариант 1.

X=-0,93; y=8,3

Вариант 2.

X =5,5; y =1,2;

Вариант 3.

X =-11,8; y =0,35

Вариант 4.

X=5,5; y=-0,57

Вариант 5.

X =-7,85 y =0,13

Вариант 6.

X=1,13 y=-5,75

Вариант 7.

X=-7,65 y=3,51

Вариант 8.

X =-2,25 y =11,7

Вариант 9.

X =3,13 y =-8,1

Вариант 10.

X=5,9 y =-8,66

Вариант 11.

X=6,65 y=-5,8

Вариант 12.

X=0,53 y=4,3

Вариант 13.

X=0,77 y=0,89

Вариант 14.

X= 1.5 y= -0.176

Вариант 15.

X=1.34 y= 1.34

Литература:

1. Гвидо ванн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лабораторная работа № 3
Ветвление

Цель: Изучить использование условный оператора if, составить алгоритм ветвления программы.

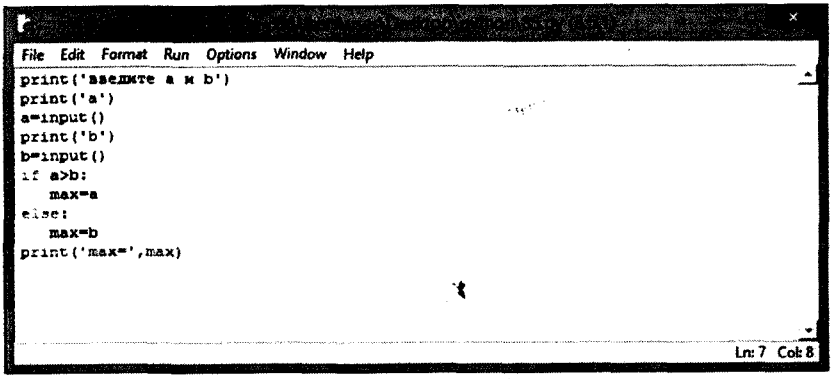
Ключевые слова: Выбор, условие перехода.

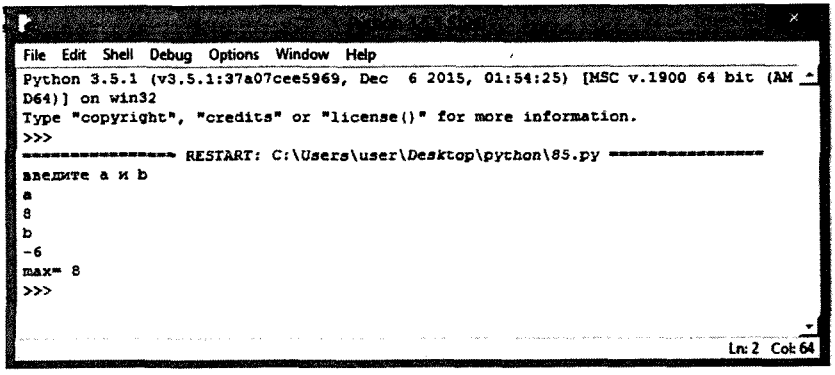
Повторение теоретического материала: Условный оператор if-elif-else (также названный переходящим оператором) основной предпочтительный инструмент в Python. Она выбирает что действие работать, в зависимости от значения переменных во время условий испытания. Во-первых, если зарегистрированная часть условного выражения еще может далее сопровождаться одной или несколькими дополнительными частями elif, и наконец дополнительной частью. Общая форма условного оператора, если запись следующие:

```
if условие 1:
   UNIT1
elif условие2:
    blok2
uslovie3 else:
    Box 3
```

Пример 1: Из двух чисел, выбрать большее. Run IDLE, откройте новое окно и введите код:

```
print ( 'Enter a u b')
print ( 'a')
a = input ()
print ( 'b')
b = input ()
if a> b:
  max = a
else:
  max = b
print ( 'max =', max)
```





Пример 2:

Дано 3 числа. Найти минимальное среди них и вывести на экран.

Решение

Командами

a=input(‘’)

b=input(‘’)

c=input('')

Введём три числа, присвоив значения переменным a, b, c.

Условной конструкцией if-else проверим на истинность логическое выражение a<b. Если истинно, то переходим на проверку логического выражения a<c. Если оно истинно, то присвоим «y» присвоим значение переменной «a», т.е. «a» будет минимальным, а иначе «y» присвоим значение переменной «c».

Если в начале логическое выражение a<b оказалось ложным, то переходим на проверку логического выражения b<c.

Если оно истинно, то «y» присвоится значение переменной «b», иначе «c».

Командой print() выводим минимальное значение.

```
#нахождение минимального из 3-х чисел
a=input('Введите целое число \n')
b=input('Введите целое число \n')
c=input('Введите целое число \n')
if a<b:
    if a<c:
        y=a
    else:
        y=c
else:
    if b<c:
        y=b
    else:
        y=c
print('Минимальное:',y)
```

```
Введите целое число
2
Введите целое число
5
Введите целое число
1
Минимальное: 1
```

Задания для самостоятельной работы:

1) $f(x)=\begin{cases}5x^2+6, \text{если } -2<x<5\\ x^3+7, \text{если } x\geq 5\\ -1, \text{если } x\leq -2\end{cases}$

2) $f(x)=\begin{cases}\sqrt{x^3+5}, \text{если } x\geq 0\\ 3x^4+9, \text{если } -3<x<0.\end{cases}$

3) $f(x)=\begin{cases}\sqrt[3]{x^2+6x}, \text{если } -4<x\leq 5\\ x^5+3.5, \text{если } x>5\end{cases}$

4) $f(x)=\begin{cases}x\cos x, \text{если } x<122\\ 5x^2+1.7, \text{если } x\geq 122\end{cases}$

5) $f(x)=\begin{cases}x^3\cos x, \text{если } 0<x<2\\ 3x^4+7, \text{если } x\geq 2\\ \sqrt{5x^2+16}, \text{если } x\leq 0.\end{cases}$

6) $f(x)=\begin{cases}xtgx-\sin x, \text{если } x<1.5\\ x^3+\sin x+7, \text{если } 1.5\leq x<2.5\\ 3x^3+5, \text{если } x\geq 2.5\end{cases}$

7) $f(x)=\begin{cases}\sqrt{3x^2+4}, \text{если } 0<x<1\\ 5-\sin^2 x, \text{если } x\geq 1\end{cases}$

8) $f(x)=\begin{cases}\sqrt{x^2+|x|}, \text{если } -5<x<0\\ 5x^2+\cos x, \text{если } 0\leq x\leq 2\end{cases}$

9) $f(x)=\begin{cases}x\cos^2 x+\sin x, \text{если } 0\leq x<1\\ \sqrt[3]{x^2+6\sin x}, \text{если } 1\leq x\leq 2\\ 1.7x^3+7, \text{если } x>2\end{cases}$

10) $f(x)=\begin{cases}\sin x+\sqrt{x^2+1.2}, \text{если } x<0\\ tg^2 x+\cos x+35, \text{если } 2<x\leq 6\end{cases}$

11. Даны три целых числа. Выбрать из них те, которые принадлежат интервалу [1,3].

12. Дан номер года (положительное целое число). Определить количество дней в этом году, что обычный год насчитывает 365 дней, а високосный — 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 — являются).

13. Написать программу вычисления стоимости покупки с учетом скидки. Скидки предоставляется в том случае, если сумма покупки больше 500 руб., в 5% - если сумма больше 1000 руб.

14. Написать программу, которая бы по введенному номеру единицы измерения (1 — килограмм, 2 – миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер) и массе M выдавала соответствующие массы в килограммах.

15. Найти косинус минимального из 4 заданных чисел.

16. Вывести на экран синус максимального из 3 заданных чисел.

17. Даны три стороны одного треугольника и три стороны другого треугольника. Определить, являются эти треугольники равновеликими, т. Е. имеют ли они равные площади. Если это верно, вывести «Foul!!!»

18. Составьте программу подсчёта площади равнобедренного треугольника. Если площадь –треугольника чётная, разделить её на 2, в противном случае вывести сообщение «Не делиться на 2!»

19. Составить программу, которая по данному числу (1-12) выводит название соответствующего месяца на английском языке.

20. Составить программу, осуществляющую перевод величин из радианной меры в градусную, и наоборот. Программа должна запрашивать, какой перевод нужно осуществить, и выполнить указанное действие.

21. Дано три числа. Найти количество положительных чисел среди них;

22. Если действительные числа x и y – одного знака, найти их среднее геометрическое, в противном случае найти их среднее арифметическое.

23. Определить, существует ли прямоугольный треугольник со сторонами x,y,z. Если – да, найти его площадь.

24. Определить, существует ли треугольник с длинами сторон a, b, c. Если – да, вычислить площадь по формуле Герона.

Формула Герона имеет вид: S=p(p-a)(p-b)(p-c), где p=12(a+b+c)

25. Вычислить значение функции f(x), если

$$f(x)=\begin{cases} 0{,}5-^{4} x\ , & x \geq 0 \\ \dfrac{\sin^2 x^2}{x+1}, & x<0 \end{cases}$$

Контрольные вопросы:

1. Какую инструкцию называют неполным ветвлением?
2. Каковы основные операторы, описывают ветвящийся процесс?
3. Что базовая структура записи является условным оператором?

Литература:

1. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лабораторная работа № 4
Циклы While и For

Цель: познакомиться с циклическими конструкциями.
Ключевые слова: счетчик, истина, тело цикла.

Повторение теоретического материала: В Python существуют два типа цикличных выражений:
Цикл while .
Цикл for
Цикл While в Python может быть описан следующей схемой:

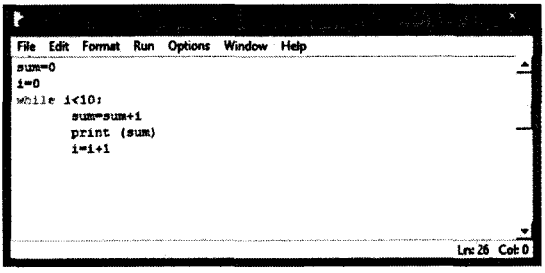
While *< Условие >:* ***or*** ***while (a <b) and (b <c):***
Тело цикла {} ***Тело цикла {}***

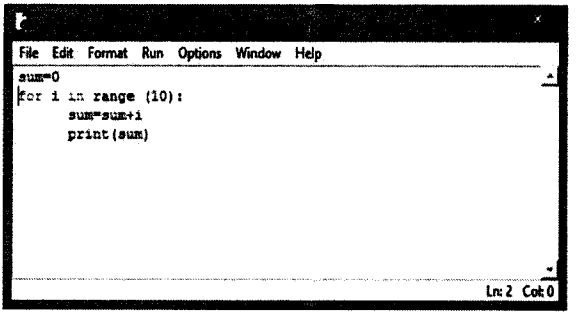
Структура цикла for Python может быть описана следующей схемой:

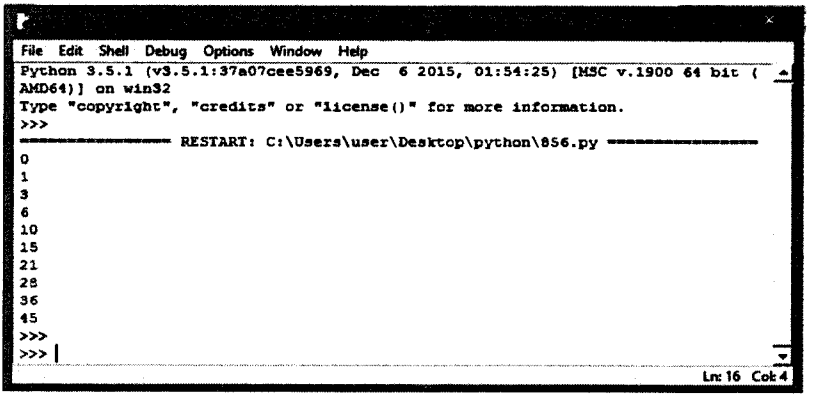
for < переменная > in <Конечное значение >:
Тело цикла {}

Пример: Числа от 0 до 9 (включая 9). Найдите сумму чисел.

Цикл for	и	**цикл while**
Sum = 0		*sum = 0*
i = 0 for I		*in range (10)*
While i <10:		*sum = sum + i*
Sum = sum + i		*print (sum)*
print (sum)		
i = i + 1		







Задания для самостоятельной работы:

Вариант 1

Дано вещественное число – цена 1 кг конфет. Вывести стоимость 1, 2, ... 10 кг конфет. Решить задачу используя циклическую конструкцию for.

Дана непустая последовательность целых чисел, оканчивающаяся нулем.

Найти:

а) сумму всех чисел последовательности;

б) количество всех чисел последовательности

Решить задачу используя циклическую конструкцию while.

Вариант 2

Даны два числа A и B (A<B). Найти сумму всех целых чисел от A до B включительно. P< задачу используя циклическую конструкцию for.

Дана последовательность отрицательных целых чисел, оканчивающаяся положитель числом. Найти среднее арифметическое всех чисел последовательности (без учета положительным числа).

Решить задачу используя циклическую конструкцию while.

Вариант 3

1. Даны два числа A и B (A<B). Найти сумму квадратов всех целых чисел от A до B включи Решить задачу используя циклическую конструкцию for.

2. Дана последовательность из n целых чисел. Первое число в последовательности чётно сумму всех идущих подряд в начале последовательности чётных чисел. Условный опера использовать

Решить задачу используя циклическую конструкцию while.

Вариант 4

- Найти среднее арифметическое всех целых чисел от a до 200 (значения a и b вводятся с клавиатуры; a< 200). Решить задачу используя циклическую конструкцию for.
- Дана последовательность из n вещественных чисел, начинающаяся с положительного Определить, какое количество положительных чисел записано в начале последователь! Условный оператор не использовать. Решить задачу используя циклическую конструкцию while.

Вариант 5

- Найти сумму всех целых чисел от a до b (значения a и b вводятся с клавиатуры; b >a). Решить задачу используя циклическую конструкцию for.
- Дано целое число N (> 0), являющееся некоторой степенью числа 2: $N = 2^{Л}K$ Найти целое — показатель этой степени.

Решить задачу используя циклическую конструкцию while.

Вариант 6

- Найти сумму квадратов всех целых чисел от а до 50 (значение а вводится с клавиатуры Решить задачу используя циклическую конструкцию for.
- Дано целое число N (> 1). Найти наименьшее целое число K, при котором выполняется неравенство 5ЛК > N.

Решить задачу используя циклическую конструкцию while.

Вариант 7

1. Дана непустая последовательность целых чисел, оканчивающаяся нулем.

Найти:

а) сумму всех чисел последовательности;

б) количество всех чисел последовательности.

Решить задачу используя циклическую конструкцию for.

2.Дано целое число N (> 1). Найти наибольшее целое число K, при котором выполняется неравенство $2^{Л}K > N$.

Решить задачу используя циклическую конструкцию while.

Вариант 8

1. Дана последовательность из n вещественных чисел. Первое число в последовательное нечетное. Найти сумму всех идущих подряд в начале последовательности нечетных чис Условный оператор не использовать. Решить задачу используя циклическую конструкц

2. Дано целое число N (> 0). Используя операции деления нацело и взятия остатка от деле количество и сумму его цифр.

Решить задачу используя циклическую конструкцию while.

Вариант 9

1. Среди чисел 1, 4, 9, 16, 25, ... найти первое число, большее n. Решить задачу используя циклическую конструкцию for.

