ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

АБАЙ АТЫНДАҒЫ ҚАЗАҚ ҰЛТТЫҚ ПЕДАГОГИКАЛЫҚ УНИВЕРСИТЕТІ

Халықова Г.З., Идрисов С.Н.,
Маликова Н.Т., Азат Г.

PYTHON ТІЛІНДЕ ПРОГРАММАЛАУ НЕГІЗДЕРІ

(Оқу құралы)

Алматы
2022

УДК 004.4 (075.8)
ББК 32.973.202-018.2 я73
Р 99

Абай атындағы ҚазҰПУ жанындағы ҚР БҒМ РОӘК
«Педагогикалық ғылымдар» білім беру саласындағы
ОӘБ жарық көру үшін баспаға ұсынды
Хаттама №2, 24.12.2021 ж.)

Пікір жазғандар: п.ғ.д., профессор Б.Д. Сыдықов
п.ғ.д., доцент С.Т.Мұхамбетжанова
ф.-м.ғ.к, доцент Е.П.Макашев

Халықова Г.З.
Р 99 **Python тілінде программалау негіздері:** Оқу құралы. **Халықова Г.З., Идрисов С.Н., Маликова Н.Т., Азат Г.** – Алматы, ТОО Лантар Трейд, 2022. – 316 б.

ISBN 978-601-361-055-9

Оқу құралында Python тілінде программалау негіздері қарастырылған. Тілдің негізгі ұғымдары мен операторлары талданып, мысалдармен берілген. Студенттердің өзіндік жұмыстарына арналған жаттығулар мен программалар кітапханасы келтірілген. Оқу құралы педагогикалық жоғары оқу орындарының «Информатика» мамандығы білім беру бағдарламасына сай жазылған.

Оқу құралы жоғары педагогикалық оқу орындарының студенттеріне, сондай-ақ, жалпы білім беретін мектептер мен колледждердің мұғалімдері мен оқушыларына арналған.

УДК 004.4 (075.8)
ББК 32.973.202-018.2 я73

ISBN 978-601-361-055-9

АЛҒЫ СӨЗ

Программалау тілдерін оқыту мектептегі информатика курсының негізгі бөлімдерінің бірі болып есептеледі. Сол себепті программалауды оқыту болашақ информатика мұғалімдерін даярлау бағдарламасынан да негізгі орын алады.

Программалау – бұл өнер. Программалауды меңгеру үшін программалау тілдеріне арналған оқу құралдарын оқу жеткіліксіз. Өнер өз бетімен жұмыс істеуді талап ететіндіктен, программалауды жеткілікті деңгейде меңгеру оны озіндік дамытуды талап етеді, программаны өз бетімен жазып, проблемалық есептерді шешу, алгоритмдерін құрып, оны өз бетімен программалай білу қажет.

Қазақстан Республикасында жалпы білім беретін орта мектептің информатика курсында программалау тілі ретінде Python тілін оқыту ұсынылған және жоғары кәсіби мамандар даярлау білім беру бағдарламаларына, сәйкес ақпараттық технологияларға қатысты мамандықтардың білім беру бағдарламаларына да енгізілген.

Python программалау тілі нәтижені шығару үшін интерпретаторды пайдаланатын жоғары деңгейдегі («адамға оқуға ыңғайлы») программалау тілі болып табылады. Python тілі пайдаланушы өзінің программасына ешқандай қиындықсыз қоса алатын кеңейтілген стандартты модульдер кітапханасын қамтитын жүйе. Python программалау тілі Нидерланд Ұлттық математика және компьютерлік ғылымдар ғылыми зерттеу институтында 80-жылдардың аяғы мен 90-жылдардың басында жасалған (авторы Гвидо ван Россум (Guido van Rossum)). Бұл программалау тілі басқа программалау тілдерін (дәлірек айтқанда, C, C++ және Unix командалық қабықшасын) дамыту барысында туындаған. Тілдің «Python» деген атауы «Монти Пайтонның ұшатын циркі (Monty Python's Flying circus)» атты танымал британдық комедиялық шоудың құрметіне берілген.

Python философиясының негізгі принциптерінің бірі программалауды жаңадан үйренушілерге программа

кодының түсінуге ыңғайлылығы болып табылады. Оны Python тілінің төмендегідей негізгі ерекшеліктері ретінде тұжырымдауға болады:

- Python тілі – тегін таратылатын, ашық кодты программалық құрал;
- тілді меңгеру жеңіл, оның синтаксисі қарапайым;
- программа кодын оқу өте жеңіл, басқа программалау тілдеріндегідей шамадан тыс тыныс белгілерді пайдалану талап етілмейді;
- программа модульдік құрылыммен жазылады;
- ауқымды стандартты кітапханамен қамтамасыз етілген, оны пайдаланушы программасында оңай пайдалана алады;
- Python тіліндегі программа әртүрлі платформаларда орындалады, оның интерфейсі де өзгермейді және компиляция талап етілмейді;
- Python тілі жоғары деңгейдегі программалау тілі болып табылады, ол жадыны статикалық үлестіру қабілетіне ие, сонымен бірге, төменгі деңгейдегі модульдерді қосуға мүмкіндік береді;
- Python тілі әмбебап тілдер қатарына жатады, ол әрі процедуралық, әрі объектіге бағытталған программалау әдістерін сүйемелдейді;
- Python тілі пайдалануға икемді тіл, бұл тілдің көмегімен консольдық программаларды, графикалық интерфейс қосымшаларын, веб сервистерімен әрекеттесуді қамтамасыз ететін сценарийлерді құруға мүмкіндік береді.

Осы аталған мүмкіндіктеріне қарай, Python тілі әлемнің көптеген мемлекеттеріндегі оқу орындарында программалау тілі ретінде оқытылады.

Қазіргі уақытта программалауды үйретуге арналған әдебиеттер мен электрондық ресурстар жеткілікті, бірақ олардың көпшілігінде алгоритмі белгілі стандартты есептерді шешумен шектеледі. Бұл программалаудың төменгі деңгейін меңгеруге жеткілікті болғанымен, пайдаланушының қызығушылығын оятып, дамыта алмайды.

Ұсынылып отырған оқу құралында оқушыға – студентке – болашақ информатика мұғаліміне – нақты есептердің шығарылған мысалдар жиынтығы мен Python тілінің көмегмен шығарылатын қызықты есептер келтірілген. Ұсынылған есептер жиынтығы оқушылардың, студенттердің оз бетімен орындауын талап етеді. Оқу құралының негізіне практикалық оқыту принципі алынған, негізгі программалау курсының барлық бөлімінің тақырыптарын қамтиды. Оқу құралы программалаудың негізгі бөлімдерін қамтитын он төрт бөлімнен тұрады. Әрбір бөлімнің соңында оқу материалын пысықтайтын бақылау сұрақтары мен жаттығулар келтірілген. Оқу құралының соңында программалар кітапханасы, есептер мен студенттердің білімін өзіндік бақылауға арналған тест сұрақтары қамтылған. Программалаудың негізгі дағдыларын меңгеру әрі аудиториядағы оқытушының көмегімен, әрі өзіндік жаттығу жұмыстарын орындау арқылы қалыптасады.

Авторлар оқу құралын жазуға орта мектепте программалауды оқыту, педагогикалық жоғары оқу орындарында программалауды оқыту әдістемесі бойынша жинақталған тәжірибелерін пайдаланды.

1. PYTHON ТІЛІНІҢ НЕГІЗГІ ТҮСІНІКТЕРІ

1.1 Python программалау тіліне кіріспе

Python программалау тілін құруды 1980-жылдардың соңына қарай CWI (голл. Centrum Wiskunde & Informatica; Математика және информатика орталығы) атты голландиялық институттың қызметкері Гвидо ван Россум бастады. Бірақ Python программалау тілінің құрылғаны туралы 1996 жылы жазды. Қазіргі уақытта тілдің екі тармағы бар: Python 2.x және Python 3.x. Екінші және үшінші нұсқаларының арасында аздаған өзгешелігі бар. Python программалау тілі арнайы https://www.python.org сайты арқылы сүйемелденеді және тілдің интерпретаторы тегін таратылады.

Сонымен бірге, қазіргі уақытта практикада пайдаланылып жүрген операциялық жүйелердің бәріне арналған нұсқалары бар. Біз **Windows** операциялық жүйелеріне арналған нұсқаны қарастырамыз. Сондықтан оны https://www.python.org/downloads/windows арнайы сайтынан жүктеген дұрыс.

1.2 Тілдің негізгі ерекшеліктері. IDLE (Integrated Development and Learning Environment) ортасы

Қазіргі программалау жүйелерінде пайдаланушының жазған программасын машиналық тілге аударатын трансляторлар қарастырылған. Транслятор дегеніміз – программистің жазған қолданбалы программасын жоғары деңгейдегі программалау тіліне (машиналық код) аударып, оны орындауды жүзеге асыратын жүйелік программа болып табылады. Транслятордың екі түрі бар: интерпретатор және компилятор.

Интерпретатор программаның кезекті командасын оқып, оны бірден орындайды, программа мәтіні толығымен бірден машиналық кодқа аударылмайды. Дәлірек

айтқанда, программаны боліктеп оқып, бөліктеп орындайды. Ал, компилятор программа мәтінін толығымен оқып, оны машиналық кодқа аударып, машиналық тілдегі программаның аяқталған нұсқасын құрады. Егер программада қандай да бір қате кездесетін болса, компилятор сол қатені түзетпейінше, оны машиналық тілге аудармайды және программа да орындалмайды. Python тілінде жазылған программаның алдымен қатесі толығымен тексеріліп, сонан кейін машиналық тілге аударылады. Бірақ Python тілі интерпретатор режимінде де жұмыс істей алады (біз кейбір мысалдарды осы интерпретатор режимінде қарастырамыз).

Сондай-ақ, Python тілі программа құруға арналған жоғары деңгейдегі программалау жүйесі болып табылады, жүйеге транслятор, мәтіндік редактор (программаның бөлігін көшіруге, өшіруге және орнын алмастыруға мүмкіндік береді) және анықтамалық жүйені, программаны жүргізу және т.б. элементтерді қамтиды. Қазіргі уақытта Python тілінің көптеген нұсқалары бар. Сонымен, Python программалау тілі интерпретацияланатын тіл болып табылады. Программа коды машиналық тілге оны оқу барысында интерпретатор арқылы аударылады.

Python тілінің синтаксисі өте қарапайым, сондықтан басқа программалау тілдерімен салыстырғанда программаны оқып, түсіну қиындық туғызбайды. Бұл тілде программа жазуда «жақша», «нүктелі үтір» тәрізді қосымша тыныс белгілер көп пайдаланылмайды. Бірақ күрделі құрылымдарды сипаттап көрсету программистен «программистік шеберлікті» талап етеді. Мұндай қарапайымдылығына қарамастан Python программалау тілі қазіргі уақытта көптеген салаларда пайдаланылатын әмбебап тілдердің қатарына жатады. Сонымен қатар, программалау тілдерінің парадигмасына сай, императивті, функционалдық және объектіге бағытталған программалау да сүйемелденеді. Файлдың форматы: «.py».

Ұсынылып отырған оқу құралында біз Python тілінде программалаудың негізі болып табылатын құрылымдық программалаудың негіздерін қарастырамыз.

Python программалау тілінде жазылған программаны орындау орнатылған интерпретаторды талап етеді.

Интерпретатор – бұл программа мәтінін жол бойынша оқып, өңдеп программалау тілінде орындайтын программа болып табылады.

Интерпретатордың соңғы нұсқасын орнатқан дұрыс (сайт жоғарыда көрсетілген). Python программалық қамтамасыз етілуіне IDLE интегралданған программа құру ортасы кіреді, ол әрі интерактивті режимде, әрі мәтіндік редактор режимінде жұмыс істеуге мүмкіндік береді. Сонымен қатар, ортаның басқа да мүмкіндіктері бар.

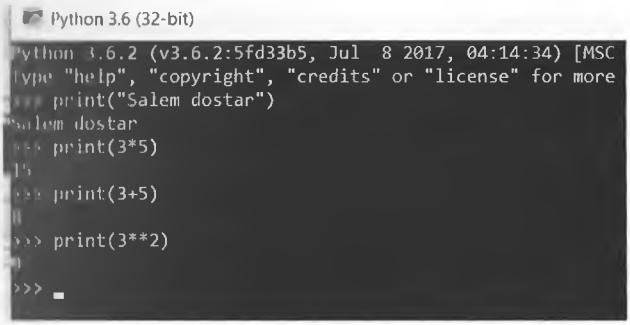
Python программа жазудың екі тәсілін ұсынады:

- интерактитивті режим: әрбір команданың орындалу нәтижесін көрсете отырып, жүйе мен пайдаланушы арасындағы диалогті ұйымдастыруға мүмкіндік береді;
- файлды орындау режимі: программа мәтіні жазылған файлды тұтасымен орындайды.

Интерактивті режимді программаның қандай да бір бөлігін жылдам тексеру қажет болған жағдайда немесе калькулятор ретінде пайдалануға болады. Қалған жағдайларда IDLE интегралданған программа құру ортасын пайдаланған дұрыс.

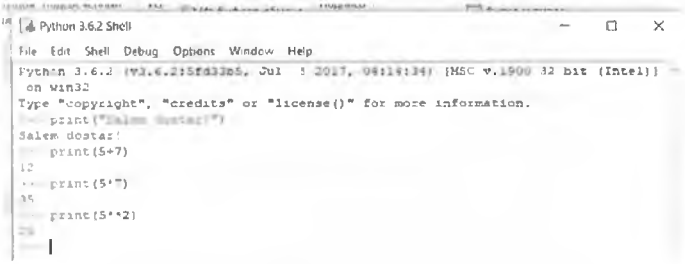
IDLE (Integrated Development and Learning Environment) ортасында программа құрылады. IDLE (Python 3.7.9) қосымшасы арқылы орындалады. Ол үшін алдымен аталған қосымшаны ашып, файл құру қажет. Одан кейін ашылған терезеге программа коды жазылады. Есептеулерді интерпретатор режимінде де орындауға болады.

Енді интерактивті режимді пайдаланып, программа мысалын жазып көрейік (1.1-сурет).



Сурет 1.1. Python тілінің интерпретатор режимі

Алғашқы программа «Salem dostar» сөзін экранға шығару. Келтірілген мысалдардан экранға шығару жүйенің орнатылған **print()** функциясы арқылы орындалып тұрғанын көруге болады. Экранға мәтінді шығару үшін мәтін тырнақшаның ішіне алынып жазылады. Сонымен қатар, осы функцияны интерактивті режимде калькулятор ретінде пайдалануға болады.



Сурет 1.2. Python тілінің интерактивті режимі

1.3 Арифметикалық амалдар және өрнектер

Python тілінде басқа программалау тілдеріндегі тәрізді арифметикалық амалдар мен функциялар пайдаланылады. Төмендегі кестеде арифметикалық амалдар мен жүйеде орнатылған бір және екі аргументті функциялар келтірілген.

Арифметикалық амал белгілері және жақшалар арқылы жалғастырылған атауларды, сандарды және функцияларды **өрнектер** дейді. Мысалы, (5+7*2):1,9 және т.с.с. Python тілінде арифметикалық өрнектердің мәндері есептелгенде, әдеттегі математикадағыдай, амалдар өз орындалу тәртібі бойынша есептеледі: көбейту, бөлу, бүтін бөлу, бүтін қалдық алу, қосу, азайту тәртібімен амалдың жазылу реті бойынша орындалады. Егер өрнекте жақшалар пайдаланылса, онда алдымен жоғарыдағы тәртіп бойынша жақша ішіндегі өрнектің мәні табылады.

Кесте 1.

Python тілінде орындалатын арифметикалық амалдар

x + y	Қосу — x пен y қосындысы, мысалы, 5 + 2 = 7
x - y	Алу — x пен y-тің айырмасы, мысалы, 5 - 2 = 3
x * y	Көбейту — x пен y-тің көбейтіндісі, мысалы, 5*2 = 10
x / y	Бөлу - x-ті y-ке бөлгендегі бөлінді, мысалы, 5/2 = 2.5, 10 / 5 = 2.0, нәтиже барлық уақытта бүтін сан бола бермейді. Сондықтан ол нақты тип ретінде сипатталады.
x // y	Бүтін санды бүтін санға бөлгендегі бүтін бөлінді, мысалы, 5 // 2 = 2; 9 // 2 = 4.0, Нәтиже бүтін сан болады.
x % y	Бүтін санды бүтін санға бөлгендегі бүтін қалдық, Мысалы, 5 % 2 = 1
x ** y	X санының y дәрежесі, мысалы, 5 ** 2 = 25
abs(x)	X санының модулі
Round (x)	X санын дөңгелектеу, мысалы, round (5.2) = 5

round(x, n)	X санын үтірден кейінгі n таңбаға дейін дөңгелектейді, мысалы, round(5.625789,2) = 5.62
pow(x, y)	X санының y дәрежесі, x ** y жазбасымен бірдей
divmod (x, y)	Бұл функцияның орындалу нәтижесінде екі мән алынады: бүтін және қалдық, оның жазылу форматы төмендегідей: q, r = divmod(x, y)

Бақылау сұрақтары мен жаттығулар

1. Транслятор, интерпретатор және компилятор дегеніміз не?
2. Python программалау жүйесінің құрамына қандай элементтер кіреді?
3. Мына өрнектерді Python тілінде жазып, мәндерін есептеңдер:

а) (5+2 • 4):3;

б) $8-2+\left\lfloor \frac{21}{4} \right\rfloor$;

в) (2+3)-ті 6-ға бөлгендегі бүтін қалдықты табу;

г) 5+2:10;

д) (8+4):10;

е) 8:8;

4. Мына өрнекті Python тілінде жазыңдар:

$$\frac{3{,}18-4{,}98}{1{,}171-0{,}27} \bullet (2{,}7-1{,}9)+5{,}96$$

5. Төмендегі өрнектің мағынасы бола ма:

а) 2.5 // 7.1;

б) 7 % 1.2 ;

2. МӘЛІМЕТТЕР ЖӘНЕ ОЛАРДЫҢ ТИПТЕРІ. АЙНЫМАЛЫЛАР

2.1. Python тіліндедегі мәліметтер типтері

Python тілінде тұрақтылар, айнымалылар, функциялар, өрнектер пайдаланылады.