2. Среди чисел 1, 5 10, 16, 23, ... найти первое число, большее n. Условный оператор не исп Решить задачу используя циклическую конструкцию while.

Вариант 10

1. Известны оценки по физике каждого из 20 учеников класса. Определить среднюю оце: Решить задачу используя циклическую конструкцию for.

2. Дано число A (> 1). Вывести наибольшее из целых чисел K, для которых сумма 1 + 1/2 + . будет меньше A, и саму эту сумму.

Решить задачу используя циклическую конструкцию while.

Вариант 11

1. Известно сопротивление каждого из элементов электрической цепи. Все элементы сое последовательно. Определить общее сопротивление цепи. Решить задачу используя цик конструкцию for.

2. Дано целое число N (> 0). Найти наибольшее целое число K, квадрат которого не превос КЛ2 < N. Функцию извлечения квадратного корня не использовать.

Решить задачу используя циклическую конструкцию while.

Вариант 12

1. Известны оценки по физике каждого ученика двух классов. Определить среднюю оцег каждом классе. Количество учащихся в каждом классе одинаковое. Решить задачу испол циклическую конструкцию for.

2. Выведите на экран для числа 2 его степени от 0 до 20 Решить задачу используя циклическую конструкцию while.

Вариант 13

1. В области 12 районов. Известны количество жителей (в тысячах человек) и площадь 0 каждого района. Определить среднюю плотность населения по области в целом. Решить используя циклическую конструкцию for.

2. Мой богатый дядюшка подарил мне один доллар в мой первый день рождения. В кажд рождения он удваивал свой подарок и прибавлял к нему столько долларов, сколько лет исполнилось. Написать программу, указывающую, к какому дню рождения подарок пре 100$.

Решить задачу используя циклическую конструкцию while.

Вариант 14

1. Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько клеток через 3, 6, 9, ..., 24 часа, если первоначально была одна амеба. Решить задачу используя циклическую конструкцию for.

2. Вывести таблицу значений функции y = -0.5x + x. Значения аргумента (x) задаются мин максимумом и шагом. Например, если минимум задан как 1, максимум равен 3, а шаг 0. вывести на экран изменение x от 1 до 3 с шагом 0.5 (1, 1.5, 2, 2.5, 3) и значения функции (у каждом значении x.

Решить задачу используя циклическую конструкцию while.

Вариант 15

4. Начав тренировки, лыжник в первый день пробежал 10 км. Каждый следующий день увеличивал пробег на 10% от пробега предыдущего дня. Определить:

а) пробег лыжника за второй, третий, ..., десятый день тренировок;

б) какой суммарный путь он пробежал за первые 7 дней тренировок.

Решить задачу используя циклическую конструкцию for.

5. Найти сумму и произведение цифр, введенного целого числа. Например, если введено то сумма его цифр равна 10 (3+2+5), а произведение 30 (3*2*5).

Решить задачу используя циклическую конструкцию while.

Контрольные вопросы:

1. В каких случаях используются операторы цикла?
2. Какие способы организации циклов в Python?
3. В каком случае удобно использовать цикл с параметром?

Литература:

1. Гвидо ванн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лабораторная работа № 5
Работа со строками

Цель работы: познакомится с методами работы со строками.

Строка — базовый тип представляющий из себя неизменяемую последовательность символ «string» — «строка».

Функции и методы работы со строками

Функция или метод	Назначение
S1 + S2	Конкатенация (сложение строк)
S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза
len(S)	Длина строки
S.join(список)	Соединение строк из последовательности str через разделитель, задан строкой
S1.count(S[, i, j])	количество вхождений подстроки s в строку s1. Результатом является ч Можно указать позицию начала поиска i и окончания поиска *j*
S.find(str, [start], [end])	Поиск подстроки в строке. Возвращает номер первого вхождения или -
S.index(str, [start], [end])	Поиск подстроки в строке. Возвращает номер первого вхождения или е ValueError
S.rindex(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения ил вызывает ValueError
S.replace(шаблон, замена)	Замена шаблона
S.split(символ)	Разбиение строки по разделителю
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру

Ниже приведена программа, демонстрирующая использование функций и методов работы строками

```
File  Edit  Format  Run  Options  Window  Help
s1="Пропаганда"
s2="Сенсация"
s3="Сенсация*Сенсация*Сенсация*Сенсация"
s4='ОхОхОхАх'
print('s1 = ',s1)
print('s2 = ',s2)
print('s3 = ',s3)
print('s4 = ',s4)
print('s1+s2 = ',s1+s2) #сложение двух строк
print('s1*3 = ',s1*3)   #умножение строки на 3, т.е.строка выведется 3 раза
print('s1[2] = ',s1[2]) #вывод элемента строки s1 с индексом 2
print('s1[2,4] = ',s1[2:4]) #извлечение среза строки s1 начиная с индекса 2
                            #и заканчивая индексом 4
print('s3.count = ',s3.count(s2))   #количество вхождений подстроки s2 в S3,
                                    #в результате выведтеся число
print('s1.find(''a'') = ',s1.find('a'))  #поиск подстроки 'a' в строке s1
                                         #результатом будет номер первого вхождения
print('s1.index(''n'') = ',s1.index('n'))#поиск подстроки 'n' в строке s1
                                         #результатом будет номер первого вхождения
print('s1.rindex(''д'') = ',s1.rindex('a'))#поиск подстроки 'a' в строке s1
                                           #возвращает номер последнего вхождения
print('s4.replace(''Ох'',''Ах'',2) = ',s4.replace('Ох','Ах',2))#замена шаблона. Стр
                                           #строка 'Ах' - это замена
                                           #в строке 4 последоваетльность 'Ох' буд
                                           #на 'Ах' с шагом 2
print('s3.split(''*'') = ',s3.split('*'))#разбиение по разделителю *
print('s1.upper = ',s1.upper())#перевод символов в верхний регистр
print('s1.lower = ',s1.lower())#перевод символов в нижний регистр
```

Пример программы на Python

```
s1 =  Пропаганда
s2 =  Сенсация
s3 =  Сенсация*Сенсация*Сенсация*Сенсация
s4 =  ОхОхОхАх
s1+s2 =  ПропагандаСенсация
s1*3 =  ПропагандаПропагандаПропаганда
s1[2] =  о
s1[2,4] =  оп
s3.count =  4
s1.find(а) =  4
s1.index(п) =  3
s1.rindex(д) =  9
s4.replace(Ох,Ах,2) =  АхАхОхАх
s3.split(*) =  ['Сенсация', 'Сенсация', 'Сенсация', 'Сенсация']
s1.upper =  ПРОПАГАНДА
s1.lower =  пропаганда
```

Пример

Проверить, будет ли строка читаться одинаково справа налево и слева направо (т. Е. является палиндромом).

Решение

Сначала введём строку командой: s=input('Введите строку ').

Затем определим логическую переменную flag и присвоим ей значение 1: flag=1.

Для начала в введённой строке нужно удалить пробелы. Для этого воспользуемся циклической конструкцией for, которая выполнится столько раз, какую имеет длину строка. Длину строки определим функцией len(s).

В теле цикла будем проверять следующее условие: s[i]!=' '. Данное логическое выражение будет истинно в том случае, если i-ый элемент строки не будет равен пробелу, тогда выполнится строка, следующая после двоеточия: string+=s[i].

К сроке string, которая была объявлена в начале программы, будет добавляться анн вольно уже без пробелов.

Для проверки строки на «палиндром» воспользуемся циклической конструкцией for.

Длина половины строки находится делением нацело на 2. Если количество символов нечетное, то стоящий в середине не учитывается, т.к. его сравниваемая пара – он сам.

Количество повторов цикла равно длине половины строки. Длину строки определим функцией, где аргумент введённая нами строка s. Зная длину строки, можно вычислить количество потоков цикла. Для этого целочисленно разделим длину строки на 2: len(s)//2.

Для задания диапазона для цикла используем функцию range(), в которой аргументом будет половина длины строки:

```
range(len(s//2 )).
For i in range(len(s//2 )).
```

Если символ с индексом i не равен «симметричному» символу с конца строки (который находят путем индексации с конца)

```
if s[i] != s[-1-i],
```

Далее, при помощи условной конструкции if-else в зависимости от значения flag либо – 0, либо выводится сообщение, что строка палиндром, либо нет.

Пример программы на Python

```
s=input('Введите строку \n')
flag=1
string=''
for i in range(len(s)):
    if s[i]!=' ':
        string+=s[i]
print(string)
for i in range(len(s)//2):
    if string[i]!=string[-i-1]:
        flag=0
        break
if flag: print('Палиндром')
else: print('не палиндром')
```

Результат выполнения программы

```
Введите строку
а роза упала на лапу азора
арозаупаланалапуазора
Палиндром
```

Задания для самостоятельной работы (по вариантам)

Вариант 1

Дана строка, содержащая русскоязычный текст. Найти количество слов, начинающихся с буквы а.

Вариант 2

В строке заменить все двоеточия (☺ знаком процента (%). Подсчитать количество замен.

Вариант 3

В строке удалить символ точку (.) и подсчитать количество удаленных символов.

Вариант 4

В строке заменить букву(а) буквой (о). Подсчитать количество замен. Подсчитать, сколько сколько букв в строке.

Вариант 5

В строке заменить все заглавные буквы строчными.

Вариант 6

В строке удалить все буквы «а» и подсчитать количество удаленных символов.

Вариант 7

Дана строка. Преобразовать ее, заменив звездочками все буквы «п», встречающиеся среди п-символов. Здесь n – длина строки.

Вариант 8

Дана строка, заканчивающаяся точкой. Подсчитать, сколько слов в строке.

Вариант 9

Определить, сколько раз в тексте встречается заданное слово.

Вариант 10

Дана строка-предложение на английском языке. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы.

Вариант 11

Дана строка. Подсчитать самую длинную последовательность подряд идущих букв «н». Преобразовать ее, заменив точками все восклицательные знаки.

Вариант 12

Дана строка. Вывести все слова, оканчивающиеся на букву «я».

Вариант 13

Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобка.

Вывести на экран все символы, расположенные внутри этих скобок.

Вариант 14

Дана строка. Вывести все слова, начинающиеся на букву «а» и слова оканчивающиеся на букву о.

Вариант 15

Дана строка текста. Подсчитать количество букв «т» в строке.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лабораторная работа № 6
Работа со списками

Цель работы: Изучение одномерных массивов в Python.

Массивы (списки) в Python — это определенное количество элементов одного типа, которые имеют общее имя, и каждый элемент имеет свой индекс — порядковый номер.

Часто для работы с массивами используются списки.

Список (list) – это структура данных для хранения объектов различных типов.

Списки являются упорядоченными последовательностями, которые состоят из различных типов данных, заключающихся в квадратные скобки [ ] и отделяющиеся друг от друга с помощью запятых.

Создание списков на Python.

Создать список можно несколькими способами

1. Получение списка через присваивание конкретных значений.

Так выглядит в коде Python пустой список:

s = [] # Пустой список

Примеры создания списков со значениями:

```
l=[5,75,-4,7,-51]# список целых чисел
l=[1.13,5.34,12.63,4.6,34.0,12.8]# список из вещественных чисел
l=["Оля", "Владимир", "Михаил", "Дарья"]# список из строк
l=["Москва", "Иванов", 12, 124]   # смешанный список
l=[[0, 0, 0], [1, 0, 1], [1, 1, 0]]   # список, состоящий из списков
l=['s', 'p', ['isok'], 2] # список из значений и списка
```

Списки можно складывать (конкатенировать) с помощью знака «+»:

```
l=[1, 3]+[4,23]+[5]
print('l=[1, 3]+[4,23]+[5] =',l)
```

Результат:

```
>>>
l=[1, 3]+[4,23]+[5] = [1, 3, 4, 23, 5]
>>>
```

6. **Создание списка при помощи функции Split().**

Используя функцию split в Python можно получить из строки список.

Stroka =»Привет, страна»

lst=stroka.split(«,»)

```
stroka ="Здравствуй, Дедушка Мороз"  #stroka - строка
lst=stroka.split(",") #lst - список
print('stroka = ',stroka)
print('lst=stroka.split(","):',lst)
```

Результат:

```
stroka =  Здравствуй, Дедушка Мороз
lst=stroka.split(","): ['Здравствуй', ' Дедушка Мороз']
```

7. **Методы списков.**

Метод	Что делает
list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет перед i-ым элементом значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError такого элемента не существует
list.popdi])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удал последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом ведется от start до end)
list.count(x)	Возвращает количество элементов со значением x
list.reverseO	Разворачивает список
list.copyO	Поверхностная копия списка
list.clearQ	Очищает список

Пример программы на Python

```
a=[0,2,2,2,4] #список a
b=[5,6,7,2,9] #список b
print('Исходный список a:',a)
print('Исходный список b:',b)
x=99
y=5

a.append(x)
print('a.append(x):',a)

a.extend(b)
print('a.extend(b):',a)

a.insert(3,x)
print('a.insert(3,x):',a)

a.remove(x)
print('a.remove(x):',a)

print('a.pop(5):',a.pop(5))
print(a)

print('a.index(y,0,len(a)):',a.index(y,0,len(a)))

print('a.count(2):',a.count(2))

a.reverse()
print('a.reverse():',a)

z=a.copy()
print('z=a.copy():',z)

z.clear()
print('z.clear():')
print('z =',z)
```

Результат выполнения программы

```
Исходный список a: [0, 2, 2, 2, 4]
Исходный список b: [5, 6, 7, 2, 9]
a.append(x): [0, 2, 2, 2, 4, 99]
a.extend(b): [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.insert(3,x): [0, 2, 2, 99, 2, 4, 99, 5, 6, 7, 2, 9]
a.remove(x): [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.pop(5): 99
[0, 2, 2, 2, 4, 5, 6, 7, 2, 9]
a.index(y,0,len(a)): 5
a.count(2): 4
a.reverse(): [9, 2, 7, 6, 5, 4, 2, 2, 2, 0]
z=a.copy(): [9, 2, 7, 6, 5, 4, 2, 2, 2, 0]
z.clear():
z = []
```

Пример

Из массива X длиной n, среди элементов которого есть положительные, отрицательные, равные нулю, сформировать новый массив Y, взяв в него только те элементы из X, которые больше заданного числа M. Вывести на экран число M, данный и полученные массивы.

Решение:

```
n=int(input('Введите длину массива\n'))
m=int(input('Введите число M\n'))
x=[]
y=[]
for i in range(n):
    print('Введите ',i,'элемент:')
    x.append(int(input()))
for i in range(n):
    if abs(x[i])>m:
        y.append(x[i])
print('Введённое число M:',m)
print('Массив X:',x)
print('Массив Y:',y)
```

```
Введите длину массива
5
Введите число M
20
Введите  0 элемент:
21
Введите  1 элемент:
22
Введите  2 элемент:
5
Введите  3 элемент:
6
Введите  4 элемент:
8
Введённое число M: 20
Массив X: [21, 22, 5, 6, 8]
Массив Y: [21, 22]
```

В массиве целых чисел все отрицательные элементы заменить на положительные. Вывести исходный массив и получен

Решение:

```
n=int(input('Введите длину массива:'))
a=[]
for i in range(n):
    print('Введите',i,'элемент:')
    a.append(int(input()))
print('Исходный массив:',a)
for i in range(n):
    if a[i]<0:
        a[i]=-a[i]
print('Полученный массив:',a)
```

```
Введите длину массива:5
Введите 0 элемент:
-5
Введите 1 элемент:
-4
Введите 2 элемент:
-6
Введите 3 элемент:
5
Введите 4 элемент:
-7
Исходный массив: [-5, -4, -6, 5, -7]
Полученный массив: [5, 4, 6, 5, 7]
```

Задания для самостоятельной работы:

Вариант 1

1. Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры.