Python тілінде сандық типтердің төрт түрі және жолдық типтер пайдаланылады: INT (бүтін), **float** (нақты), **boolean** (логикалық), **complex( )**, **str** (жолдық).

Айнымалы типтерді пайдаланушылар өздері анықтайды. Оларға шектелген және көрсетілген типтер жатады.

Бүтін тұрақтылар – кез келген нүктесіз жазылған бүтін сан. Олар теріс "-" немесе оң "+" таңбалы болуы мүмкін, мысалы, 14, -356,0,4590. Типі бүтін болатын айнымалыны сипаттау үшін INT қызметші сөзі пайдаланылады. Айнымалыларды белгілеу үшін оларға атау беріледі. Бүтін сандарға мына амалдар қолданылады: +, -, *, // (бүтін бөлу: 5 // 2 нәтиже 2), % (бүтін санды бүтін санға бөлгендегі бүтін қалдық), мысалы, 5%2, нәтиже 1. Бұл операция арқылы санның тақ немесе жұп екенін анықтауға, Х саны У-ке қалдықсыз бөліне ме деген сұраққа жауап беруге болады.

Нақты сандар: Python тілінде нақты мән қабылдайтын айнымалыны бейнелеу үшін **float** қызметші сөзі пайдаланылады.

Бүтін және нақты типтерді арифметикалық тип дейді. Нақты типтегі шамаларға (//) және (%) амалдарынан басқа барлық арифметикалық амалдарды қолдануға болады.

Нақты сандардың аралығы –1E38-ден 1E38-ге дейін мантиссасын 11 мәнді цифрға дейін алуға болады.

Python тілінде программалауда **комплекс сандар** да пайдаланылады.

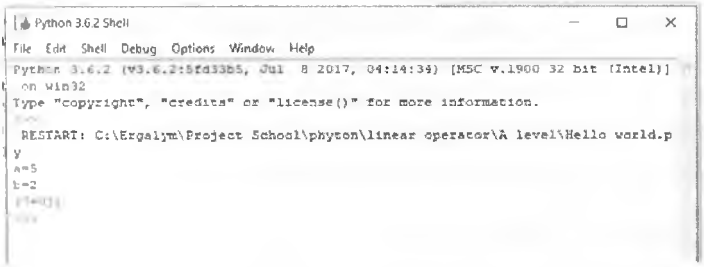
Комплекс сан – бұл a + bi түріндегі өрнек, мұндағы a және b – нақты сандар, i – жорамал бірлік деп аталады, оның квадраты - 1-ге тең ($i2 = -1$). a саны $z = a + bi$

түріндегі комплекс санның нақты бөлігі, ал b саны комплекс санның жорамал бөлігі болып табылады. Егер b = 0 болса, онда z = a болады. Бұдан нақты сандар – бұл комплекс сандардың дербес жағдайы екенін көруге болады.

Python тілінде комплекс сандарды сипаттау үшін **complex( )** қызметші сөзі пайдаланылады. Мысалы, IDLE редактор терезесіне төмендегі программа мәтінін енгіземіз:

```
a=complex(input("a="))
b=complex(input("b="))
z=a+b
print(z)
```

Программаны орындау үшін F5 командасын, басқару батырмасын басамыз немесе меню жолынан **Run → Run Module** (F5) командаларын орындаймыз. Нәтижесінде программаның орындалуын экраннан көреміз:



Сурет 2.1. Комплекс сандардың есептелуі

2.2 Нақты сандарға пайдаланылатын функциялар

Python тілінің библиотекасында өте коп функциялар орнатылған. Библиотекадағы функцияларды шақырудың бірнеше жолдары бар. Соның бірі import math деп аталады.

Алдымен math библиотекасында орналасқан функцияларды қарастырайық.

Бүтін және нақты сандарды бір өрнекте пайдалануда есептеу нақты сандармен жүргізіледі. Кей жағдайларда нақты санды бүтін санға дөңгелектеуге тура келеді.

Нақты сандарды дөңгелектеудің 4 тәсілі бар:

Int – санның бөлшек бөлігін шығарып тастайды.

Round – санды жұп санға қарай дөңгелектейді.

Floor – төменге, аз санға қарай дөңгелектейді.

Ceil – жоғары қарай, үлкен санға дөңгелектейді.

Әртүрлі сандарды дөңгелектеу мысалы 2.1-кестеде келтірілген.

Кесте 2.1.

Нақты сандарды дөңгелектеу функциялары

Функция	2.5	3.5	-2.5
int	2	3	-2
round	2	4	-2
floor	2	3	-3
ceil	3	4	-2

Floor және **cell** функциялары **math** библиотекасында орналасқан. Программада бұл функцияларды пайдаланудың екі тәсілі бар. Бірінші тәсілде math библиотекасынан импортталады және функцияны шақырғанда алдына math сөзі, одан кейін функция жазылады:

```
import math
print(math.floor(-2.5))
print(math.ceil(-2.5))
```

Екінші тәсілде кейбір функциялар библиотекадан импортталады және оны math сөзін жазбай пайдалануға мүмкіндік бар.

```
from math import floor, ceil
print(floor(-2.5))
print(ceil(-2.5))
```

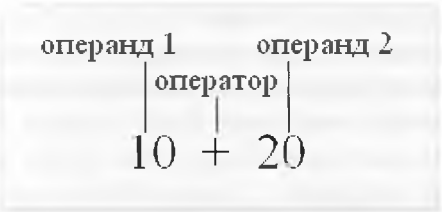
2.3. Жолдық типтер

Жолдар – тырнақшаға алынып жазылған символдар жиынтығы (мысалы, «алма», «компьютер» және т.б.).

Python тілінде (үшінші нұсқада) барлық жолдар utf-8 кодтау тәсілімен сақталады және 1 байттан жоғары орын алады. Жолдық типтерді көрсету үшін **str** типі пайдаланылады. (Жолдарға пайдаланылатын функциялар 7-тақырыпта толық қарастырылады).

2.4. Программалауда пайдаланылатын амалдар

Операция – бұл мәліметтермен қандай да бір әрекеттерді орындау, мұндағы мәліметтерді операндалар деп те айтады. Әрекетті орындайтын команда - **оператор** деп аталады.



Сурет 2.2. Оператор және операнда

Математикада, программалауда қосу «+» таңбасы сандарға қатысты қосу амалы (операциясы) болып табылады, ал жолдарға қосу амалы пайдаланғанда жолмен жолды «жалғастыру» болып табылады. Мысалы,

```
>>>'Қош'+'келдіңіздер!'
'Қошкелдіңіздер !'
```

Бірақ әртүрлі мәліметтер типін, мысалы, сандық типке жолдық типті қосуға болмайды. Мұндай жағдайда сәйкес қатенің типі көрсетіледі. Мысалы,

```
>>>10 +'Қош келдіңіздер!'
'Қошкелдіңіздер !'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and
'str'
```

2.5. Мәліметтер типтерін түрлендіру

Python тілінде мәліметтерді бір типтен басқа типтерге түрлендіру функциялары қарастырылған. Дәлірек айтқанда, сандық типті жолдық типке және керісінше арнайы орнатылған функциялардың **(int(), float(), str())** көмегімен түрлендіруге болады.

Егер 10 санын апостроф ішіне алып жазатын болсақ, онда ол жолдық шама болып табылады. Оны түрлендіру функцияларының көмегімен бүтін шамаға түрлендіруге болады.

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/ex1/display.py
    >>> int('3') + 9
    12
    >>> float('5.7') + int('3')
    8.7
    >>> str(7) + str(6.8)
    '76.8'
```

Мысалдан көріп отырғанымыздай, аталған функциялар жақшаның ішінде орналасқан мәнді сәйкес бүтін және нақты сандарға немесе жолдық шамаға

түрлендіреді. Бұдан бұл функция «барлық шамаларды түрлендіреді» деген тұжырым жасауға болмайды, өйткені аталған функциялар литерлік шамаларды түрлендіре алмайды. Мысалы,

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/ex1/display.py

>>> int('Qosh keldinizder')
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    int('Qosh keldinizder')
ValueError: invalid literal for int() with base
10: 'Qosh keldinizder'
>>>
```

Бұл мәннің қателігін көрсететін хабарлама, литерлік шаманы (әріп символдарымен берілген жолды) 10-дық санау жүйесіндегі бүтін типке түрлендіруге болмайтынын көрсетеді.

Сонымен қатар, **int()** функциясы, мысалы, екілік, он алтылық санау жүйелеріндегі сандарды ондық санау жүйесіне түрлендіре алады:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/ex1/display.py

>>> int('1010', 2)
10
>>> int('B', 16)
11
>>>
```

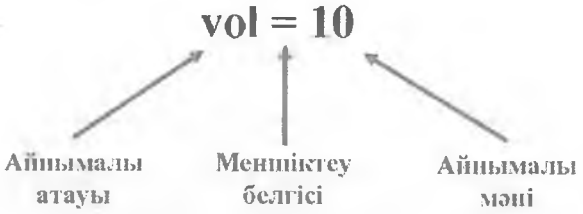
2.6. Айнымалылар

Мәліметтер компьютер жадысының ұяшығында сақталады. Пайдаланушының компьютерге енгізген саны компьютер жадысының қандай да бір ұяшығына барып орналасады. Мәліметтердің қайда орналасқанын қалай білуге және бұл мәліметтермен қалай жұмыс істеуге болады?

Мәліметтермен жұмыс істеу айнымалыларды пайдалану арқылы жүзеге асады.

Мәліметтер мен айнымалылардың арасындағы байланыс программалау тілдері мен мәліметтер типтеріне тәуелді ажыратылады. Программада мәліметтер қандай да бір атаумен байланыстырылады, атау символдар тізбегінен құралады және одан кейін мәліметтермен жұмыс істеу айнымалы атауы арқылы жүзеге асады. Математикада айнымалылар деп мәндері өзгеріп отыратын шамаларды айтады, программалау тілдерінде де дәл осы мағынада пайдаланылады. Программа құру барысында айнымалының бастапқыда бір мәнге ие болғанын, одан кейін программаның орындалуы барысында оның өзгеретінін көруге болады.

Басқа программалау тілдері тәрізді Python тілінде де мәліметтер мен айнымалылардың арасындағы байланыс «=» таңбасы арқылы орнатылады. Программалау тілдерінде бұл операцияны **меншіктеу** деп атайды. Мысалы, vol = 10 өрнегі компьютер жадысының белгілі бір бөлігінде орналасқан 10 санын көрсететін объект vol деп аталатын айнымалымен сипатталады.



Сурет 2.3. Меншіктеу операциясы

Айнымалылар кез келген символдар тізбегінен құралуы мүмкін, бірақ программалау тілдерінде айнымалыларға берілетін атау белгілі бір жалпы ережеге бағынуы тиіс:

- біріншіден, айнымалыға берілетін атау – мәннің мазмұны немесе қызметіне қарай таңдалғаны дұрыс, айнымалының атауы түсінікті болуы тиіс;
- екіншіден айнымалы атауы тілдің командаларымен (қызметші сөздерімен сәйкес келмеуі тиіс);
- үшіншіден, айнымалы атауы әріптен немесе сызықшадан (_) басталуы тиіс;
- төртіншіден, айнымалы атауында бос орын болмауы тиіс.

Интерпретатор режимінде айнымалының мәнін білу үшін айнымалы атауын жазып, Enter батырмасын басу жеткілікті.

Мысалы,

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/ex1/display.py
        >>> vol = 10
        >>> vol
        10
        >>>
```

Интерпретатор режимінде есептелетін күрделірек мысал қарастырайық.

Мектептегі информатика пәнінің жылдық сағаты – 68, аптасына 2 сағаттан оқытылады. Бірінші тоқсан аяқталғаннан кейінгі қалған сағат санын есептеп көрейік.

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/ex1/display.py
>>> total_h = 68
>>> week_h = 2
>>> quater_w = 8
>>> total_h = total_h - week_h*quater_w
>>> total_h
52
>>>
```

Бұл мысалда үш айнымалы пайдаланылды: **total_h** – жалпы сағат саны, **week_h** – аптасына оқылатын сағат саны, **quater_w** – бір тоқсандағы оқылатын апта саны. Айнымалылардың әрқайсысына мәндері меншіктелгеннен кейін меншіктеу операторы орындалады. Алдымен меншіктеу белгісінің оң жағындағы өрнек есептеліп, нәтиже **total_h** айнымалысына меншіктеледі. Нәтижесінде **total_h** айнымалысының бастапқы мәні өшіріліп, оның орнына соңғы есептелген мән меншіктеледі.

Бақылау сұрақтары мен жаттығулар

1. Python тілінде қандай мәліметтер типтері пайдаланылады?
2. Бүтін типтер қалай сипатталады және қандай амалдар қолданылады?
3. Нақты сандар қалай сипатталады және қандай амалдар қолданылады?
4. Нақты сандарды дөңгелектеу тәсілдерінің айырмашылықтары неде?
5. Комплекс сандар қалай сипатталады және комплекс сандардағы жорамал бірлік нені білдіреді?
6. Мәліметтер типтерін түрлендіру қалай жүзеге асырылады?
7. Интерпретатор режимінде сандарды ондық санау жүйесінен басқа да санау жүйелеріне түрлендіруге бола ма?

8. Жолдар деп нені түсінеміз және олар қалай сипатталады?

9. Жолдық шама ретінде сипатталған сандарды нақты немесе бүтін типке түрлендіруге бола ма?

10. Операция, операнд және оператор ұғымына сипаттама беріңдер.

11. Мәліметтер типтерін түрлендіру қалай жүзеге асырылады?

12. Программалау тілдерінде айнымалылар не үшін пайдаланылады?

13. Айнымалы атауларына қандай талаптар қойылады?

14. Программалау тілдерінде меншіктеу операторы қалай орындалады?

15. Программалау тілдеріндегі меншіктеу белгісі мен «теңдік» белгісінің айырмашылы қандай?

16. Интрепретатор режимінде айнымалының мәнін қалай анықтауға болады?

17. Python тілінде айнымалы атауларына қандай талаптар қойылады?

18. 7-сынып оқушысының информатика пәнінен бір апта ішіндегі сабаққа дайындалу уақытын, айнымалыларды пайдалана отырып, интерпретатор режиміне есептеңдер. Пән аптасына 2 рет оқылады.

19. 7-сынып оқушысының оқитын пәндерінің саны 12-14 болса және әр пәнді оқуға 30 минут уақыттан уақыт бөлінсе, оқушы аптасына қанша сағат сабаққа дайындалады?

20. Мектепте информатика пәні 4-сыныптан бастап, 11-сыныпқа дейін аптасына 2 сағаттан оқытылады және әрбір сынып саны 1 деп есептеп, информатика мұғалімінің апталық жүктемесін есептеңдер. Егер сынып саны 2 болса (мысалы, 4-сынып саны 2, «а» және «ә» сыныптары бар), апталық жүктеме қанша сағат болады? Есептеу интерпретатор режимінде жүзеге асырылсын.

3. PYTHON ТІЛІНІҢ ОПЕРАТОРЛАРЫ

3.1 Python тілінде мәліметтерді енгізу және шығару

Python тілінің шығару функциялары

Программалауда мәліметтерді енгізу мен шығару маңызды әрекеттер болып табылады. Егер программаға ешқандай мәліметтер енгізілмесе, программа толыққанды жұмыс істей алмайды. Ал мәліметтерді шығару – программаның жұмыс нәтижесін көруге мүмкіндік береді. Программаға мәліметтер файлдан, клавиатурадан, желіден немесе басқа программаның шығару нәтижелерінен енгізілуі мүмкін. Программадағы мәліметтерді шығару процесі де әртүрлі болуы мүмкін: файлға шығару, қағазға шығару, мониторға шығару. Программаның орындалу нәтижесі көбінесе монитор экранына шығарылады. Программаны ашық жүйе ретінде қарастыруға болады және программа ақпарат алмасу негізінде жұмыс істейді. Егер табиғатта үнемі зат пен энергияның алмасуы өтетін болса, ал программада – мәліметтер мен ақпарат алмасу процесі өтеді.

Егер программадағы операторлар бірінен кейін бірі орындалатын болса, мұндай программаның құрылымы сызықтық болып табылады және ол сызықтық алгоритмдерді программалауға жатады.

Python тілінде мәліметтерді шығару **print()** функциясының көмегімен жүзеге асырылады. Бұл функция жақшаға алынған мәндерді экранға шығару қызметін атқарады.

Біз осыған дейін интерактивті режимде print() функциясын пайдаланып кордік.

Енді экранға басып шығару функциясына бірнеше мысал қарастырайық.