Найти максимальный элемент. Вывести массив на экран в обратном порядке.

2. В массиве действительных чисел все нулевые элементы заменить на среднее арифметическое элементов массива.

Вариант 2

1. Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры.

Найти минимальный элемент. Вывести индекс минимального элемента на экран.

2. Дан массив целых чисел. Переписать все положительные элементы во второй массив, а отрицательные в третий.

Вариант 3

1. В одномерном числовом массиве D длиной n вычислить сумму элементов с нечетными и вывести на экран массив D, полученную сумму.

2. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Вариант 4

1. Дан массив целых чисел. Найти максимальный элемент массива и его порядковый номер.

2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Вариант 5

1. Дан одномерный массив из 10 целых чисел. Вывести пары отрицательных чисел, стоящих по порядку.

2. Дан целочисленный массив размера 10. Создать новый массив, удалив все одинаковые элементы, оставив их 1 раз.

Вариант 6

1. Дан одномерный массив из 10 целых чисел. Найти максимальный элемент и сравнить с ним остальные элементы. Вывести количество меньших максимального и больших максимального элемента.

2. Одномерный массив из 10-и целых чисел заполнить с клавиатуры, определить сумму тех чисел, которые >5.

Вариант 7

1. Дан массив целых чисел. Найти сумму элементов с четными номерами и произведение элементов с нечетными номерами. Вывести сумму и произведение.

2. Переставить в одномерном массиве минимальный элемент и максимальный.

Вариант 8

1. Найдите сумму и произведение элементов списка. Результаты вывести на экран.

2. В массиве действительных чисел все нулевые элементы заменить на среднее арифметическое элементов массива.

Вариант 9

1. Дан одномерный массив, состоящий из N вещественных элементов. Ввести массив с клавиатуры.

Найти и вывести минимальный по модулю элемент. Вывести массив на экран в обратном порядке.

2. Даны массивы A и B одинакового размера 10. Вывести исходные массивы. Поменять места содержимое и вывести в начале элементы преобразованного массива A, а затем — элементы преобразованного массива B.

Вариант 10

1. Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран это число, а иначе сообщение об их отсутствии.

2. Дан одномерный массив из 15 элементов. Элементам массива меньше 10 присвоить нулевые значения, а элементам больше 20 присвоить 1. Вывести на экран монитора первоначальный преобразованный массивы в строчку.

Вариант 11

1 Найти наибольший элемент списка, который делиться на 2 без остатка и вывести его на экран.

2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из четных исходного массива, меньше 10, или сообщить, что таких чисел нет. Полученный массив вывести в порядке возрастания элементов.

Вариант 12

1. Найти наименьший нечетный элемент списка и вывести его на экран.

2. Даны массивы A и B одинакового размера 10. Поменять местами их содержимое и вывести элементы преобразованного массива A, а затем — элементы преобразованного массива B.

Вариант 13

1. Дан одномерный массив целых чисел. Проверить, есть ли в нем одинаковые элементы. Вывести элементы и их индексы.

2. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Вариант 14

1. Найти максимальный элемент численного массива и поменять его местами с минимальным.

2. Программа заполняет одномерный массив из 10 целых чисел числами, считанными с клавиатуры.

Определить среднее арифметическое всех чисел массива. Заменить элементы массива больше среднего арифметического на 1.

Вариант 15

1. Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран эти элементы.

2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лабораторная работа № 7
Кортежи

Цель: ознакомиться с кортежами

Кортеж — это список, элементы которого нельзя менять. Обычно используется в случаях, когда данные постоянны на всем протяжении выполнения программы. К ним можно отнести различные константы, данные о совершенных сделках и т.п. Кортеж заключается в круглые скобки (...) и также может быть многомерным.

Создание кортежа. Так же как и для списков, кортеж можно задать в виде литерала.

Пример 1.

```
K=(1,2,3,'Python')
к=
(1,2,3,('Python','C#','Delphi'),'plus@mail.ru')
```

Кортеж можно создать с помощью функции `tuple ()`.

Пример 2.

```
Print(tuple('tuple'))
s-(1,2,3,'Python',5,(2,'a'))
print(s)
```

Результат:

```
Cf, 'u\ 'p', '1', 'e')
(1, 2, 3, 'Python', 5, (2, 'a'))
```

Операции над кортежами. Методы кортежей. Операции над кортежами и методы кортежей аналогичны средствам работы со строками при условии, что они не изменяют их содержимое. К ним можно отнести конкатенацию кортежей, повторение и ряд других.

Лабораторная работа № 9
Функции и процедуры

Цель работы: изучение процедур и функций в Python.

- синтаксис процедур и функций, процедура с параметром, локальные и глобальные переменные;
- применять синтаксис процедур и функций при составлении программы;
- основными навыками работы с функциями и процедурами.

Подпрограмма – это именованный фрагмент программы, к которому можно обратиться другого места программы

Подпрограммы делятся на две категории: процедуры и функции.

Процедуры.

Рассмотрим синтаксис процедуры:

def имя процедуры(Список параметров):

Система команд

Для определения процедуры используется ключевое слово def, затем указывается имя процедуры и в скобках её формальные параметры, если они присутствуют. После ставится двоеточие следующей строки с отступом в 4 пробела указываются команды.

Процедура — вспомогательный алгоритм, выполняющий некоторые действия.

Процедура должна быть определена к моменту её вызова. Определение процедуры начинается со служебного слова def.

Вызов процедуры осуществляется по ее имени, за которым следуют круглые скобки, например Err().

В одной программе может быть сколько угодно много вызовов одной и той же процедур.

Использование процедур сокращает код и повышает удобочитаемость.

Процедура с параметрами.

Как используются в Python параметры процедуры, рассмотрим на примере.

Пример.

Написать процедуру, которая печатает раз указанный символ (введенный с клавиатуры) с новой строки.

Def printChar(s):

print (s)
sim = input('введите символ')
printChar(sim) # первый вызов, вывод введенного символа
printChar('*') # второй вызов, вывод *

```
def printChar(s):
    print (s)
sim = input('введите символ: ')
printChar(sim) # первый вызов, вывод введенного символа
printChar('*') # второй вызов, вывод *
>>>
введите символ: 41
41
*
```

Глобальная переменная — если ей присвоено значение в основной программе (вне процедуры)

Локальная переменная (внутренняя) известна только на уровне процедуры, обратиться с основной программы и из других процедур нельзя.

Параметры процедуры — локальные переменные.

2. Примеры использования локальных и глобальных переменных.

Пример 1.

X = 3 # глобальная переменная
def pr(): # процедура без параметров
print (x) # вывод значения глобальной переменной
pr()

```
x = 3 # глобальная переменная
def pr(): # процедура без параметров
  print (x) # вывод значения глобальной переменной
pr()
>>> ==========
>>>
3
>>>
```

Пример 3. Создайте кортеж в диапазоне (0, 20) с шагом 1. Свяжите его с переменной. Выведите эту переменную на экран.

Решение:

```
A = tuple(range(0, 20, 1))
print(A)
```

Вывод программы:

```
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19)
```

Задания для самостоятельной работы:

a) Создайте два кортежа: один из чисел в диапазоне (1, количество учеников в группе) с шагом 1, второй _ из фамилий учеников вашей группы. Пусть они соответствуют друг другу;

b) Посмотрите, какая фамилия у студента с номером 5.

c) А теперь посмотрите, что записано во второй кортеж под номером 5.

d) Объедините два кортежа в один, присвоив результат новой переменной. Выведите получившийся список на экран.

e) Возьмите срез из соединенного кортежа так, чтобы туда попали некоторые части обоих первых кортежей. Срез свяжите с очередной новой переменной. Выведите значение этой переменной.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лабораторная работа № 8
Словари

Цель: ознакомиться с словарями

Словарь, или Dictionary — отображение между ключами (keys) и значениями (values), при котором ключу однозначно соответствует значение. В других языках программирования словари могут называться другими именами: хэши, ассоциативные массивы. В словаре не может быть двух одинаковых ключей, тогда как на значения таких ограничений не накладывается.

Его синтаксис следующий:

```
словарь:      (ключ1:     значение1, ключ2:
   значение2, . . .,ключN: значениеN, }
```

Словарь чаще всего создается с помощью литерала или функции diet (). Конструкция аналогична конструкции при создании строк, списков или кортежей.

Пример 1. Создание словаря.

```
D= {1:'Google',2:'Mail', 3:'Yandex', 4:'Yahoo'}
print(d [1])
dl=dict(g='Google', m='Mail', y='Yandex',
yl='Yahoo')
print(dl ['y'])
d2=dict(((1,'Google'), (2,'Mail'), (3, 'Yandex'),
(4, 'Yahoo')])
print(d2 [2])
d3=*dict. F roi^keys ([ 'Google', 'Mail',
'Yandex', 'Yahoo' ], [1,2,3,4])
print(d2 [4])
```

Результат:

```
Google
Yandex
Mail
Yahoo
```

Как видно из примера, задать словарь с помощью diet () можно двумя способами, а также это можно сделать посредством метода fromkeysO . Доступ к элементу словаря осуществляется путем указания после имени словаря в квадратных скобках ключа эле* мента, что аналогично индексу для элемента списка или кортежа.

Пример 2. Создайте словарь, который будет содержать значения параметров функции y = Acos(ωt+f). А затем по ключу запросите значения каждого из них.

Решение:

```
print ('y = A cos (wt+f)')
Parameters = {'A':10 , 'w':1, 'f':0}
Key = str(input('Какой параметр? '))
print( Parameters [Key ])
```

Вывод программы:

```
y = A cos(wt+f)
Какой параметр? A
10
```

Задания для самостоятельной работы:

1. Создайте словарь, связав его с переменной School, и наполните его данными, которые бы отражали количество учащихся в пяти разных классах (например, 1а, 1б, 2в и т. д.); выведите содержимое словаря на экран. Узнайте сколько человек в каком-нибудь классе. Класс запрашивается у пользователя с клавиатуры, если такого запрашиваемого класса в школе нет, то выдаётся сообщение: «Такого класса на существует».

2. Создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), полученный объект dict_items передайте в написанную вами функцию, которая создает и возвращает новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

3. Создайте словарь, связав его с переменной School, и наполните его данными, которые бы отражали количество учащихся в пяти разных классах (например, 1а, 1б, 2в и т. д.). В школе произошли изменения, внесите их в словарь: в трёх классах изменилось количество учащихся; результат выведите на экран.

4.Дан словарь с однозначным соответствием английских и русских слов в таком формате:

cat - кошка
dog - собака
mouse - мышь
house - дом
eats - ест
in - в
too - тоже

Дан текст для перевода, например:

Mouse in house. Cat in house.
Cat eats mouse in dog house.
Dog eats mouse too.

Требуется сделать подстрочный перевод с помощью имеющегося словаря. Незнакомые словарю слова нужно оставлять в исходном виде.

5. Создайте словарь, связав его с переменной School, и наполните его данными, которые бы отражали количество учащихся в пяти разных классах (например, 1а, 1б, 2в и т. д.). В школе появилось два новых класса, новый словарь выведите на экран.

6. Создайте словарь, связав его с переменной School, и наполните его данными, которые бы отражали количество учащихся в пяти разных классах (например, 1а, 1б, 2в и т. д.). В школе расформировали один из классов, выведите содержимое нового словаря на экран.

7. Дан текст на некотором языке. Требуется подсчитать сколько раз каждое слово входит в этот текст и вывести десять самых часто употребляемых слов в этом тексте и количество их употреблений.

Используйте словарь, в котором ключ - слово, а значение - количество таких слов.

8. Создайте словарь, связав его с переменной School, и наполните его данными, которые бы отражали количество учащихся в пяти разных классах (например, 1а, 1б, 2в и т. д.). Вычислите общее количество учащихся в школе.

9. Дан список стран и языков на которых говорят в этой стране. На ввод задается N - длина списка и список языков. Для каждого языка укажите, в каких странах на нем говорят.

10. Имеется англо-русский словарь в таком формате:

cat - кошка
dog - собака
home - домашняя папка, дом
mouse - мышь, манипулятор мышь
to do - делать, изготавливать
to make - изготавливать

Требуется создать русско-английский словарь и вывести его в формате:

делать - to do
дом - home
домашняя папка - home
изготавливать - to do, to make
кошка - cat
манипулятор мышь - mouse
мышь - mouse
собака - dog

Порядок строк в выходном файле должен быть словарным с человеческой точки зрения (так называемый лексикографический порядок слов). То есть выходные строки нужно отсортировать.

11. Дана строка. Подсчитать, сколько раз в строке встречается каждый символ. Результат вывести в формате: для строки 'how many times'

{1: ['a', 'e', 'i', 'h', 'o', 'n', 's', 't', 'w', 'y'], 2: [' ', 'm']}

12. Создайте словарь, содержащий пары ‘игрушка’ – ‘ограничение по возрасту’. Отсортируйте словарь по ключам.

13. Создайте англо-русский словарь. Инвертируйте его, т.е. на его основе создайте русско-английский словарь.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лабораторная работа № 9
Функции и процедуры

Цель работы: изучение процедур и функций в Python.

- синтаксис процедур и функций, процедура с параметром, локальные и глобальные переменные;
- применять синтаксис процедур и функций при составлении программы;
- основными навыками работы с функциями и процедурами.

Подпрограмма – это именованный фрагмент программы, к которому можно обратиться другого места программы

Подпрограммы делятся на две категории: процедуры и функции.

Процедуры.

Рассмотрим синтаксис процедуры:

def имя процедуры(Список параметров):

Система команд

Для определения процедуры используется ключевое слово def, затем указывается имя процедуры и в скобках её формальные параметры, если они присутствуют. После ставится двоеточие следующей строки с отступом в 4 пробела указываются команды.

Процедура — вспомогательный алгоритм, выполняющий некоторые действия.

Процедура должна быть определена к моменту её вызова. Определение процедуры начинается со служебного слова def.

Вызов процедуры осуществляется по ее имени, за которым следуют круглые скобки, например Err().

В одной программе может быть сколько угодно много вызовов одной и той же процедур.

Использование процедур сокращает код и повышает удобочитаемость.

Процедура с параметрами.

Как используются в Python параметры процедуры, рассмотрим на примере.

Пример.

Написать процедуру, которая печатает раз указанный символ (введенный с клавиатуры) с новой строки.

Def printChar(s):

print (s)
sim = input('введите символ')
printChar(sim) # первый вызов, вывод введенного символа
printChar('*') # второй вызов, вывод *

```
def printChar(s):
    print (s)
sim = input('введите символ: ')
printChar(sim) # первый вызов, вывод введенного символа
printChar('*') # второй вызов, вывод *
>>>
введите символ: 41
41
*
```

Глобальная переменная — если ей присвоено значение в основной программе (вне процедуры)

Локальная переменная (внутренняя) известна только на уровне процедуры, обратиться с основной программы и из других процедур нельзя.

Параметры процедуры — локальные переменные.

2. Примеры использования локальных и глобальных переменных.

Пример 1.

X = 3 # глобальная переменная
def pr(): # процедура без параметров
print (x) # вывод значения глобальной переменной
pr()

```
x = 3 # глобальная переменная
def pr(): # процедура без параметров
  print (x) # вывод значения глобальной переменной
pr()
>>> ==========
>>>
3
>>>
```

Пример 2.