```
print(10+20+30)
```

Программаның орындалу нәтижесі Python 3.6.2 Shell терезесіне шығарылады. Мұндағы 3.6.2 программаның нұсқасы болып табылады.

Сурет 3.1-а. Python тілінде тіліндегі программаның орындалу нәтижесі

Экраннан тек нәтижені ғана көруге болады. Алынған нәтиженің қай сандардың қосындысы екенін көру үшін оны мәтін ретінде басып шығаруға болады. Ол үшін экранға басылып шығарылатын символдар апостроф ішіне алынады, одан кейін үтір арқылы есептелінетін өрнек жазылады. Мысалы,

print(10+20+30)

print('10+20+30=',10+20+30)

Программаның орындалу нәтижесі 3.1-ә-суретте келтірілген.

Сурет 3.1-ә. Python тілінде тіліндегі программаның орындалу нәтижесі

Сонымен, қатар print() функциясының атаулы параметрлерін пайдалану арқылы да ажыратып жазуға болады:

print('30','20','10',sep='+')

Мұндағы «**sep**» атаулы параметрі ағылшынның «seperator – ажыратқыш» сөзінен алынған. Өрнектің нәтижесін шығару үшін екінші **print**() шығару функциясын жазамыз:

print('30','20','10',sep='+')
print('=',10+20+30)
Оның орындалу нәтижесі:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/ex1/display.py
        30+20+10
        = 60
        >>>
```

Нәтиженің екінші жолға шығарылғанын көруге болады. Әрбір оператор орындалғаннан кейін курсор келесі жолға көшіріледі. Сондықтан нәтиже екінші жолға шығарылып тұр. Нәтиженің бір жолға жазылуы үшін екінші параметрді пайдалана аламыз, ол **end** деп аталады және жолдың соңы екенін көрсетеді, дәлірек айтқанда, курсорды келесі жолға көшірмейді.

```
print('30','20','10',sep='+', end=' ')
print('=',10+20+30)
```

End параметрінен кейінгі апостроф бос орын қалдыру үшін қойылған. Оның орындалу нәтижесі:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/ex1/display.py
        30+20+10 = 60
        >>>
```

3.2 Мәліметтерді енгізу. Input() функциясы

Бұл функцияның қызметі басқа прогаммалау тілдеріндегі енгізу операторларының қызметіне ұқсас. Python тілінде мәліметтерді клавиатурадан енгізу **input()** функциясы арқылы жүзеге асырылады. Функция орындалғанда программадағы басқару пайдаланушыға беріледі, пайдаланушы клавиатурадан мән енгізіп, Enter клавишын басқаннан кейін **input()** функциясы енгізілген мәнді қабылдап, программаға береді және ары қарай

программа орындалуын жалғастырады. Бұл функцияны интерактивті режимде де пайдалануға болады. Бұл жағдайда енгізілген мән Enter клавишын басқаннан кейін бірден экраннан көрінеді.

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python36-
32/ex1/display.py
        >>> input()
        Hello!
        'Hello!'
        >>>
```

Input() функциясы енгізілген мәнді программаға береді және оны қандай да бір айнымалыға меншіктеуге болады. Енгізілген мән айнымалыға меншіктелген жағдайда интерпретатор оны бірден экранға басып шығармайды. Оны шығару үшін **print()** функциясы пайдаланылады. Мысалы,

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python36
-32/ex1/display.py
        >>> natije = input()
        All right!!!
        >>> print(natije)
        All right!!!
        >>>
```

Бұл мысалда енгізілген мән **natije** айнымалысына меншіктеліп тұр. **print()** функциясы арқылы айнымалының мәні шығарылатын болса, айнымалының аты апострофқа алынбайды.

1-мысал. Екі бүтін сан берілген. Берілген сандардың қосындысын, айырмасын, көбейтіндісін және бөліндісін табатын программа құрайық.

```
# екі санға қосу, азайту, көбейту, бөлу амалдарын қолдану
                    a=int(input("a="))
                    b=int(input("b="))
print ("a+b=", a+b,"a-b=", a-b,"a*b=", a*b, "a/b=", a/b )
```

Программаның орындалу нәтижесі:

```
RESTART: C:\Ergalym\Project School\phyton\linear operator\A level\екі санға қосу, азайту, көбейту, бөлу амалдарын қолдану.py
a=5
b=2
a+b= 7 a-b= 3 a*b= 10 a/b= 2.5
>>>
```

Барлық программалау тілдеріндегідей, Python тілінде де программаға түсіндірме беруге болады, ол # белгісінен басталып жазылады. Программадағы түсіндірме тек адамға арналған, сондықтан программа кодында түсіндірме кездескенде интерпретатор немесе компилятор оны орындамай өтіп кетеді.

2-мысал. Квадраттың периметрін есептейтін программа құрайық.

```
# квадраттың периметрі
a=int(input("a="))
p=4*a
print("p=", p)
```

Прогамманың орындалу нәтижесі:

```
RESTART: C:\Ergalym\Project School\phyton\linear operator\A level\квадраттың периметрі.py
a=5
p= 20
>>>
```

3-мысал. Кубтың көлемін және толық бетінің ауданын есептейтін программа құрайық.

```
# кубтың көлемі және толық бетінің ауданы
a=int(input("a="))
V=a**3
s=6*a**2
print("Көлемі:", V)
print("Толық бетінің ауданы:", s)
```

```
RESTART: C:\Ergalym\Project
School\phyton\linear operator\A level\кубтың
көлемі және ауданы.py
        a=3
        Көлемі: 27
        Толық беті: 54
        >>>
```

4-мысал. $У = X^4 + 3X^2 - 5X + 4$ өрнегінің мәнін есептейтін программа құрайық.

```
# y=x**4+3*x**2-5*x+4 өрнегін шешу
x=int(input("x="))
y=x**4+3*x**2-5*x+4
print("y=", y)
```

```
RESTART: C:\Ergalym\Project School\phyton\linear
operator\B level\өрнекті шешу.py
        x=2
        y= 22
        >>>
```

5-мысал. Екі бүтін сан берілген. Бірінші санды екінші санға бөлгендегі нәтиженің бүтін бөлігі мен қалдығын табатын программа құрайық.

```
# санның бүтін бөлігі мен қалдығын табу
x=int(input("num 1="))
y=int(input("num 2="))
```

```
A=x//y
B=x%y
print("Бүтін бөлігі:", A, "    ", "Қалдығы:", B)
```

```
RESTART: C:\Ergalym\Project
School\phyton\linear operator\B level\санның
бүтін бөлігі мен қалдығы.py
        num 1=12
        num 2=5
        Бүтін бөлігі: 2    Қалдығы: 2
        >>>
```

6-мысал. Үшбұрыштың барлық қабырғаларының ұзындығы берілген. Үшбұрыштың ауданын есептейтін программа құрайық.

```
# Үшбұрыштың жарты периметрі арқылы ауданын табу
from math import*    # математикалық кітапхана
a=int(input("a="))
b=int(input("b="))
c=int(input("c="))
p=(a+b+c)/2          # үшбұрыштың жарты периметрі
S=sqrt(p*(p-a)*(p-b)*(p-c)) # үшбұрыштың ауданы
print("Периметрі:", float(p),  "Ауданы:", float(S))
```

```
RESTART: C:\Ergalym\Project
School\phyton\linear operator\B level\санның
бүтін бөлігі мен қалдығы.py
        a=9
        b=8
        c=12
        Периметрі: 14.5
        Ауданы: 35.99913193397863
        >>>
```

7-мысал. Төрт орынды бүтін сан берілген. Осы санның цифрларының көбейтіндісін табатын программа құрайық.

```
x = int(input("4 орынды сан енгізіңіз:"))
k = 1
a = x//1000
k = k * a
print("1-number=",a)
a = (x//100)%10
k = k * a
print("2-number=",a)
a = (x%100)//10
k = k * a
print("3-number=",a)
a = x%10
k = k * a
print("4-number=",a)
print("санның цифрларының көбейтіндісі: ",k)
```

```
RESTART:C:/Ergalym/Project School/phyton/linear
operator/A level/number 4 pos.py
        4 орынды сан енгізіңіз:2345
        1-number= 2
        2-number= 3
        3-number= 4
        4-number= 5
        санның цифрларының көбейтіндісі:   120
        >>>
```

8-мысал. Енгізілген санның жұп екенін тексеретін программа жазайық.

```
n = int(input())
k = n % 2 == 0
print(k)
```

Бақылау сұрақтары мен жаттығулар

1. Енгізу және шығару функцияларының қызметтеріне сипаттама беріңдер.

2. Тұрақтылар енгізу функциясының параметрі бола ала ма?

3. Енгізу және шығару функцияларында енгізілетін және шығарылатын параметрлер санына шектеу қойыла ма?

4. Шығару функциясын пайдаланып, программаға түсініктеме жазуға бола ма?

5. Шығару функциясындағы «sep» және «end» параметрлерінің қызметі қандай және қай кезде пайдаланылады?

6. Егер a=5, b=7, c=2 болса, шығару функциясының параметрлері

(7, a+c,b) болса, экранда қандай сандар басылып шығады?

7. Төмендегі өрнектердің мәндерін есептеуге программа құрыңдар:

а) $y = \dfrac{\sin^2 2x^2 + 3\sqrt{x}}{\ln 5x + e^{6,5}}$

б) $y = \dfrac{1 + e^{\sin x}}{e^{\sqrt{x}} + x}, (x = 6,5)$

в) $y = \sqrt{x + \sqrt{(x+1) + \sqrt{x+2}}}, (x = 2);$

г) $y = \dfrac{\sqrt{x^2 + 1} + \sqrt{x-1}}{\sqrt{x+1}}, (x = 3);$

д) $y = \dfrac{\sin 2,7 + \ln 5}{e^{1,5} + \cos 0,71}$

8. x және y-тің әр түрлі мәндерін енгізіп, төмендегі өрнектердің мәндерін есептеуге программа құрыңдар:

а) $z = \dfrac{\sqrt{x^2 + y^2 - \sqrt{x+y}}}{e^x + e^y}$

б) $z = \dfrac{\sin(\cos x) - \cos(\sin y)}{\cos x + \sin x}$

в) $z = \dfrac{\sqrt{|\sin x|} + \sqrt{|\cos x|}}{\sqrt{|\sin x + \cos x|}}$

г) $z = \dfrac{e^{\sin x} - e^{\cos y}}{e^{(\sin x + \cos y)}}$

3.3 Логикалық өрнек және логикалық операторлар

Программалауда егер өрнектің есептелген нәтижесі ақиқат немесе жалған болса, мұндай өрнекті логикалық өрнек деп атайды. Мысалы, 12<25 өрнегі логикалық өрнек болып табылады, өйткені оның нәтижесі ақиқат немесе жалған болуы мүмкін. Ал 12+25 өрнегі - логикалық өрнекке жатпайды, өйткені оның нәтижесі сан болып табылады. Мәліметтердің төртінші типі логикалық тип болып табылады, оны буль типі деп атайды. Бұл типтің тек екі ғана мәні болады: **True** (ақиқат) және **False** (жалған). Мысалы,