X = 3 # глобальная переменная
def pr(a): # процедура с параметром
print (a) # 4
pr(x) # передача параметра глобальной переменной (3)

```
File  Edit  Format  Run  Options  Windows  Help
x = 10  # глобальная переменная
def pr(a): # процедура с параметром
  print (a) # 4
pr(x) # передача параметра глобальной переменной (3)
>>>
10
>>>
```

Существует возможность изменить значение глобальной переменной (не создавая локальную) процедуре с помощью слова global:

x = 3 # глобальная переменная
def pr(): # процедура без параметров
global x
x = pow(x,10)
print (x) # вывод измененного значения глобальной переменной
pr()

```
x=3 # глобальная переменная
print('Начальное значение: ',x)
def pr(): # процедура без параметров
    global x
    x=pow(x,10)
    print ('Изменённое значение: ',x)
pr()
>>>
Начальное значение:  3
Изменённое значение:  59049
>>>
```

Функции.

Функция – подпрограмма, к которому можно обратиться из другого места программы.

Для создания функции используется ключевое слово def, после которого указывается им список аргументов в круглых скобках. Тело функции выделяется также как тело условия цикла: четырьмя пробелами.

Рассмотрим синтаксис функции:

def имя функции(Список параметров):

Система команд

return выражение

Часть функций языка Python являются встроенными функциями, которые обеспечены синтаксисом самого языка. Например, int, input, randint.

Рассмотрим пример создания пользовательских функций.

Пример 1.

Вычислить сумму цифр числа.

У у ц фр

def sumD(n): # определение функции с параметром

sumD = 0

while n!= 0:

sumD += n % 10

n = n // 10

return sumD # возврат значения функции

основная программа

print (sumD(int(input())) # вызов функции с параметром

```
def sumD(n): # определение функции с параметром
    summa = 0
    while n!= 0:
        summa += n % 10
        n=n//10
    return summa # возврат значения функции
# основная программа
print (sumD(int(input()))) # вызов функции с параметром
>>>
123456789
45
>>>
```

Пример.

Определить, являются ли три треугольника равновеликими. Длины сторон вводить с клавиатуры. Для подсчёта площади треугольника использовать формулу Герона. Вычислить площади оформить в виде функции с тремя параметрами.

Формула Герона:

Решение:

```
import math
def s(x,y,z):
    p=(x+y+z)/2
    s=math.sqrt(p*(p-x)*(p-y)*(p-z))
    return s
A=[]
for i in range(3):
    print('Введите стороны ',i,'-го треугольника:')
    a=int(input('a:'))
    b=int(input('b:'))
    c=int(input('c:'))
    A.append(s(a,b,c))
for i in range(3):
    print('Площадь ',i,'-го треугольника {:.2f}'.format(A[i]))
if A[0]==A[1]:
    if A[0]==A[2]:
        print('Треугольники равновеликие')
else: print('Треугольники не равновеликие')
```

```
Введите стороны  0 -го треугольника:
a:3
b:4
c:5
Введите стороны  1 -го треугольника:
a:6
b:7
c:8
Введите стороны  2 -го треугольника:
a:9
b:10
c:11
Площадь  0 -го треугольника 6.00
Площадь  1 -го треугольника 20.33
Площадь  2 -го треугольника 42.43
Треугольники не равновеликие
```

Ввести одномерный массив A длиной m. Поменять в нём местами первый и последний элементы. Длину массива и его элементы ввести с клавиатуры. В программе описать процедуру для замены элементов массива. Вывести исходные и полученные массивы.

Решение:

```
def zam(X):
    tmp=X[0]
    X[0]=X[len(X)-1]
    X[len(X)-1]=tmp
A=[]
m=int(input('Введите длину массива:'))
for i in range(m):
    print('Введите ',i,'элемент массива')
    A.append(int(input()))
print(A)
zam(A)
print(A)
```

```
Введите длину массива:5
Введите  0 элемент массива
0
Введите  1 элемент массива
1
Введите  2 элемент массива
2
Введите  3 элемент массива
3
Введите  4 элемент массива
4
[0, 1, 2, 3, 4]
[4, 1, 2, 3, 0]
```

Задания для самостоятельной работы:

Вариант 1.

1. Составить программу для вычисления площади разных геометрических фигур.

2. Даны 3 различных массива целых чисел (размер каждого не превышает 15). В каждом найти сумму элементов и среднеарифметическое значение.

Вариант 2.

1. Вычислить площадь правильного шестиугольника со стороной a, используя подпрограмму вычисления площади треугольника.

2. Пользователь вводит две стороны трех прямоугольников. Вывести их площади.

Вариант 3.

1. Даны катеты двух прямоугольных треугольников. Написать функцию вычисления для гипотенузы этих треугольников. Сравнить и вывести какая из гипотенуз больше, а какая меньше.

2. Преобразовать строку так, чтобы буквы каждого слова в ней были отсортированы по возрастанию.

Вариант 4.

1. Даны две дроби A/B и C/D (A, B, C, D — натуральные числа). Составить программу деления на дробь. Ответ должен быть несократимой дробью. Использовать подпрограмму алгоритма Евклида для определения НОД.

2. Задана окружность (x-a)2 + (y-b)2 = R2 и точки P(p1, p2), F(f1, f1), L(l1,l2). Выяснить и вывести на экран, сколько точек лежит внутри окружности. Проверку, лежит ли точка внутри окружности, оформить в виде процедуры.

Вариант 5.

1. Даны две дроби A/B и C/D (A, B, C, D — натуральные числа). Составить программу вычисления первой дроби второй. Ответ должен быть несократимой дробью. Использовать подпрорамму алгоритма Евклида для определения НОД.

2. Напишите программу, которая выводит в одну строчку все делители переданного ей числа, разделяя их пробелами.

Вариант 6.

1. Составить программу нахождения наибольшего общего делителя (НОД) и наименьшее кратного (НОК) двух натуральных чисел НОК(A, B) = (A*B)/НОД(A,B). Использовать подпрограмму алгоритма Евклида для определения НОД.

2. Составить программу вычисления площади выпуклого четырехугольника, заданного с четырех сторон и диагонали.

Вариант 7.

1. Даны числа X, Y, Z, T — длины сторон четырехугольника. Вычислить его площадь, если между сторонами длиной X и У —

прямой. Использовать две подпрограммы для вычисления площадей: прямоугольного треугольника и прямоугольника.

2. Напишите программу, которая переводит переданное ей неотрицательное целое число в одзначный восьмеричный код, сохранив лидирующие нули.

Вариант 8.

1. Найти все натуральные числа, не превосходящие заданного n, которые делятся на кА своих цифр.

2. Ввести одномерный массив А длиной m. Поменять в нём местами первый и последний элементы. Длину массива и его элементы ввести с клавиатуры. В программе описать процедуру для замены элементов массива. Вывести исходные и полученные массивы.

Вариант 9.

1. Из заданного числа вычли сумму его цифр. Из результата вновь вычли сумму его цифр. Через сколько таких действий получится нуль?

2. Даны 3 различных массива целых чисел. В каждом массиве найти произведение элемент среднеарифметическое значение.

Вариант 10.

1. На отрезке [100, N] (210 < N < 231) найти количество чисел, составленных из цифр a, b,c.

2. Составить программу, которая изменяет последовательность слов в строке на обратную.

Вариант 11.

1. Два простых числа называются «близнецами», если они отличаются друг от друга на 2.(например, 41 и 43). Напечатать все пары «близнецов» из отрезка [n, 2n], где n — заданное натуральное число, большее 2..

2. Даны этих матриц. Нахождение максимального элемента матрицы оформить в виде процедуры.

Вариант 12.

1. Два натуральных числа называются «дружественными», если каждое из них равно сумме делителей (кроме его самого) другого (например, числа 220 и 284). Найти все пары «дружественных» чисел, которые не больше данного числа N.

2. Даны длины сторон треугольника a, b, c. Найти медианы треугольника, сторонами которого являются медианы исходного треугольника. Для вычисления медианы проведенной к с тороне,

использовать формулу. Вычисление медианы оформить в виде процедуры.

Вариант 13.

1. Натуральное число, в записи которого n цифр, называется числом Армстронга, если сумма цифр, возведенная в степень n, равна самому числу. Найти все числа Армстронга от 1 до 100.

2. Три точки заданы своими координатами X(x1, x2), Y(y1, y2) и Z(z1, z2). Найти и напечатать координаты точки, для которой угол между осью абсцисс и лучом, соединяющим начало координат с точкой, минимальный. Вычисления оформить в виде процедуры.

Вариант 14.

1. Составить программу для нахождения чисел из интервала [M, N], имеющих наибольшее количество делителей.

2.Четыре точки заданы своими координатами X(x1, x2), Y(y1, y2), Z(z1, z2), P(p1, p2). Выяснить, какие из них находятся на максимальном расстоянии друг от друга и вывести на экран их этого расстояния. Вычисление расстояния между двумя точками оформить в виде процедуры.

Вариант 15.

1. Найти все простые натуральные числа, не превосходящие n, двоичная запись которые представляет собой палиндром, т. Е. читается одинаково слева направо и справа налево.

2. Четыре точки заданы своими координатами X(x1, x2, x3), Y(y1, y2, y3), Z(z1, z2, z3), T(t1,

Выяснить, какие из них находятся на минимальном расстоянии друг от друга и вывести значение этого расстояния. Вычисление расстояния между двумя точками оформить в процедуры.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лабораторная работа № 10
Обработка одномерного массива

Цель: Освоение методов работы с одномерными массивами
Ключевые слова: Структура индекса массива.

Повторение теоретического материала: Дан одномерный массив численных значений, нумеруя N элементы. Найдите среднее число элементов в массиве.

Дано:

N – число элементов в массиве;

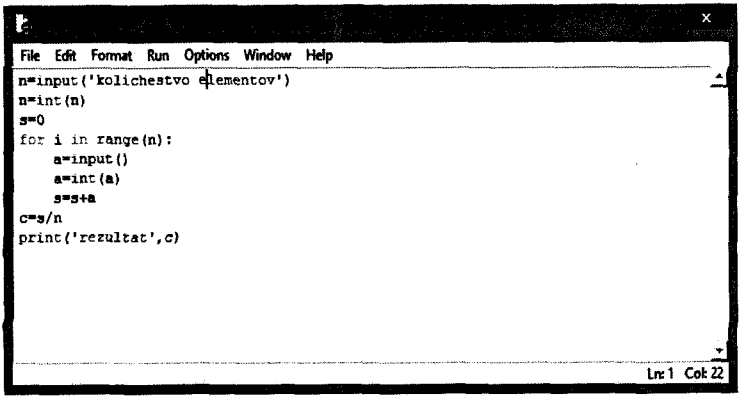
i – индекс массива (параметр цикла). A [i] – элемент массива;

Найти:

S – сумму элементов массива

C – среднее арифметическое элементов массива, C = S / N

N = input ('Number elements: ')
N = int (N)
S = 0
for i in range (N):
a = input ()
a = int (a)
S = S + a
C = S / N
print ('Result:', C)



Пример 2.

X = 3 # глобальная переменная
def pr(a): # процедура с параметром
print (a) # 4
pr(x) # передача параметра глобальной переменной (3)

```
File  Edit  Format  Run  Options  Windows  Help
x = 10  # глобальная переменная
def pr(a): # процедура с параметром
  print (a) # 4
pr(x) # передача параметра глобальной переменной (3)
>>>
10
>>>
```

Существует возможность изменить значение глобальной переменной (не создавая локальную) процедуре с помощью слова global:

x = 3 # глобальная переменная
def pr(): # процедура без параметров
global x
x = pow(x,10)
print (x) # вывод измененного значения глобальной переменной
pr()

```
x=3 # глобальная переменная
print('Начальное значение: ',x)
def pr(): # процедура без параметров
    global x
    x=pow(x,10)
    print ('Изменённое значение: ',x)
pr()
>>>
Начальное значение:  3
Изменённое значение:  59049
>>>
```

Функции.

Функция – подпрограмма, к которому можно обратиться из другого места программы.

Для создания функции используется ключевое слово def, после которого указывается им список аргументов в круглых скобках. Тело функции выделяется также как тело условия цикла: четырьмя пробелами.

Рассмотрим синтаксис функции:

def имя функции(Список параметров):

Система команд

return выражение

Часть функций языка Python являются встроенными функциями, которые обеспечены синтаксисом самого языка. Например, int, input, randint.

Рассмотрим пример создания пользовательских функций.

Пример 1.

Вычислить сумму цифр числа.

У у ц фр

def sumD(n): # определение функции с параметром

sumD = 0

while n!= 0:

sumD += n % 10

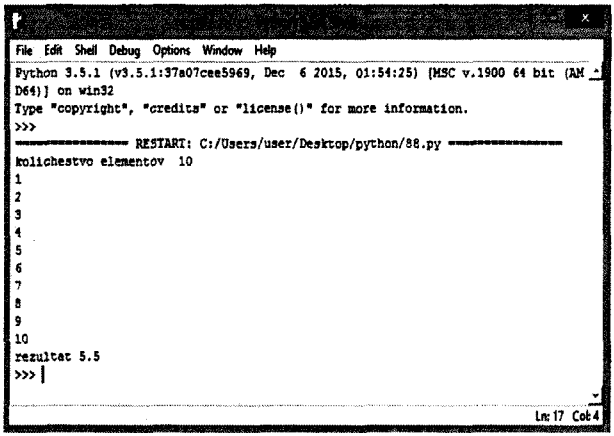
n = n // 10

return sumD # возврат значения функции

основная программа

print (sumD(int(input())) # вызов функции с параметром

```
def sumD(n): # определение функции с параметром
    summa = 0
    while n!= 0:
        summa += n % 10
        n=n//10
    return summa # возврат значения функции
# основная программа
print (sumD(int(input()))) # вызов функции с параметром
>>>
123456789
45
>>>
```



Задания для самостоятельной работы:

1. Обработать на компьютере массив в соответствии с вариантом задания.

1. X(100) вычислить сумму и количество элементов массива X

2. A(80) вычислить среднее арифметическое значение элемента массива A.

3. X(70) переписать элементы массива X в массив Y и подсчитать их количество.

4. B(50) определить максимальный элемент массива B и его порядковый номер.

5. C(40) вычислить минимальный, элемент массива C и его номер.

6. D(80) найти максимальный и минимальный элементы массива d и поменять их местами.

7. Y(20) вычислить среднее геометрическое элемента массива У.

8. Z(30) расположить в массиве r сначала положительные, а затем отрицательные элементы массива z.

9. N(50) определить сумму элементов массива n, кратных трем

10. X(n) вычислить сумму и количество элементов массива X.

11.A(n) найти среднее геометрическое элементов массива A.

12. X(n) переписать в массив Y подряд положительные элементы массива X.

13. X(n) переписать подряд в массив Y положительные и в массив z отрицательные элементы массива X.

14. B(k) определить максимальный элемент массива B и его порядковый номер.

15. C(k) определить минимальный элемент массива C и его порядковый номер.

2. Проверить правильность выполнения программы с помощью тестового варианта.

Литература:

1. Гвидо ванн Россум и др. Язык программирования Python. / 2001 — 454 с.

2. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

3. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

5. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лабораторная работа № 11
Обработка матриц

Цель: Освоение методов работы с двухмерными массивами
Ключевые слова: Строка, столбец, индекс, матрица.

Повторение теоретического материала: Двухмерные массивы называются, прямоугольная (табличная) структура. Массивы описывает таким же образом, как одномерные. Различие заключается в том, что объект имеет две координаты (два индекса), номер строки и номер столбца, который является элементом.

Пример 1.
Дан двумерный массив размером 3x3. Определить максимальное значение среди элемент третьего столбца массива; максимальное значение среди элементов второй строки массива полученные значения.
Решение:

```
n=3
a=[]
for i in range(n):
    b = []
    for j in range(n):
        print('Введите [',i,',',j,'] элемент']
        b.append(int(input()))
    a.append(b)
#вывод массива
for i in range(n):
    for j in range(n):
        print(a[i][j], end=' ')
    print()

#максимальное значение среди элементов третьего столбца
maximum=a[0][2]
for i in range(n):
    for j in range(n):
        if maximum<a[i][2]:
            maximum=a[i][2]
print('Максимальный в 3 столбце:',maximum)

#максимальное значение среди элементов второй строки
maximum=a[1][0]
for i in range(n):
    for j in range(n):
        if maximum<a[1][j]:
            maximum=a[1][j]
print('Максимальный во второй строке:',maximum)
```

```
Введите [ 0 , 0 ] элемент
1
Введите [ 0 , 1 ] элемент
2
Введите [ 0 , 2 ] элемент
3
Введите [ 1 , 0 ] элемент
4
Введите [ 1 , 1 ] элемент
5
Введите [ 1 , 2 ] элемент
6
Введите [ 2 , 0 ] элемент
7
Введите [ 2 , 1 ] элемент
8
Введите [ 2 , 2 ] элемент
9
1 2 3
4 5 6
7 8 9
Максимальный в 3 столбце: 9
Максимальный во второй строке: 6
```

Пример 2.

Дан двумерный массив размером mxn. Сформировать новый массив заменив положительные элементы единицами, а отрицательные нулями. Вывести оба массива.

Решение:

```
m=int(input('Введите количество строк'))
n=int(input('Введите количество столбцов'))
a=[]
for i in range(m):
    b = []
    for j in range(n):
        print('Введите [',i,',',j,'] элемент')
        b.append(int(input()))
    a.append(b)
#вывод массива
print('Исходный массив:')
for i in range(m):
    for j in range(n):
        print(a[i][j], end=' ')
    print()

for i in range(m):
    for j in range(n):
        if a[i][j]<0: a[i][j]=0
        elif a[i][j]>0: a[i][j]=1

#вывод массива
print('Измененный массив:')
for i in range(m):
    for j in range(n):
        print(a[i][j], end=' ')
    print()
```

```
Введите количество строк:3
Введите количество столбцов:4
Введите [ 0 , 0 ] элемент
-1
Введите [ 0 , 1 ] элемент
5
Введите [ 0 , 2 ] элемент
4
Введите [ 0 , 3 ] элемент
-5
Введите [ 1 , 0 ] элемент
-2
Введите [ 1 , 1 ] элемент
-1
Введите [ 1 , 2 ] элемент
0
Введите [ 1 , 3 ] элемент
4
Введите [ 2 , 0 ] элемент
-5
Введите [ 2 , 1 ] элемент
4
Введите [ 2 , 2 ] элемент
5
Введите [ 2 , 3 ] элемент
-5
Исходный массив:
-1 5 4 -5
-2 -1 0 4
-5 4 5 -5
Полученный массив:
0 1 1 0
0 0 0 1
0 1 1 0
```

Задания для самостоятельной работы:

Вариант 1.

1. Вычислить сумму и число положительных элементов матрицы A[N, N], находящихся над диагональю.

2. Дана матрица B[N, M]. Найти в каждой строке матрицы максимальный и минимальный элемент и поменять их с первым и последним элементами строки соответственно.

Вариант 2.

1. Дана целая квадратная матрица n-го порядка. Определить, является ли она магическим квадратом, т. Е. такой матрицей, в которой суммы элементов во всех строках и столбцах одинаковы.

2. Дана прямоугольная матрица A[N, N]. Переставить первый и последний столбцы местами и вывести на экран.

Вариант 3.

1. Определить, является ли заданная целая квадратная матрица n-го порядка симметричной (относительно главной диагонали).

2. Дана вещественная матрица размером n x m. Переставляя ее строки и столбцы, добиться тог, чтобы наибольший элемент (или один из них) оказался в верхнем левом углу.

Вариант 4.

1. Дана прямоугольная матрица. Найти строку с наибольшей и строку с наименьшей суммой элементов. Вывести на печать найденные строки и суммы их элементов.

2. Дана квадратная матрица A[N, N], Записать на место отрицательных элементов матрицы нули, а на место положительных — единицы. Вывести на печать нижнюю треугольную матрицу в общепринятом виде.

Вариант 5.

1. Упорядочить по возрастанию элементы каждой строки матрицы размером n x m.

2. Дана действительная матрица размером n x m, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением. Если число четное, то заменяется нулем, нет, -единицей. Вывести на экран новую матрицу.

Вариант 6.

1. Дана целочисленная квадратная матрица. Найти в каждой строке наибольший элемент и в каждом столбце наименьший. Вывести на экран.

2. Дана действительная квадратная матрица порядка N (N — нечетное), все элементы которого различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.

Вариант 7.

1. Квадратная матрица, симметричная относительно главной диагонали, задана верхним треугольником в виде одномерного массива. Восстановить исходную матрицу и напечатать строкам.

2. Для заданной квадратной матрицы сформировать одномерный массив из ее диагональных элементов. Найти след матрицы, просуммировав элементы одномерного массива. Преобразовать исходную матрицу по правилу: четные строки

разделить на полученное значение, нечетные оставить без изменения.

Вариант 8.

1. Задана матрица порядка n и число k. Разделить элементы k-й строки на диагональный элемент расположенный в этой строке.

2. Задана квадратная матрица. Получить транспонированную матрицу (перевернутую относительно главной диагонали) и вывести на экран.

Вариант 9.

1. Для целочисленной квадратной матрицы найти число элементов, кратных k, и наибольшее число элементов.

2. В данной действительной квадратной матрице порядка n найти наибольший по модулю чмсло. Получить квадратную матрицу порядка n — 1 путем отбрасывания из исходной матрицы строки и столбца, на пересечении которых расположен элемент с найденным значением.

Вариант 10.

1. Найти максимальный среди всех элементов тех строк заданной матрицы, которые упорядочены, (либо по возрастанию, либо по убыванию).

2. Расположить столбцы матрицы D[M, N] в порядке возрастания элементов k-й строки (1 <= k)

Вариант 11.

1. В данной действительной квадратной матрице порядка *n* найти сумму элементов строки, в котором расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.

2. *Среди столбцов заданной целочисленной матрицы,* содержащих только такие элементы, которые по модулю не больше 10, найти столбец с минимальным произведением элементов и поменять его с соседним.

Вариант 12.

1. Для заданной квадратной матрицы найти такие k, что k-я строка матрицы совпадает с k-м столбцом.

2. Дана действительная матрица размером n x m. Требуется преобразовать матрицу: поэлементно вычесть последнюю строку из всех строк, кроме последней.

Вариант 13.

1. Определить наименьший элемент каждой четной строки матрицы A[M, N].

2. Найти наибольший и наименьший элементы прямоугольной матрицы и поменять их местами.

Вариант 14.

1. Задана квадратная матрица. Переставить строку с максимальным элементом на главной диагонали со строкой с заданным номером m.

2. Составить программу, которая заполняет квадратную матрицу порядка *n* натуральными числами 1,2, 3, …, n2, записывая их в нее «по спирали».

Например, для n = 5 получаем следующую матрицу:

1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
14 12 11 10 9

Вариант 15.

1. Определить номера строк матрицы R[M, N], хотя бы один элемент которых равен c, и элемент строки умножить на d.

2. Среди тех строк целочисленной матрицы, которые содержат только нечетные элементы, найти строку с максимальной суммой модулей элементов.

Литература:

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017.— 624с.: ил.—(Мировой компьютерный бестселлер).

2. Гвидо анн Россум и др. Язык программирования Python. / 2001 — 454 с.

3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. Тит. Англ.

5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. Вкл]: ил. — (Библиотека ALT).

Лабораторная работа № 12
Построение графиков

Цель: Построение графиков в Python

Построение графиков _ один из главных этапов обработ данных. Все современные компьютерные програм предоставляющие функцию построения графиков, условно мож разделить на две категории: программы с визуальным интерфейсом, где построение и редактирование графика осуществляется средствами разного рода меню, полей ввода, лист-боксов, чек боксов и других виджетов, и программы, где для построения графика необходимо писать команды, объединяемые в тек называемы скрипты. К первой категории относятся, например, Origin, MS Excel, OpenOffice/LibreOffice Calc, Statistica, Grapher, ко второй _ gnuplot, многие математические пакеты, например, MATLAB и SciLab и различные библиотеки вроде PGPlot и PLPlot имеющие поддержку во многих языках программирования.

Основное преимущество скриптового способа построени графика в том, что вы можете встроить его без проблем в вашу программу, производящую вычисления. Кроме того, скрипты позволяют легко перестраивать графики с новыми данными, автоматизировать построение графиков, дают почти неограниченный контроль над точностью позиционирования и размером деталей.

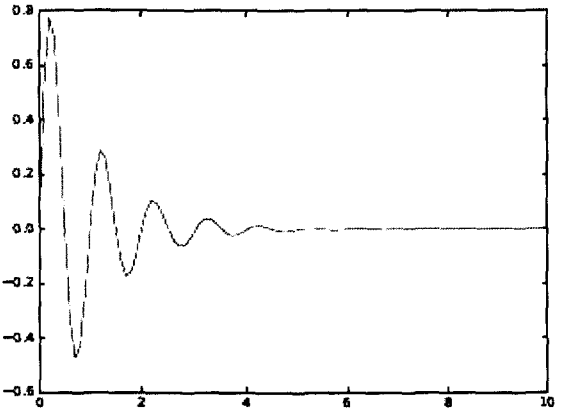
Модуль matplotlib _ специализированная библиотека для языка Python. Хотя основное её преимущество в простоте и быстроте использования, она позволяет делать графики очень высокого типографского качества. Модуль matplotlib базируется на возможностях numpy, установка которого обязательна для функционирования matplotlib.

Пример задачи 1. (Затухающая синусоида, вариант 1). Постройте график затухающей синусоиды $e^{-x}\sin(2\pi x)$ на отрезке [0; 10], используя шаг по абсциссе, равный 0.1.

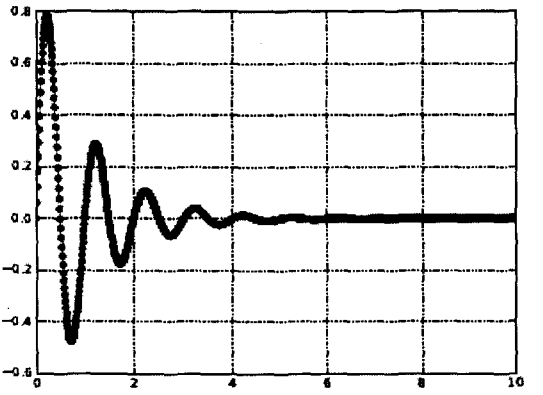
Решение задачи 1.

```
From numpy import *
from matplotlib . pyplot import *
```

```
x = arange (0, 10, 0.1)
f = exp (-x)* sin (2* pi*x)
plot (x, f)
show ()
```



(a)



(b)

Рис. 1. Иллюстрация к задачам 1 – (a) и 2 – (b).

Пример задачи 2. (Затухающая синусоида, вариант 2) Для построенного в предыдущем задании графика измените: цвет линии, тип линии и маркеров, шаг выборки данных, введите сетку и сохраните полученный график в файл.

Решение задачи 2

```
from numpy import *
from matplotlib . pyplot import *
x = arange (0, 10, 0.01)
f = exp (-x)* sin (2* pi*x)
plot (x, f, '-o', color ='black ')
grid ( True )
savefig ('plot .png ')
show ()
```

Пример задачи 3. (Семейство затухающих синусоид) Постройте семейство функций e^{-x} $\sin(2\pi x+\phi 0)$ на одном графике различными цветами при $\phi 0 = 0$, $\phi 0 = \pi/6$ и $\phi 0 = \pi/3$. Сделайте для графика легенду.

Решение задачи 3.

```
From numpy import *
from matplotlib . pyplot import *
x = arange (0, 10, 0.01)
f1 = exp (-x)* sin (2* pi*x)
f2 = exp (-x)* sin (2* pi*x+pi /6)
f3 = exp (-x)* sin (2* pi*x+pi /3)
plot (x, f1 , label =r'$\sin (2\ piЪx)$')
plot (x, f2 , label =r'$\sin (2\
piЪxЪ+Ъ\piЪ/6)$')
plot (x, f3 , label =r'$\sin (2\
piЪxЪ+Ъ\piЪ/3)$')
legend (loc='best ')
grid ( True )
show ()
```

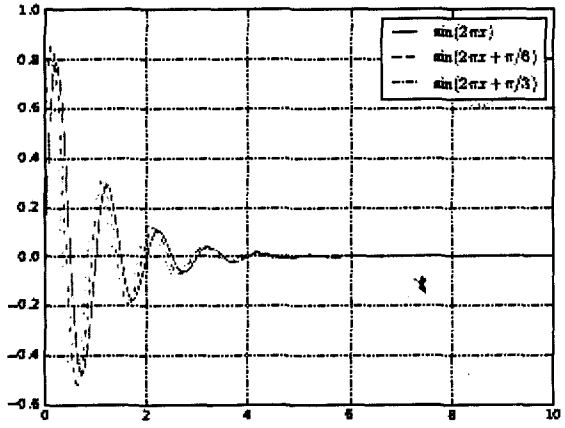
Пример задачи 4. (Семейство графиков с затухающими синусоидами)

Перестройте графики так, чтобы каждая кривая располагалась на одном графике с помощью команды subplot, легенду уберите, а её текст переместите в название соответствующего графика. Графики расположите на полотне в один столбец.

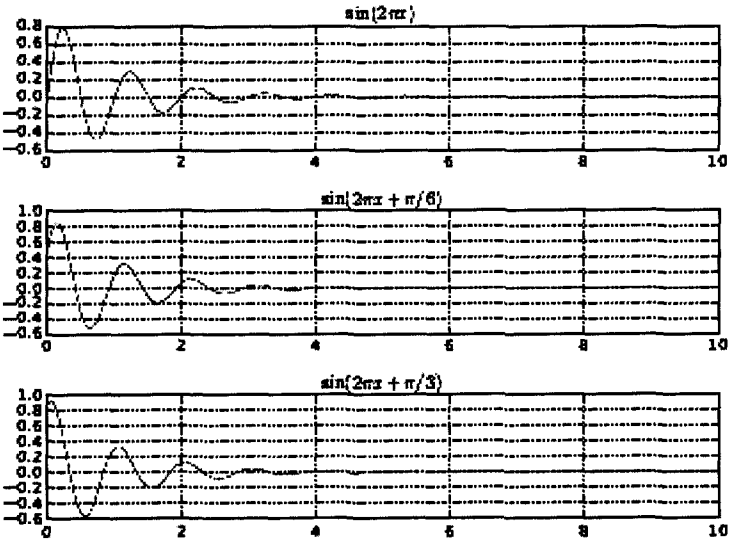
Решение задачи 4.

```
From numpy import *
from matplotlib . pyplot import *
x = arange (0, 10, 0.01)
f1 = exp (-x)* sin (2* pi*x)
f2 = exp (-x)* sin (2* pi*x+pi /6)
f3 = exp (-x)* sin (2* pi*x+pi /3)
subplot (3, 1, 1)
plot (x, f1)
title (r'$\sin (2\ pi x)$')
grid ( True )
subplot (3, 1, 2)
plot (x, f2)
title (r'$\sin (2\ pi x + \pi /6) $')
grid ( True )
subplot (3, 1, 3)
plot (x, f3)
title (r'$\sin (2\ piЂxЂ+Ђ\piЂ/3) $')
grid ( True )
tight_layout ()
show ()
```

Пример задачи 5. Постройте закрашенную контурную диаграмму и трёхмерный график для функции двух переменных $z = \frac{2xy}{x^2 + y^2}$, определённой в прямоугольной области (x ∈ [−3; 3], y ∈ [−3; 3]):



(a)



(b)

Рис. 2. Иллюстрация к задачам 3 (a) и 4 (b).