```
>>> x = True
>>> type(x)
<class 'bool'>
>>> y = False
>>> type(y)
<class 'bool'>
>>>
```

Мұнда x айнымалысына True мәні меншіктеледі (мұндағы True және False сөздері бас әріппен жазылады), одан кейін Python тілінің орнатылған type() функциясының көмегімен оның типі тексеріледі.

Интерпретатор мұның класының **bool** айнымалысы екенін хабарлайды. Ал, у айнымалысы да **bool** айнымалысына жатады. Программалауда әдетте True – 1-ге, ал False - 0-ге теңестіріледі. Мұны интерпретатордың көмегімен тексеріп көруге болады:

```
>>> int(True)
1
>>> int(False)
0
>>>
```

Сондай-ақ, қандай да бір мәнді буль типіне түрлендіруге болады:

```
>>> bool(7.8)
True
>>> bool(-15)
True
>>> bool(0)
False
>>> bool('')
False
>>> bool(' ')
True
>>>
```

Бұдан, ноль мәнінің жалған екенін, ал нольге тең емес мәннің бәрі ақиқат болатынын көруге болады.

3.3.1. Логикалық операторлар

Бір объектінің элементтерін екінші объектінің элементтерімен салыстыру үшін шартты белгілер (үлкен, кіші, тең, тең емес) пайдаланылады. Салыстыру шартты белгілер арқылы тексеріледі. Шартты тексеру үшін

программалау тілдерінде математикадағы арнайы белгілер пайдаланылады.

Салыстыру белгілері	сипаттамасы
>	үлкен
<	кіші
==	тең
>=	Үлкен немесе тең
<=	Кіші немесе тең
!=	Тең емес

Бірқатар программалау тілдерінде меншіктеу командасы теңдік «=» белгісі арқылы белгіленеді, оны салыстыру операциясымен (==) шатастыруға болмайды. Төмендегі келтірілген мысалда алдымен a және b айнымалыларының мәндері қосылып, одан кейін салыстыру амалы орындалады. Мысалы, 15-ке 20 қосылады, одан кейін осы мән 30 санымен салыстырылады. Бұл арифметикалық амалдардың басымдылығы салыстыру амалдарынан жоғары екенін көрсетеді.

```
>>> a = 20
>>> b = 15
>>> a + b > 30
True
>>> a < 30 - b
False
>>> a != b
True
>>> a == b
False
```

Төмендегі мысалда өрнек екі операцияны қамтиды. Алдымен, a және b айнымалылары салыстырылып, одан кейін нәтиже c айнымалысына меншіктеледі. Ал a, b, c өрнегі экранға айнымалылардың мәндерін шығарады.

```
>>> c = a == b
>>> a, b, c
(20, 15, False)
>>>
```

3.3.2. Күрделі логикалық өрнектер

Егер логикалық өрнек бір ғана шарттан тұратын болса, онда ол қарапайым шарт болып есептеледі. Егер өрнек екі немесе одан да көп шарттарды (логикалық өрнектерді) қамтитын болса, онда ол құрама шарттар деп аталады. Құрама шарттарды жазу үшін арнайы логикалық операторлар пайдаланылады:

Операторлар	Сипаттамасы
And	және
Or	немесе
Not	терістеу

Практикада «және», «немесе» логикалық операторлары жиі пайдаланылады. **And** операторы пайдаланылған жағдайда **True** мәнін алу үшін оператор байланыстырып тұрған екі шартта ақиқат болуы (шарт сақталуы) тиіс. **Or** операторы пайдаланылған жағдайда **True** мәнін алу үшін оператор байланыстырып тұрған екі шарттың біреуі ақиқат болуы тиіс.

```
>>> k = 5
>>> p = 12
>>> k < 5 and p > 11
False
>>>
```

Келтірілген мысалда алдымен k айнымалысына 5, ал p айнымалысына 12 мәні меншіктеледі. k < 5 and p > 11 логикалық өрнегі былай орындалады:

k < 5 – өрнегінің мәні False ,

ал **p > 11** өрнегінің мәні True (ақиқат).

Одан кейін **False and True** өрнегі орындалады, оның мәні **False**.

And және Or екі өрнектің арасына қойылып, екі өрнекті де тексереді, сондықтан ол бинарлық операция болып табылады. Ал Not өрнектің алдына қойылады, егер өрнектің мәні ақиқат болса, Not оның мәнін теріске шығарады және керісінше.

Бақылау сұрақтары мен жаттығулар

1. Логикалық өрнек дегеніміз не?
2. Логикалық өрнектің мәні қандай жағдайда ақиқат болады?
3. **type()** функциясы қандай қызмет атқарады?
4. Логикалық операторлар қандай жағдайларда пайдаланылады?
5. Амалдардың орындалу басымдылығына сипаттама беріңдер.
6. a және b айнымалыларының қандай мәндерінде **a + b < 25** өрнегінің мәні жалған болады?
7. **k > 10 and p < 15** өрнегінің мәні ақиқат болуы үшін k және p айнымалыларының мәндері қандай болуы керек?
8. Екі айнымалыға әртүрлі екі сандық мән меншіктеңдер. Біріншісі ақиқат, екіншісі жалған мәнін беретіндей **and** операторының көмегімен екі логикалық өрнек жазыңдар.
9. 1-есептегі айнымалыларды пайдалана отырып, логикалық өрнекті **or** операторының көмегімен орындаңдар.
10. Жолдық типті айнымалыларды пайдалана отырып, логикалық өрнек жазыңдар.

4. PYTHON ТІЛІНІҢ ҚҰРАМА ОПЕРАТОРЛАРЫ

4.1 Тармақталу командасы. Шартты оператор

Тармақталуы бар процестерді ұйымдастыру үшін шартты оператор пайдаланылады. Тармақталу процесі белгілі бір шарттың орындалуы немесе орындалмауына тәуелді басқарылады.

Егер логикалық өрнектің мәні ақиқат болса, онда фигуралық жақшаның ішіндегі өрнек орындалады, егер логикалық өрнектің мәні жалған болса, онда фигуралық жақша ішіндегі өрнек орындалмайды. Программалау тілдерінде пайдаланылатын шартты операторлың құрылымы төмендегідей:

If логикалық өрнек {
1-өрнек;
2-өрнек;
...
}

If логикалық өрнек құрылымы шартты операторлың *тақырыбы* деп аталады. Фигуралық жақшаның ішідегі өрнек шартты операторлың *денесі* деп аталады. Шартты операторлың денесі бірнеше өрнекті қамтуы немесе бір оператор немесе бос болуы мүмкін.

Python тілінде фигуралық жақшаның орнына қос нүкте пайдаланылады. Ал оператор денесі бос орындармен оңға қарай жылжу арқылы орналасады, ол төрт бос орынға тең. Сондай-ақ, оңға қарай жылжуды клавиатурадағы **Tab** клавишасы арқылы да орындауға болады. Қос нүкте қойып, жаңа жолға көшкеннен кейін программалау ортасы автоматты түрде жаңа жолға көшуді орындайды. Интерактивті режимде жұмыс істеген жағдайда, жылжыту қолмен орындалады.

Python тілі синтаксисі айқын, программа коды оқуға жеңіл тілдер қатарына жатады. Тілдің синтаксисінен жақша тәрізді қосымша элементтер алынып тасталған. Өрнектерді бөлу жаңа жолға көшіру арқылы, ал

қабаттасқан өрнектер бос орындармен оңға қарай жылжу арқылы сипатталады. Басқа программалау тілдерінде бұл программа кодының оқылуын оңайлату үшін орындалса, бұл Python тілінде синтаксистік ережелерге енгізілген.

Python тілінде шартты операторды пайдалануға мысалдар қарастырайық.

```
if n < 100:
    b = n + a
```

Мұндағы логикалық орнек **n < 100**. Егер логикалық орнектің мәні ақиқат болса, **b = n + a** өрнегі орындалады, ал логикалық орнектің мәні жалған болса, онда бұл өрнек орындалмайды. Бұл мысал дәл осы күйінде жүйеге енгізілсе, шартты оператор орындалмайды. Себебі, көрсетілген айнымалылар сипатталмаған және тиісті мән енгізілмеген. Ол үшін программада a және n айнымалыларының мәнінің енгізілуі мен b айнымалысы есептегіш ретінде сипатталуы тиіс.

1-мысал. Қойылған шартқа сәйкес өрнектің мәнін есептеу.

Программа коды:

```
b = 0
a = int(input())
n = int(input())
if n < 100:
    b = n + a
print(b)
```

Программаның орындалу нәтижесі:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/
Python36-32/ex if 1.py
        65
        70
        135
        >>>
```

Программадағы соңғы оператор шартты операторға кірмейді. Сондықтан ол оператор оңға қарай жылжытылмаған. Программаның орындалуына түсініктеме берейік.

Программаның орындалуы мәндер енгізілуден басталады. Мұнда a айнымалысына 65, n айнымалысына 70 мәні енгізілген. Одан кейін шарт тексеріледі. **70 < 100** шарты ақиқат болғандықтан, шартты оператор денесіндегі өрнек (**b = n + a** операторы) орындалып тұр. Енді шарт жалған болатын мәндер енгізіп көрейік.

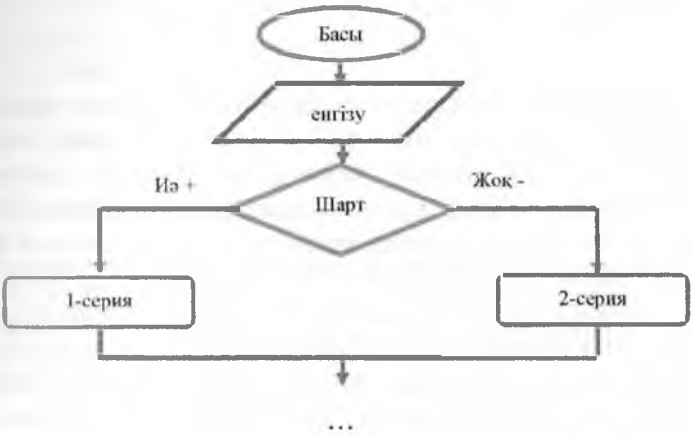
Енді a айнымалысына 102, n айнымалысына 105 мәні енгізілді. Алынған нәтиже 0. Себебі, **102 < 100** шарты жалған болғандықтан шартты оператор денесіндегі өрнек (b = n + a операторы) орындалған жоқ, b айнымалысының бастапқы мәні 0, ол экранға басылып шығып тұр.

```
    RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/ex if 1.py
    102
    105
    0
    >>>
```

if сөзінен кейін жазылатын шартты жазу үшін Python тілінде өрнектерді салыстыру =, >, >=, <, <=, <> белгілері пайдаланылады.

Шартты операторлардың блок-схемасы 4.1–суретте келтірілген.

Бір мезгілде бір емес, бірнеше шартты, яғни құрама шартты жазу үшін логикалық амалдарды пайдалануға болады.



Сурет 4.1. Тармақталуы бар программаның блок-схемасы.

Шартты оператордың тармағы біреу ғана емес, бірнешеу болуы мүмкін. Логикалық өрнектің орындалу нәтижесі жалған болған жағдайда (False), дәлірек айтқанда, қойылған шарт сақталмаған жағдайда, программадағы басқару бірден оператордың негізгі тармағына бармайды. Шарттың орындалу нәтижесі жалған болған жағдайда, ішкі қабаттасқан операторлар бөлігі орындалуы мүмкін. Шартты оператордың кеңейтілген нұсқасында кездескен ішкі операторлар бөлігін орындамай, шартты оператордың негізгі тармағына бармайды.

Программалау тілдерінде орындалатын операторларды шарттың сақталуына байланысты екі тармаққа болу шартты операторға else бөлігін қосу арқылы жүзеге асырылады. Сондықтан бұл операторды кейде if-else операторы деп те атайды. Оператордың жазылу форматы:

if логикалық өрнек:
    1-өрнек
    2-өрнек
    ...

else:
3-өрнек
...

Шартты оператордың орындалуы: алдымен шарт – логикалық өрнектің мәні тексеріледі, егер мән ақиқат болса, онда **if** сөзінен кейінгі операторлар сериясы орындалады, ал егер шарт нәтижесі жалған болса, онда **else** сөзінен кейінгі операторлар сериясы орындалады. **else** сөзінен кейін логикалық өрнек жазымайды.

2-мысал. Екі бүтін сан берілген. Егер бірінші санның абсолюттік шамасы екінші санның абсолюттік шамасынан үлкен болса, онда бірінші санды 5 есе кішірейтіп, екінші санды квадраттап экранға шығару, кері жағдайда екі санды да өзгеріссіз экранға шығаратын программа құрайық.

```
# екі санды салыстыру
x = int(input("x="))
y = int(input("y="))
if abs(x)>abs(y):
   print("жауабы:", x/5, y*y)
else:
   print("жауабы:", x, y)
```

Программаның орындалу нәтижесі:

```
      RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/comparison of 2 numbers.py
      x=15
      y=5
      жауабы: 3.0 25
      >>>
```

3-мысал. Берілген санның жұп немесе тақ екенін анықтайтын программа құрайық.

санның жұп немесе тақ сан екенін анықтау

```
a = int(input("кез келген бүтін сан енгіз:"))
if a % 2 == 0:
   print("Бұл жұп сан")
else:
   print("Бұл тақ сан")
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/тақ не жуп сан1.py
        кез келген бүтін сан енгіз:25
        Бұл тақ сан
        >>>
```

Шартты оператордағы логикалық өрнек барлық уақытта бір ғана шарттан тұратын қарапайым шарт болмауы, ол құрама шарт болуы мүмкін. Мұндай жағдайда арнайы логикалық операторлар пайдаланылады.

4-мысал.

$$Y = \begin{cases} X + 50, \text{ егер } 16 \le X \le 26 \\ 2X + 20, \text{ басқа жағдайлар үшін} \end{cases}$$

функциясыны
ң мәнін есептейтін программа құрайық.

```
    # құрама шартты функцияның мәнін есептеу
from math import*
x = int(input("Бір сан енгіз:"))
if x>=16 and x<=26:
    y=x+50
else:
    y=2*x+20
print("y=", int(y))
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/мысал 4 тармақ.ру
        Бір сан енгіз:5
        y= 30
        >>>
```

Python программалау тілінің басқа программалау тілдерінен өзгешелігі құрама шарт жақшаға алынбайды.

Бұл мысалда көрсетілген аралықта жатпайтын мән енгізілді, сондықтан **else** сөзінен кейінгі оператор орындалды.

Осы құрама шартты

if 16<=x<=26:

түрінде жазуға болады. Бұл Python тілінің синтаксисіне енгізілген. Программа ешқандай озгеріссіз орындалады.

```
# құрама шартты функцияның мәнін есептеу
from math import*
x = int(input("Бір сан енгіз:"))

if 16<=x<=26:
    y=x+50
else:
    y=2*x+20
print("y =", int(y))
```

Программаның орындалуы:

```
RESTART:
C:\Users\Asus\AppData\Local\Programs\Python\Pyth
on36-32\мысал 4 тармақ.py
        Бір сан енгіз:20
        y = 70
        >>>
```

Практикалық есептер шығару барысында логикалық өрнек тек шарттарды ғана қамтымайтын жағдайлар болуы мүмкін. Мысалы,

```
a = -7 < 0
if a:
   print(a)
```

Программаның орындалуы:

```
        RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/мысал 5 тармақ.py
        True
        >>>
```

Бұл мысалдағы **a** айнымалысының мәні буль мәнімен беріліп тұр. Программаның орындалуынан көріп отырғанымыздай, нәтиже ақиқат болып тұр. **a = -7 < 0** өрнегінде меншіктеу операторы салыстыру амалы орындалып болғаннан кейін ғана орындалады. Дәлірек айтқанда, алдымен **-7 < 0** шарты тексеріледі, одан кейін нәтиже **a** айнымалысына меншіктеледі. Алдыңғы тақырыпта нольге тең емес мәннің бәрі ақиқат болатыны туралы айтылған. Сондықтан, жүйе **True** (ақиқат) мәнін шығарып тұр.