Решение задачи 5.

```
From numpy import *
from matplotlib . pyplot import *
from matplotlib . mlab import *
from mpl_toolkits . mplot3d import Axes3D
x = arange (-3, 3, 0.01)
y = arange (-3, 3, 0.01)
X, Y = meshgrid (x, y)
Z = 2*X*Y/(X **2+ Y **2)
contourf (X, Y, Z)
savefig ('plot .png ')
fig = figure ()
ax = Axes3D (fig )
ax. Plot_surface (X, Y, Z)
savefig ('3D.png ')
show ()
```

Задания для самостоятельной работы:

Для каждого из заданий данного раздела следует выполнить 1 вариант с номером (n − 1)%m + 1, где n _ номер в списке группы, а m _ число заданий.

Задание 6. Постройте графики следующих функций, используя шаг выборки данных по абсциссе из **примера 1**:

1. x^2 на отрезке $x \in [-2; 2]$;
2. x^3 на отрезке $x \in [-2; 2]$;
3. x^4 на отрезке $x \in [-2; 2]$;
4. $\cos(2\pi t)$ на отрезке $t \in [-10; 10]$;
5. $\frac{1}{t}\cos(2\pi t)$ на отрезке $t \in [1; 10]$;
6. $e^{-t}\cos(2\pi t)$ на отрезке $t \in [-10; 10]$;
7. $4\sin(\pi t + \pi/8) - 1$ на отрезке $t \in [-10; 10]$;
8. $2\cos(t-2) + \sin(2 * t - 4)$ на отрезке $t \in [-20\pi; 10\pi]$;
9. $\ln(x+1)$ на отрезке $x \in [0; e-1]$;
10. $\log_2(|x|)$ на отрезке $x \in [-4; 4]$ за исключением точки $x = 0$;
11. 2^x на отрезке $x \in [-2; 2]$;
12. e^x на отрезке $x \in [-2; 2]$;
13. 2^{-x} на отрезке $x \in [-2; 2]$;
14. $\sqrt[3]{x}$ на отрезке $x \in [1; 125]$;
15. $\sqrt[5]{x}$ на отрезке $x \in [1; 32]$.|

Задание 7.Для построенного в рамках задания 6 графика измените:

- цвет линии;
- тип линии и маркеров;
- шаг выборки данных.

Далее введите сетку. Сохраните полученный график в файл, попробуйте сохранять файл в разных форматах: png, pdf, jpg, eps.

Задание 8. Постройте семейство функций на одном графике различными цветами:

1. степенные многочлены с целыми степенями от 1 до 6 на отрезке [−1; 1];

2. синусоиды y = sin(ωt) с частотами ω= 2π, ω = 3π, . . . , ω = 8π на отрезке t ∈ [−1; 1];

3. синусоиды $y = \sin(2\pi t + \phi 0)$ с начальными фазами $\phi 0 = 0$, $\phi 0 = \pi/6, \ldots, \phi 0 = 5\pi/6$ на отрезке $t \in [-1; 1]$;

4. логарифмические функции log2(x), ln(x) и $\log_{10}(x)$ на отрезке $x \in [1; 10]$;

5. гиперболические функции sh(x), ch(x) и th(x) на отрезке $x \in [-10; 10]$, для их вычисления воспользуйтесь их выражением через экспоненту.

Задание 9. Для построенного в задании 8 графика сделайте сетку и легенду. Перестройте графики так, чтобы каждая кривая располагалась на одном графике с помощью команды subplot, легенду уберите, а её текст переместите в название соответствующего графика. Графики расположите на полотне:

- в одни столбец;
- в два столбца;
- в 3 столбца;
- в одну строку.

Перестройте графики из задания каждый в своём окне. Сделайте так, чтобы эти графики автоматически сохранялись каждый в свой файл.

Задание 10. Постройте круговую диаграмму, которая показывала бы доли от общего числа студентов вашей группы, сдавших сессию на:

1. одни пятёрки,
2. пятёрки и четвёрки,
3. с тройками, но без задолжностей,
4. с задолжностями, сумевших в итоге пересдать,
5. несдавших и отчисленных (если такие имеются).

Задание 11. Постройте закрашенную контурную диаграмму и трёхмерный график для следующих функций двух переменных, определённых в прямоугольной области $x \in [-3; 3]$, $y \in [-3; 3]$:

1. $z = x^2 + y^2$,
2. $z = x^2 - y^2$,
3. $z = x^3 + y^3$,
4. $z = x^3 - y^3$,
5. $z = x^2 - y^2 + x$,
6. $z = x^2 - y^2 + y$,
7. $z = x^2 + y^2 + x$,
8. $z = x^2 + y^2 + y$,
9. $z = \sin(xy)$,
10. $z = \cos(xy)$,
11. $z = \mathrm{tg}(xy)$,
12. $z = xy$,
13. $z = x - \sin(xy)$,
14. $z = x + \cos(xy)$,
15. $z = \sqrt{x^2 + y^2}$.

Построенные графики сохраните в файлы с расширением ***png.***

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лабораторная работа № 13
Создание типа данных «класс»

Задания для самостоятельной работы:

1. Создать класс с полями, указанными в индивидуальном задании.

2. Реализовать в классе методы:

- конструктор по умолчанию;
- деструктор для освобождения памяти (с сообщением об уничтожении объекта);
- функции обработки данных, указанные в индивидуальном задании;
- функцию формирования строки информации об объекте.

Создать проект для демонстрации работы: сформировать объекты со значениями-константами и с введенными с клавиатуры значениями полей объекта. В основной ветке программы создайте три объекта класса. Вывести результаты работы на экран.

Варианты заданий приведены в таблице 1.

Таблица 1. Варианты заданий к лабораторной работе № 13

№	Класс-родитель и его поля	Функция-метод 1 обработки данных	Функция-метод 2 обработки данных
1	Дата (три числа): день, месяц, год	Определить, является ли год високосным (кратным 4)	Увеличить дату' на 5 дней
2	Работник: фамилия, оклад. Год поступления на работу	Вычислить стаж работы работника на данном предприятии	Сколько дней прошло после года поступления на работу
3	Книга: название, количество страниц, цена	Вычислить среднюю стоимость одной страницы	Увеличить цену книги в два раза, если название начинается со слова «Программирование»
4	Время (три числа): часы. Минуты, секунды	Вычислить количество полных минут в указанном времени	Уменьшить время на 10 минут

№	Класс-родитель и его поля	Функция-метод 1 обработки данных	Функция-метод 2 обработки данных
5	Товар: наименование, цена, год выпуска	Определить, сколько лет назад был выпущен товар	Увеличить цену товара на 20%. Если в наименовании товара есть слово «ТУ».
6	Дата (три числа): день, месяц, год	Увеличить год на 1	Уменьшить дату на 2 дня
7	Книга: название, автор, год издания	Вычислить, сколько лет книге	Количество дней, прошедших после года издания книги
8	Работник: фамилия, оклад. Дата рождения	Вычислить возраст работника	Сколько календарных дней до исполнения работнику 50 лет
9	Время (три числа): часы. Минуты, секунды	Вычислить количество секунд в указанном времени	Увеличить время на 5 секунд
10	Четыре целых числа: a. b. C. D	Вычислить среднее арифметическое чисел	Определить максимальное из чисел
11	Работник: фамилия. Должность, оклад	Увеличить оклад на 15% (каждому работнику)	Работникам, у которых фамилия начинается С сочетания букв «Иван», присвоить должность «инженер».
12	Книга: название, количество страниц, цена	Увеличить количество страниц на 10	Уменьшить цену в два раза, если количество страниц больше 100 (после увеличения)
13	Дата (три числа): день, месяц, год	Определить, совпадают ли номер месяца и число дня	Увеличить дату на один месяц
14	Товар: наименование, цена в рублях, изготовитель	Пересчитать цену товара в евро	Увеличить цену товара в евро, если название товара содержит слово «Samsung».
15	Время (три числа): часы. Минуты, секунды	Определить количество минут до полуночи (24:00:00)	Увеличить время 100 минут

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

Лабораторная работа № 14-15
Наследование и полиморфизм

Задания для самостоятельной работы:

1. На основании предложенной предметной области спроектировать 3-4 класса, используя механизм наследования. Для каждого класса использовать отдельный модуль.
2. Предусмотреть у класса наличие полей, методов и свойств. Названия членов класса должны быть осмысленны и снабжены комментариями.
3. Один из наследников должен перегружать метод родителя.
4. Один из классов должен содержать виртуальный метод, который переопределяется в одном наследнике и не переопределяется в другом.
5. Продемонстрировать работу всех объявленных методов.
6. Продемонстрировать вызов конструктора родительского класса при наследовании.

Варианты заданий приведены в таблице 2.

Таблица 2. Варианты заданий к лабораторной работе № 14

Вариант 1*	Написать программу, в которой описана иерархия классов: средство передвижения (велосипед, автомобиль, грузовик). Базовый класс должен иметь поля для хранения средней скорости, названия модели, числа пассажиров, а также методы получения потребления топлива для данного расстояния и вычисления времени движения на заданное расстояние. Продемонстрировать работ}' всех методов классов, предоставив пользователю выбор типа объекта для демонстрации.
Вариант 2	Написать программу, в которой описана иерархия классов: человек («дошкольник», «школьник», «студент», «работающий»). Базовый класс должен иметь поля для хранения ФИО. Возраста, пола, а также методы получения среднего дохода и среднего расхода в денежном эквиваленте. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа объекта для демонстрации.
Вариант 3	Написать программу, в которой описана иерархия классов: геометрические фигуры (круг, прямоугольник, треугольник). Реализовать методы вычисления площади и периметра фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.

Вариант 4	Написать программу, в которой описана иерархия классов: геометрические фигуры (эллипс, квадрат, трапеция). Реализовать методы вычисления площади и периметра фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 5	Написать программу, в которой описана иерархия классов: геометрические фигуры (ромб, параллелепипед, эллипс). Реализовать методы вычисления площади и периметра фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 6	Написать программу, в которой описана иерархия классов: геометрические фигуры (куб. цилиндр. Тетраэдр). Реализовать методы вычисления объема и площади поверхности Реализовать методы вычисления объема и площади поверхности фигуры. Продемонстрировать работ}' всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 7	Написать программу, в которой описана иерархия классов: геометрические фигуры (куб. конус, тетраэдр). Реализовать методы вычисления объема и площади поверхности фигуры. Продемонстрировать работ}' всех методов классов, предоставив пользователю выбор типа фиг\'ры для демонстрации.
Вариант 8*	Написать программу, в которой описана иерархия классов: геометрические фигуры (ромб, прямоугольник, эллипс). Реализовать методы вычисления площади и периметра фигуры. Продемонстрировать работу' всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 9	Написать программу, в которой описана иерархия классов: функция от одной переменной (синус, косинус, тангенс). Базовый класс должен иметь методы получения значения функции яля данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу' всех методов классов всех классов.
Вариант 10	Написать программу, в которой описана иерархия классов: функция от одной переменной (секанс, косеканс, котангенс). Базовый класс должен иметь методы получения значения функции ятя данного значения переменной, а также создания экземпляра ктасса. Представляющего собой производную текущего экземпляра. Продемонстрировать работу всех методов классов всех классов.
Вариант 11	Написать программу, в которой описана иерархия классов: функция от одной переменной (арксинус, арккосинус, а также класс, необходимый для представления производных). Базовый

	класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу[7] всех методов классов всех классов.
Вариант 12	Написать программу, в которой описана иерархия классов: функция от одной переменной (арктангенс, арккотангенс, а также класс, необходимый ятя представления производных). Базовый класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу[7] всех методов классов всех классов.
Вариант 13	Написать программу, в которой описана иерархия классов: функция от одной переменной (логарифм, натуральный логарифм, а также класс, необходимый для представления производных). Базовый класс должен иметь методы получения значения функции ятя данного значения переменной. А также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу[7] всех методов классов всех классов.
Вариант 14	Написать программу, в которой описана иерархия классов: функция от одной переменной (экспонента, гиперболический синус, гиперболический косинус). Базовый класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу всех методов классов всех классов.

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТА С ПРЕПОДАВАТЕЛЕМ (СРСП)

Методические указания.

Для успешной сдачи СРСП студент должен разобрать теоретические вопросы указанной темы, проработать (разобрать решенные, решить самостоятельно) задания к практическим работам. К каждому СРСП необходимо разработать программу решения задачи, скомпилировать ее и получить результат. Студент должен уметь интерпретировать результаты работы программы, предопределять результаты при изменении текста программы. Также должен уметь формулировать постановку задач по теме.

1. Темы домашних работ:

1. Найдите последовательность в десятичной записи пи (200 000 000 символов).
2. Обработка числовых данных с помощью закона benford.
3. Генерирация текста с помощью цепей Маркова.
4. Анализ джекпотов новостей в Интернете.
5. Овладейте полезными командами оболочки с на основе cmd модуля.
6. Приложение, которое позволяет Вам читать текст с помощью просьбы ответить.
7. Конструкция множества Мандельброта.
8. Программа, которая помогает выучить иностранные языки.
9. Веб-сервис для изобретения детских имен.
10. Таблицы schulte для мобильных устройств.
11. Совместная разработка веб-сервиса с помощью общего репозитория мерзавца (опции для групп из 3-4 человек).
12. Сравнение реализации ООП в Python и других языках (C++, deplhi). Примеры.
13. Независимое исследование основ языка haskell. Описание функций, характерных для функционального программирования в haskell и Python.
14. ООП в Python. Через Примеры.
15. Покажите пример использования самых популярных шаблонов разработки.

16. Разработка brainfuck интерпретатора языка. Его соединение с машиной Тьюринга.

17. Запись подобной прологу системы баз данных.

18. Программный интерфейс Твиттера самостоятельного обучения, пишущий программу для отображения последних твитов данного пользователя.

19. Обычно используемые встроенные модули Python с примерами.

20. Запишите программу для генерации судоку данной сложности.

21. Запишите json преобразователь данных в структуры Python.

22. Запишите свою собственную асинхронную мини-платформу.

САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТА (СРС)

Методические указания.

По заданиям, где необходимо законспектировать тему студент обязан проработать не менее 3-4 источников. В конспектах должны содержаться практические примеры, иллюстрирующие суть изучаемого вопроса. Необходимо самому сформулировать и решить несколько задач по теме.

В заданиях, где вид задания – решение задач, необходимо разработать программу решения задачи. Студент должен уметь интерпретировать результаты работы программы, предопределять результаты при изменении текста программы. Также должен уметь формулировать постановку задач по теме.

№	Задание СРС	Вид задания	Сроки выполнения (день недели и время в соответствии с расписанием)	
			дата задания	срок сдачи задания
1	Язык программирования Python. Операторы языка. Программирование линейных алгоритмов	1. Даны гипотенуза и катет прямоугольного треугольника. Найти второй катет и радиус вписанной окружности. 2. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.	1 неделя	1 неделя
2	Операторы языка. Программирование разветвляющихся алгоритмов	1. Дано натуральное число n (n<= 9999). Является ли это число палиндромом (перевертышем) с учетом четырех цифр, как, например, числа 2222, 6116, 0440 и т. д.? 2. Дано натуральное число n (n < 100), определяющее возраст человека (в годах). Дать для этого числа наименования «год», «года» или «лет»: например, 1 год, 23 года, 45 лет и т.д.	2 неделя	2 неделя
3	Программирование циклических алгоритмов.	1. Определить, сколько в числе четных цифр, а сколько нечетных. Число вводится с клавиатуры. 2. Найти n-е число Фибоначчи.	3 неделя	3 неделя
4	Строки	1. Заменить пробелы символом «*». 2. Вывести из строки все цифры.	4 неделя	4 неделя

№	Задание СРС	Вид задания	Сроки выполнения (день недели и время в соответствии с расписанием)	
			дата задания	срок сдачи задания
5	Кортежи	1. В кортеже целых чисел найти индексы максимального и минимального элементов. 2. В кортеже вещественных чисел найти количество элементов, равных данному.	5 неделя	5 неделя
6	Списки.	1. В списке чисел проверить, все ли элементы являются уникальными, т. е. каждое число встречается только один раз. . 2. Из списка чисел удалить элементы, значения которых больше 50 и меньше 100. При этом удаляемые числа сохранить в другом списке.	6 неделя	6 неделя
7	Стандартный модуль array - массивы в python	1. Конспект темы 2. Дан массив. Удалить первое вхождение x из массива. Получить обратный порядок элементов в массиве.	7 неделя	7 неделя
8	Функции	1. Найти наименьшее общее кратное (НОК) двух чисел. Оформить в виде функции. 2. Написать функцию square, принимающую 1 аргумент - сторону квадрата, и возвращающую 3 значения (с помощью кортежа): периметр квадрата, площадь квадрата и диагональ квадрата.	8 неделя	8 неделя

№	Задание СРС	Вид задания	Сроки выполнения (день недели и время в соответствии с расписанием)	
			дата задания	срок сдачи задания
9	Рекурсия и рекурсивные алгоритмы	1. Дано натуральное число n. Выведите все числа от 1 до n. 2. В теории вычислимости важную роль играет функция Аккермана A(m,n), определенная следующим образом: $A(m,n)=\begin{cases} n+1, & m=0; \\ A(m-1,1), & m>0, n=0; \\ A(m-1,A(m,n-1)), & m>0, n>0. \end{cases}$ Даны два целых неотрицательных числа m и n, каждое в отдельной строке. Выведите A(m,n).	9 неделя	9 неделя
10	Стандартный модуль functools - сборник функций высокого уровня.	1. Конспект темы 2. Приведите примеры использования модуля functools.	10 неделя	10 неделя
11	Словари	1. Напишите функцию to_dict(lst), которая принимает аргумент в виде списка и возвращает словарь, в котором каждый элемент списка является и ключом и значением. Предполагается, что элементы списка будут соответствовать правилам задания ключей в словарях. 2. Дана строка в виде случайной последовательности чисел от 0 до 9. Требуется создать словарь, который в качестве ключей будет принимать данные числа (т. е.	11 неделя	11 неделя

№	Задание СРС	Вид задания	Сроки выполнения (день недели и время в соответствии с расписанием)	
			дата задания	срок сдачи задания
		ключи будут типом int), а в качестве значений – количество этих чисел в имеющейся последовательности. Для построения словаря создайте функцию count_it(s), принимающую строку из цифр. Функция должна возвратить словарь из 3-х самых часто встречаемых чисел.		
12	Файловый ввод-вывод. Чтение строк с помощью итераторов файлов.	1. Дан текстовый файл. Поменять в нем местами первую и четвертую строки. 2. Дан текстовый файл. Подсчитать длину каждой строки и результаты записать в новый файл.	12 неделя	12 неделя
13	Объектно-ориентированное программирование	1. Конспект темы. 2. Создать обобщенный класс, содержащий два поля и реализующий два действия: сложение и умножение. Написать программу, реализующую функциональность методов класса для объектов разных типов.	13 неделя	13 неделя
14	Графические возможности языка	1. Конспект темы. 2. Составить программу, результатом работы которой является законченная картинка	14 неделя	14-15 недели

Литература:

1. Бизли, Д. Python : Подробный справочник, 2010. - РМЭБ

2. Сысоева М.В., Сысоев И.В. Программирование для «нормальных» с нуля на языке Python, Учебник, Москва Базальт СПО МАКС Пресс 2018

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., перераб. и доп. - М. : Издательство Юрайт, 2019. - 161 с. - (Серия : Бакалавр. Прикладной курс).

4. Уэс Маккинли. Python и анализ данных. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.: ил.

ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ И ТЕСТЫ

1. Вопросы для подготовки к экзамену

1 Назначение языка программирования Python.
2 Переменные, значения и их типы.
3 Присваивание значения.
4 Ввод значений с клавиатуры.
5 Арифметические операции.
6 Вывод информации на экран.
7 Функции преобразования чисел.
8 Операции сравнения. Логические операции.
9 Условная конструкция if.
10 Цикл While.
11 Функция range().
12 Цикл for . Оператор else в цикле for.
13 Операторы break, continue.
14 Операции со строками. Методы.
15 Срезы (SLICING). Форматирование.
16 Понятие кортежа. Методы.
17 Понятие списка. Способы задания списка. Методы списков.
18 Вложенные списки. Матрицы.
19 Понятие функции. Параметры и аргументы.
20 Вызовы функций. Оператор возврата return.
21 Рекурсия.
22 Множества. Операции над множествами.
23 Словари. Работа со словарями. Методы словарей.
24 Файлы. Открытие файла.
25 Чтение/запись в файл.
26 Произвольный доступ к компоненте файла.
27 Модули и пакеты в Python. Основные стандартные модули и пакеты в Python.
28 Импортирование модулей.
29 Создание собственных модулей и их импортирование.
30 Пространство имен.
31 Графический интерфейс пользователя.
32 Основы Tkinter.

33 Виджеты. Стандартные виджеты Tkinter: кнопки, метки, текстовые виджеты, ползунки, переключатели. Свойства и методы.
34 Создание собственных виджетов.
35 Понятие события. Содержание события.
36 Обработка событий. События мыши.
37 Обработка событий. События клавиатуры.
38 Виды связей.
39 Диалоговые окна.
40 Графика на Python. Класс Paint.

2.Тесты

1.Что выведет данный код? print(8/-3)
10% -2
7% -3
4% 2
50% -2.6666666666666665
24% -2.6666666666666664
5% Узнать ответ

2.Что выведет данный код?
s = [420,1337, 69, 228]
print(s, s.sort())

27% [69, 228,420,1337]
5% [420,1337, 69, 228]
46% [420,1337, 69, 228] None
16% [69, 228,420,1337] None

3.Что выведет данный код?
print(all({}),any([]))

25% True True
33% False False
18% True False
14% False True

4. Что выведет код?

```
class a:
   def __enter__(self):
      print('1', end='')
      return none
   def __exit__(self, exc_type, exc_value, exc_tb):
      if exc_type is none:
         print('3', end='')
      else:
         print ('4', end='')
         return true
   with a() as a:
   print("2" if a else "3", end='')
   raise exception
   print("5", end="")
```

124
134
1245
1345
1234
12345

5. Что выведет код?

```
def fl(x):
global x
x+=l
print(x)
fl(15)
print("hello")
```

hello
16 hello
Ошибка
None

6. Что выведет код ?

```
a = [11, 3, 62]; b = sorted(a); c = ' '; f = 1
for i in range(len(b)):
c += str(b[:: -f])
f += 1
print(c)
```

8% [62]
5% [11,3,62]
62% [62,11,3][62,11][62]
12% [11,3,62][11,3][11]

7. Что выведет код выше?

```
a = [1, 1, 2, 3, 5, 8, 13, 9, 34, 55, 12]
b = [1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12]
for i in range(len(a)):
if a[i] = b[i]:
print(b(i])
else:
pass
```

6% 1,1,2,3,5,9
71% 1,5,9,12
8% 1,23,5,9
11% Ничего

8. Что выведет код выше?

```
a = (1, 4, 2, 3, 5, 8, 13, 21, 34, 55, 89]
for i in range(len(a)):
if a[i] < 5:
print(a[i])
else:
pass
```

5% 1,2,3,4,5
8% 1,4.23,5
13% 1,2,3,4
73% 1,4,2,3

9. Что выведет код выше?

```
def is.pal(s):
rs = s(::-1]
 return rs = s
print(is_pal('pip'))
print(is_pal(‘python*))
```

4% False, True
9% True, True
19% False, False
62% True, False

10. Что выведет код выше?

```
def resize(s):
if s s.lower():
return print($.lower())
elif s # s.upper():
return print(s.upper())

resize(’python’)
resize(’python*)
```

74% python, PYTHON
8% python, python
8% PYTHON, python
5% PYTHON, PYTHON

11. Что выведет код?

```
def pri(s):
print(s[o] + s[-l])
def prin($):
pri(s)
prin(’python’)
```

1% Np
5% Py
87% Pn
5% Po

12. Что выведет код?
a=10
print(str((a^2)/50)*3)
4% 2
56% 222
5% 20
14% 333

13. Что выведет код?
x = 5; y = 10
x=x^y;y=x^ y;
x = x ^ y
print(x);
print(y)
24% 10,5
19% 5,10
12% 10,10
21% 100,25

14. Что выведет код?
alphabet.lower = [chr(ord("a") + i) for i in rangeG)]
for i in range(len(alphabet_lower)):
print(alphabet_lower[i])
30% ["а", "б", "в"]
45% а б в
1% аб
11% ["а", "б"]

15. Что выведет код?
t = '10c*; sign = t[-1]; t = int(t[:-1])
if sign.lower() = 'c':
t = round(t * (9 / 5) + 32)
print(str(t) + 'F')
1% 10c
6% 50
73% 50F
10% 10F

16. Что выведет код выше?
```
a = {1 : '10', 2: '20'}
for i in range(len(a)):
a[i] = i + (int(a[lj) / 2)
print(a(i])
```
24% 5 6
35% 5.0 6.0
10% 11 10
21% 10.011.0

17. Что выведет код выше?
```
a = 10; b = *5'; c = '7'
g = b + c; e = 0
for i in range(a):
e += int(g) / 10
print(int(e))
```
45% 57
32% 57.000014
12% 7.014
4% 7

18. Что выведет код выше?
```
from math import sqrt
a = -1; b = 1; c = 20
d = b*b-4*a*c
if d > 0:
xl = (-b + sqrt(d)) / (2 * a)
x2 = (-b - sqrt(d)) / (2 * a)

print("xl = %.2f; x2 = %.2f" % (xl, x2))
elif d = = 0: xl = -b / (2 * a)
print('xl = %.2f" % xl)
else: print("No roots")
```
12% x1 =4x2 = 5
55% x1 = 4.0 x2 = 5.0
9% x1 = 2 x2 = 3
8% x1 = 2.0 x2 = 3.0

19. Что выведет код выше?
a = 5; b = 0.1
if a > b:
a = str(b/a)
a = '00,2' + '5'
c = int(a(3] + a[4]) * 4
prtnt(c - int(b))
6% 0.25
25% 100
7% 25
53% Ошибка

20. Что выведет код выше?
def some.functton(a):
return (a + 5) / 2
my.formula = (some_function(i) for i in range(2)]
print(my_formula)
5% 2 3
6% [2, 3]
75% [2.5, 3.0]
10% 2.5 3.0

21. Каким будет результат выполнения кода?
print(int(5) + float(5) *0)
25% 5
6% 0
52% 5.0
8% 0.0
1% None
8% Error

22. Каким будет результат выполнения кода?
positive_numbers = [3, 6, 4]
sum_of_nums = sum(positive.numbers)
bool.of.nums = au(positive.numbers)
print(suncof_nums or bool.of.nums)
13
3% 10

47% True
6% False
3% None
3% Error
6% Узнать ответ

23. Каким будет результат выполнения кода?

```
def func{*args):
x = 0
for arg in args:
x += arg
return x / len(args)
prtnt(func(3, 4, 5))4.0
```

47% 4
9% 12
2% 2
5% O
5% Error

24. Каким будет результат выполнения кода?

```
def is_vowel(char):
all_vowel.s = "aetou"
return char in all_vowels

print(int(is_vowel('c')))
```

5% True
9% 1
22% False
3% None
9% Error

25. Каким будет результат выполнения кода?

```
x = 1
def up():
x = 2
def do():
global x
print (x)
```

do()
up()
52% 1
28% 2
3% o
6% None
6% Error

26. Каким будет результат выполнения кода?
def foo(m: str, v: str) -> int:
return foo._____annotations__
print(foo('a', ,b,)["m"])
34% <class "str">
8% m
5% a
12% None
20% Error

27. Каким будет результат выполнения кода?
any_list = [1, 2, 3]
del any_list[0]
prtnt(len(any_list[l:]))
48% 1
17% 2
25% 3
2% O
3% None
4% Error

28. Каким будет результат выполнения кода?
adata = "yellow", 20, "red"
a, b, c = adata
print(a)
58% Yellow
16% ('Yellow', 20,'Red')
1% 20
1% Red
2% None
19% Error

29. Каким будет результат выполнения кода?

```
def spt(string, delimeters="\n):
result =[]
word=" "
for char in string:
if char not in delimeters:
word += char
else:
result.append(word)
word = " "
if word:
result.append(word)
 return result
  print(spt("Hello world"))
```

52% ["Hello", "world"]
17% ["Hello World"]
18% ["h", "e", "l", "l", "o"]
4% O
3% Error

30. Какой метод не применяется к спискам в python?
64% list.peek()
9% list.extend()
11% list.pop()
8% list.append()
8% list.insert()

31. Какого оператора нет в Python
79% Are
12% Is
3% In
2% Or
4% Not

32. Какой будет результат выполнения кода: print(+-1)?
34% -1
49% Ошибка
8% O
9% 1

33. Без чего невозможно создать исключение в Python?
47% Try
6% Finally
35% Except
12% Assert

34. Что не является инструкцией в Python?
35% get
15% def
25% del
25% lambda

35. Какой метод не применяются к множествам в Python?
59% uix()
13% difference( )
12% union()
16% issuperset()

36. Что такое инкапсуляции в питоне?
70% Ограничение доступа к составляющим объект компонентам (методам и переменным)
12% Отношение линейного порядка на множестве слов над некоторым упорядоченным алфавитом
18% Сущность в виртуальном пространстве, обладающая определённым состоянием и поведением

37. Какого исключения нет по умолчанию в Python?
75% ParsingError
9% AttributeError
5% IndexError
11% KeyError

38. Какого режима работы с файлами нету в Python?^
31% 'c'
15% 'x'
42% '+'
12% 'b'

39. Что получится в результате выполнения следующего кода в Python?

a = [1,3, 8, 7] .
a[1:4:-1]

12% []
25% [7, 8,3,1]
14% [7, 3]
49% Ошибка

40. Каким способом не получится создать кортеж?
13% a = ('s')
12% a = ('s',)
62% a='s'
13% a = tuple('s')

41. Какой библиотеки нет по умолчанию в Python?
47% pillow
19% gzip
11% hashlib
23% io

42. Какая константа не предусмотрена в Python по умолчанию?
35% ValueN
26% NotImplemented
39% Ellipsis

43. Что выведет Python?
aka = {"user","bill", "password","hillary"}
print(aka['password'])

56% TypeError
1% user
1% bill
29% password
13% hillary

44. Что делает следующий код?
def a(b, c, d): pass
77% Определяет функцию, которая ничего не делает
5% Определяет список и инициализирует его
13% Определяет функцию, которая передает параметры
5% Определяет пустой класс

45. Какой будет результат округления в Python?.
round(2.75,1)
36% 2.8
39% 2.7
25% 3

46. Зачем нужны миграции в Python Django?
73% Для переноса изменений в моделях (добавление поля, удаление модели и т. д.) на структуру базы данных
15% Они содержат основные поля и поведение данных, которые вы храните
12% Для извлечения изданного объекта некоторого его фрагмента

47. Что делает базовый класс Tk() в Python?
66% Создаёт окно
10% Зацикливает программу
24% Отображает через себя виджет

48. Какой будет ответ?
10.5%3*82//5**
21% 19
8% 19.2
21% 28.8
20% 28

49. Каким будет результат округления в Python?
round(2.5)
35% 2
37% 3
26% 3.0
2% Посмотреть ответы

50. Как будет отформатирована следующая строка в Python?
'Hello, {}!' .format('Python')
65% 'Hello, Python!'
3% 'Hello Python,!'
7% 'Hello,! Python'
18% Вернёт ошибку

51. Какой будет ответ?
13.2%5+52/4//0.3**
45% 23.2
9% 24
18% 5.07

52. Какая операция не изменяет множество в Python?
42% set.intersection()
10% set.add()
13% set.pop()
22% set.discard()

53. Что делает функция pow() в Python?
58% Возвращает результат деления по модулю числа, возведённого в степень
13% Позиционирует курсор мыши по координатам
15% Возвращает список, в котором содержится результат деления и остаток

54. Что делает метод isnumeric() в Python?
65% Возвращает флаг, указывающий на то, содержит ли строка только числа.
23% Возвращает флаг, указывающий на то, содержит ли строка число в десятичной системе исчисления.
5% Возвращает флаг, указывающий на то, все ли символы строки являются печатаемыми.

55. Что делает метод memoryview() в Python?
52% прицепляет к объекту объект для просмотра (и изменения) памяти
9% возвращает безликий объект, являющийся базовым для всех объектов.
23% возвращает строку, содержащую представление объекта

56. Что такое полиморфизм в Python?

49% Разное поведение одного и того же метода в разных классах

8% Ограничение доступа к составляющим методам и переменным

39% Полиморфизм подразумевает то, что дочерний класс содержит все атрибуты родительского класса

57. Что делает метод capitalizef) со строкой в Python?

72% Возвращает копию строки, переводя первую буквы в верхний регистр, а остальные в нижний

8% Возвращает флаг, указывающий на то, содержит ли строка символы только нижнего регистра.

7% Возвращает флаг, указывающий на то, является ли строка идентификатором

5% Возвращает копию указанной строки, с конца которой устранены указанные символы

58. Какой будет ответ?

34//8%10**5**

20% 10

18% 0,001

10% 0,01

22% 1

59. Что выведет функция:

```
for i in range(2,10, 3):
print(i, end= ' ')
```

65% 2 5 8

8% 369

4% 4710

23% Syntax Error

60. Что покажет следующий код?

```
mixed = ['мак', 'просо', 'мак', 'мак', 'просо', 'мак', 'просо', 'просо', 'просо', 'мак']
zolushka = list(filter(lambda x: x == 'мак', mixed))
```

print(zolushka)
69% ['мак', 'мак', 'мак', 'мак', 'мак']
10% ['мак']
10% 'мак'
11% ['мак', 'просо']

61. Что отобразится при выполнении данного кода на Python 3.6?
print(a) if (a:=3) > 4 else print (0)
57% SyntaxError
5% 3
3% 4
31% o
4% a

62. Что выведет этот код?
a = 0
print(0) if a else print(1)
36% 1
23% 0
2% a
39% SyntaxError

63. Объект какого класса представляет собой результат функции range?
41% range
13% tuple
38% list
8% function

64. Как проверить объект a на принадлежность к классу B?
34% isinstance(a, B)
48% a.isinstance(B)
6% isinstance(B, a)
12% a.get_object(B)

65. Что такое метод класса?

39% Метод который можно применять к классу и изменять поведение без создания экземпляра этого класса

54% Любой метод, который принадлежит классу

6% Метод который создаёт класс

1% Метод который делает из класса метакласс

66. Что такое метакласс?

49% Класс, экземпляром которого является класс

13% Любой базовый класс в Python

14% Группа классов, схожая по функционалу

24% Класс, у которого нет экземпляров

67. Что выведется в результате выполнения кода?

len(set(list(str(1232414565))))

42% 6

40% 10

4% 7

14% 0

68. Какая функция позволяет узнать длину итерируемого объекта?

86% len()

10% length()

4% co unt()

0% con()

69. Как мы будем импортировать функция randint из модуля random?

81% from random import randint

12% import random.randint

4% import randint

3% import random*

70. Какой из этих типов относится к изменяемым?

54% diet

17% str

15% tuple

9% int

5% float

71. Как удалить объект в Python?
46% del a
9% delete a
22% a.delete()
23% a.del()

72. Какое максимальное количество параметров может принять в себя объект slice?
32% 3
3% **0**
7% 1
12% 2
46% Бесконечно много

73. Какой оператор используется для возведения в степень?
86% **
3% *
11% ^
0% xx

74. Как сделать x строкой, если x=15?
87% x = str(x)
5% x = x.str()
1% x = x.str
7% x = string(x)

75. Что выведет данный код?
0.1+0.2
44% 0.30000000000000004
51% 0.3
4% 0
1% 1

76. Какой вид форматирования строк появился последним в Python?
52% f-строки
23% .format
13% %s
12% все появились одновременно

77. Каким ключевым словом обозначаются анонимные функции?

72% lambda
3% alpha
14% anon
11% anonym

78. С помощью какого метода модуля random можно получить случайное целое число на заданном промежутке?

66% randomint()
27% random()
6% intrandom()
1% int()

79. Каким методом можно поднять все символы строки в верхний регистр?

62% upper()
26% uppercase()
4% isupper()
8% capitalize()

80. Каким ключевым словом обозначается нелокальная переменная?

30% nonlocal
52% global
7% local
11% unlocal

81. Какой оператор сравнивает два операнда на принадлежность к одному и тому же объекту?

44% is
35% = =
16% in
5% or

82. Как называется модуль для тестирования?

77% pytest
8% tester
8% pytesting
7% testing

83. Как называется следующее выражение:
[xfor x in range(10)]
72% Генератор списка
6% Генератор словаря
12% Генератор
10% Генератор кортежа

84. Что выведется в результате?
complex(1)
46% (1+0j)
29% (1)
12% (!)
13% (1i+1j)

85. Сколько знаков после запятой по умолчанию оставляет функция round()?
46% 0
18% 1
19% 2
17% -1 (округляет до десятков)

86. Что выведется в результате выполнения кода?
0.1+0.2 = =0.3
38% False
57% True
2% 0
3% 1

87. Что позволяет получить функция len ()?
49% Длину итерируемого объекта
36% Длину строки
13% Длину списка
2% Длину кортежа

88. Какой модуль отвечает за потоки в Python?
26% threading
31% thread
14% gil
29% threads

89. Какая функция позволяет проверить доступные методы объектов?

38% dir()
48% methods()
7% meths()
7% funcs()

90. class A: pass B = A
Что является классом, что экземпляром?
30% оба классы
58% A класс, B экземпляр
8% A экземпляр, B класс
4% оба экземпляры

91. Что выведется в результате?
a=0
b=a
a is b
56% True
16% Ошибка
14% False
14% None

92. Что будет в результате выполнения кода:
1 +1 *'1'
73% Ошибка
5% 12
16% 2
6% 22

93. Какая из этих команд не возведет x в квадрат
49% square(x)
19% x*x
19% pow(x, 2)
13% x**2

94. Без какого условного оператора не обходится ни одно условное выражение?

92% if
2% elif
3% else
3% break

95. Как правильно обозначить константу

43% PYTHONIST = 1
28% const pythonist=1
25% pythonist = const(1)
4% pythonist = 1.const()

96. Какой метод используется для выгрузки файла из памяти? (файл открыт функцией open)

65% close
6% delete
4% stop
25% remove

97. Какая функция используется для получения модуля числа?

64% abs()
17% mod()
14% module()
5% absolute()

98. Что выведет этот код? 'python ist'[:3] * 2

55% pytpyt
10% pypy
13% istist
22% Ошибка

99. Какой стиль программирования не поддерживает Python

56% поддерживает все вышеперечисленные
8% объектно-ориентированный
31% императивный
5% функциональный

100. Что из представленного не является объектом в Python?
71% В Python всё является обьектом
7% class
11% function
11% int

101. Что выведет этот код
x = 'python' print(f'{"x"} is the best')
28% x is the best
47% python is the best
23% Ошибка
2% x

102. Какого типа объект None? (type(None))
43% NoneType
36% None
10% bool
11% NaN

103. x = frozenset('pythonist') Что вернёт x.add('python')
50% Ошибка
6% 'python'
39% {'pythonist', 'python'}
5% <class 'set'>

104. На что ссылается self в экземпляре?
66% На сам экземпляр
19% На класс
5% На магический метод
10% На переменную self

105. Что выведет следующий код:
print('pythonist'[8:100])
42% t
6% pythonist
4% pythonis
48% Ошибка

106. Какой из этих модулей не входит в стандартную библиотеку python?

51% numpy
8% math
23% cmath
18% itertools

107. Что выведет этот код:

a = [1,2,3]
a[2] = 2 print(a)
31% [1,2,2]
4% [1,2, 3]
10% [2, 2, 2]
55% Ошибка

108. Как получить данные от пользователя?

76% input ()
18% get ()
5% read ()
1% cin ()

109. Какая библиотека отвечает за время?

81% time
10% Time
6% clock
3% timer

110. Какая функция предназначена для вывода в консоль?

88% print()
5% out()
6% log()
1% write()

111. Где правильно создана переменная?

82% num = float(2)
5% $num = 2
8% var num=2
5% int num(2)

112. Какая библиотека отвечает за случайности?
84% random
9% rand
2% shuffle
5% randint

113. Что выведет следующий код?
print(type(1/2))
76% class 'float'
9% class 'int'
4% class 'number'
3% class 'tuple'
8% class 'double'

114. Что выведет следующий код?
a = [1,2,3, None, (),[],]
print(len(a))
58% 6
5% 4
11% 5
6% 7
20% Syntax error

115. Что напечатает следущий код?
a = {"user","bill", "password","vasya"}
print(a['password'])
60% TypeError
1% user
1% bill
24% password
14% vasya

116. Что выведет этот код?
(1,2,3) == sorted((1,2,3))
15% Ошибка
67% True
18% False

117. Какой метод библиотеки time заставит программу некоторое время не работать?

78% sleep ()
4% stop ()
14% wait ()
4% await ()

118. Что выведет этот код?
False = = False in [False]
38% True
41% False
21% Ошибка

119. Что выводит этот код?,
a, * b, c, d. e = 'a b c d'.splitf)
print(b)
12% []
14% b
4% ()
11% None
44% Будет ошибка.

120. Что выводит этот код
print(round(555.555, -2))
11% 600.0
9% 555
29% 555.56
13% 600
25% Будет ошибка.

121. Что выводит этот код
a = 'lang = {}'
print(a.formatfen'))
57% lang = en
11% lang = {}
10% en
15% Будет ошибка.

122. Что выводит этот код
d ={1: 'a'. True: 'b'}
print(d[1])
29% b
44% a
25% Будет ошибка

123. Что выводит этот код
print(float(*nan*) is float('nan'))
20% False
42% True
34% Будет ошибка

124. Что совпадает с шаблоном **r"(ab | cd)?"** ?
18% "ab", пустая строка
33% "abed"
13% "a", "d"
24% Ничего из перечисленного

125. Что выводит этот код
num = 11.11
print(num.imag)
17% 0.0
18% 11
18% 11.11
16% 11.0

126. Каким будет результат выполнения кода?
print(0,2*5)
27% 0 10
44% 1.0
8% 0.10
2% 10
1% *0*
17% Error

127. Каким будет результат выполнения кода?

```
nums = [1,2,3,4,5]
nums =[::-1] [1] = 3
print(nums)
```

22% [1,2,3,4,5]
9% [1,2, 3,4,3]
45% [1,2, 3, 3, 5]
2% O
2% None
14% Error

128. Каким будет результат выполнения кода?

```
x = 0
for i in range(10):
    if i%2:
        x=+i%2
print(x)
```

44% 5
20% 0
6% 2
12% 7
11% Error

129. Каким будет результат выполнения кода?

```
x = [1,2,3]
y=x.copy()
print(y is x)
```

48% False
43% True
4% None
4% Error

130. Каким будет результат выполнения кода?

```
a = [1,2,3]
del a[::-1]
print(a)
```

45% []
3% False
11% None
1% {}
24% Error

ЛИТЕРАТУРА

Основная литература

1. Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри ; [пер. с англ. М.А. Райтман]. —Москва: Издательство «Э»,2017.— 624с.: ил.—(Мировой компьютерный бестселлер).

2. Гвидо ванн Россум и др. Язык программирования Python. / 2001 — 454 с.

3. Лутц, Марк. Изучаем Python, том 1,5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. 832 с.: ил. — Парад, тит. Англ.

4. Лутц, Марк.Ю Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. тит. англ.

5. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с., ил.

6. Сысоева М. В, Сысоев И. В. Программирование для «нормальных» с нуля на языке Python: Учебник. В двух частях. Часть 1 / Ответственный редактор: В. Л. Черный : — М.: Базальт СПО; МАКС Пресс, 2018. — 176 с. [+4 с. вкл]: ил. — (Библиотека ALT).

Дополнительная литература

7. Бенгфорт Бенджамин, Билбро Ребекка, Охеда Тони. Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка. — СПб.: Питер, 2019. — 368 с. ил. — (Серия «Бестселлеры O'Reilly»).

8. Любанович Билл. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. 480 с.: ил. — (Серия «Бестселлеры O'Reilly»).

9. Мусин Д. Самоучитель Python. Выпуск 0.2, 07 мая, 2017.— 154 с.

10. Прохоренок, Н. А. Python 3 и PyQt 5. Разработка приложений / Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2016. — 832 с.: ил.

11. Программирование на языке Python: метод. указания / сост. Т.П.Рубцова, М.В.Морозова – Самара, 2017. – 48 с.: ил.

12. Свейгарт, Эл. Автоматизация рутинных задач с помощью Python: практическое руководство для начинающих.: Пер. с англ.— М.: ООО "ИД. Вильямс", 2017.—592с.: ил.—Парал. тит. англ.

13. Уэс Маккинли. Python и анализ данных / Пер. с англ. Слинкин А. А. - М.: ДМК Пресс, 2015. - 482 с.: ил.

14. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. — М. : Издательство Юрайт, 2018. — 126 с. — (Серия : Бакалавр. Прикладной курс).

Электронные ресурсы:

15. https://www.python.org/ - официальный сайт поддержки языка Python.

16. Sourceforge.net/projects/pyseripter/ - Pyscripter - свободная интегрированная среда разработки языка Python для операционной системы Windows.

17. www.pyinstaller.org - программа для преобразования скриптов на языке Python в исполняемые коды.

18. ideone.com - онлайн-среда для программирования на разных языках.

19. www.onlinegdb.com - онлайн-среда для программирования на разных языках.

20. https://pythontutor.ru/ - бесплатный онлайн-курс программирования на языке Python.

21. pythonfiddle.com - онлайн-среда для программирования на языке Python.