5-мысал. функциясының мәнін есептейтін программа құрайық.

$$Y = \begin{cases} \sqrt{X}, \text{ егер } X > 0 \\ X^5, \text{ басқа жағдайлар үшін} \end{cases}$$

```
# жүйені шешу
from math import*
x = int(input("Бір сан енгіз:"))
if x>0:
    y=sqrt(x)
else:
    y=pow(x,5)
print("y=", float(y))
```

6-мысал. Берілген санның оң немесе теріс сан екенін анықтайтын программа құрайық.

```
# санның оң немесе теріс сан екенін анықтау
x = int(input("кез-келген бүтін сан енгіз:"))
if x > 0:
   print("Бұл оң сан")
else:
   print("Бұл теріс сан")
```

4.2. Көп тармақты шартты оператор

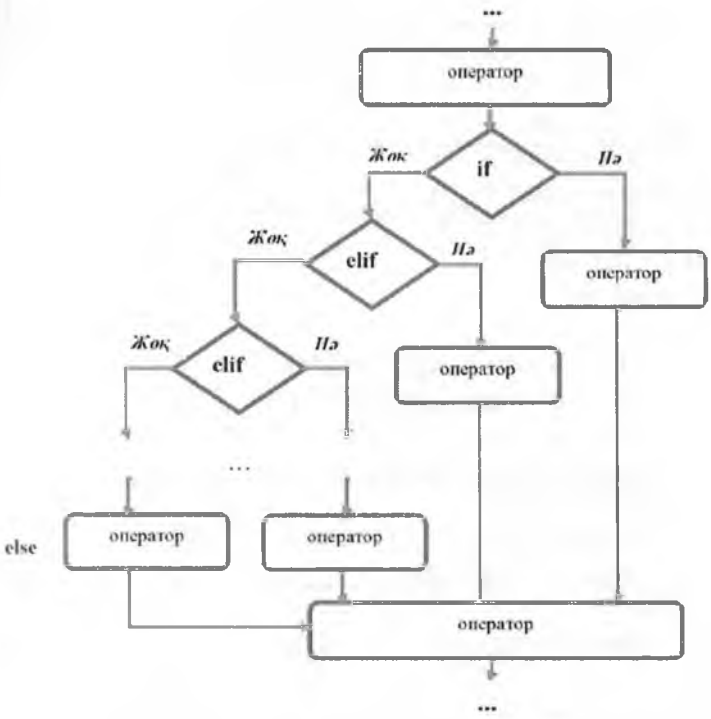
Python тілінде көп тармақты шартты тексеру мүмкіндігі қарастырылған. Ол **elif** тармағы арқылы жүзеге асырылады, **else** және **if** сөздерінен құралған. Бұл «әйтпесе егер» деп аударылады. **Else** сөзінен айырмашылығы **elif** сөзінен кейін міндетті түрде шарт жазылады. Көп тармақты шартты операторлардың орындалуы түсінікті болуы үшін оның блок-схемасын келтірейік (4.2-сурет).

Көп тармақты шартты операторды пайдалануға мысалдар қарастырайық.

1-мысал. $ax^2 + bx + c = 0$ түріндегі квадрат теңдеудің түбірлерін табатын программа құрайық.

$ax^2 + bx + c = 0$ квадрат теңдеуін шешу үшін алдымен дискриминант есептеледі:

$$d = b^2 - 4*a*c$$



Сурет 4.2. Көп тармақты шартты оператордың блок схемасы

Дискриминанттың анықталған мәніне байланысты теңдеудің түбірлері табылады:

Егер $d>0$ болса, теңдеудің екі түбірі бар, ол

$$x1 = \frac{-b + \sqrt{d}}{2a} \qquad x2 = \frac{-b - \sqrt{d}}{2a}$$

формулалары бойынша есептеледі.
Егер d = 0 болса, теңдеудің бір ғана түбірі бар:

$$x = \frac{-b}{2a}$$

Егер d<0 болса, онда теңдеудің түбірі жоқ.

```
import math
a=float(input("a: "))
b=float(input("b: "))
c=float(input("c: "))
d = (b*b-4*a*c)
print("d=",d)
if d>0:
   x1 = (-b + math.sqrt(d))/(2*a)
   x2 = (-b - math.sqrt(d))/(2*a)
   print("x1= %.2f" %x1)
   print("x2= %.2f" %x2)
elif d == 0:
   x = -b/(2*a)
   print("x= %.2f" %x)
else:
   print("No roots")
```

Программаның орындалуын 3 жағдай үшін де тексерейік.

1) d>0 болса, теңдеудің екі түбірі бар:

```
        = RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/kv equ.py =
        a: 1
        b: -5
        c: 6
        d= 1.0
        x1= 3.00
        x2= 2.00
```

2) d = 0 болса, теңдеудің бір ғана түбірі бар:

```
        RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/
Python36-32/kv equ.py =
        a: 2
        b: 4
        c: 2
        d= 0.0
        x= -1.00
        >>>
```

3) Егер d<0 болса, онда теңдеудің түбірі жоқ

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/
Python36-32/kv equ.py =
a: 8
b: 4
c: 2
d= -48.0
No roots
>>>
```

2-мысал. Енгізілген ұпай санына байланысты студенттің бағасын анықтайтын программа құрайық.

```
# Ұпай саны арқылы бағаны есептеу
B = float(input("Ұпай санын енгізіңіз:"))
if B>=85:
    print("Сіздің бағаңыз 5. Өте жақсы!")
elif B>=64 and B<85:
    print("Сіздің бағаңыз 4. Жақсы!")
elif B>=50 and B<64:
    print("Сіздің бағаңыз 3. Алға талпыныс жасаңыз!")
else:
    print("Сіздің бағаңыз 2. Сізге өте көп талпыныс қажет!" )
```

3-мысал.

$$Y = \begin{cases} X^2, \text{ егер } X > 0 \\ |X|, X < 0 \\ X + 5, X = 0 \end{cases}$$

функциясының мәнін есептейтін программа құрайық.

```
# үш шартты жүйені шешу
from math import*
x = float(input("Бір сан енгіз:"))
if x>0:   # x>0 шартты жағдай
    y=x**2
elif x<0:  # x<0 шартты жағдай
    y=abs(x)
else:     # x=0 шартты жағдай
    y=x+5
print("y=", int(y))
```

4-мысал. Енгізілген жыл ішіндегі күндер санын анықтайтын программа құрайық.

```
#Жыл қанша күннен тұрады?
year=int(input("year: "))
```

```
if year % 100==0:
  if year % 400==0:
    print("Бұл кібісе жыл яғни 366 күннен тұрады.")
  else:
    print("Бұл жылда 365 күн болады")
elif year%4==0:
  print("Бұл кібісе жыл яғни 366 күннен тұрады.")
else:
  print("Бұл жылда 365 күн болады")
```

5-мысал. Төртбұрыштың үшбұрышының координаталары берілген, төртінші бұрышының координатасын табатын программа құрайық.

```
# Төртбұрыштың 3 бұрышының координатасы арқылы, төртінші бұрышын анықтау
x1=input("Төртбұрыштың бірінші бұрышының x координатасын енгізіңіз:")
y1=input("Төртбұрыштың бірінші бұрышының y координатасын енгізіңіз:")
x2=input("Төртбұрыштың екінші бұрышының x координатасын енгізіңіз:")
y2=input("Төртбұрыштың екінші бұрышының y координатасын енгізіңіз:")
x3=input("Төртбұрыштың үшінші бұрышының x координатасын енгізіңіз:")
y3=input("Төртбұрыштың үшінші бұрышының y координатасын енгізіңіз:")
if x2==x3:
    x4=x1
if x3==x1:
    x4=x2
if x1==x2:
    x4=x3
if y2==y3:
    y4=y1
if y3==y1:
    y4=y2
```

```
if y1==y2:
    y4=y3
print("Төртінші бұрышының координатасы:")
print("x:", x4, "y:", y4)
```

Бақылау сұрақтары мен жаттығулар

1. Қандай жағдайларда шартты оператор пайдаланылады?
2. Шартты операторлардың жазылуының қандай түрлерін білесіңдер. Олардың мағына жағынан ерекшелігі неде?
3. Шартты операторға мысалдар келтіріңдер?
4. Шартты операторды пайдаланып, y=1/(x-1)+1/(x-2) мәнін есептеуге программа құрыңдар?
5. Төмендегі функциялардың мәндерін есептеуге программа құрыңдар:

а) $y=\begin{cases} X2, \text{ егер } -2 \le x \le 2 \\ 4, \text{ қалған жағдайлар үшін} \end{cases}$

б) $y=\begin{cases} Cos2x, \text{ егер } 0 \le x < 2 \\ 1+sin2x, \text{ қалған жағдайлар үшін} \end{cases}$

6. Үш санның ішінен үлкенін табуға программа құрыңдар.
7. Берілген нақты x аргументі үшін төмендегі функциялардың мәндерін есептеуге арналған программалар құрыңдар:

а) $y=\begin{cases} \dfrac{e^{\frac{\sin x}{2}}+\sin e^{\frac{x}{4}}}{x^2+2}, x<2 \\ 2x^2+\dfrac{1}{\sqrt{3x}}, x \ge 2 \end{cases}$

б) $y=\begin{cases} \sin^2 2x+5x^2, x>1{,}66 \\ \dfrac{61\sqrt{x}-17}{\sqrt{4+x^2+\cos^2 4x}}, x\le 1{,}66 \end{cases}$

в) $y=\begin{cases} \dfrac{e^x+1}{e^{x^2}}, x\le -2{,}4 \\ (x+\sin 4x)+\lg x^2, x\ge -2{,}4 \end{cases}$

8. Көп тармақты алгоритмдерді программалаудың әдістерін пайдаланып, төмендегі функциялардың мәндерін есептеуге программа жазыңдар:

а) $y=\begin{cases} \sqrt{(\sin x+1)}+\lg x^2, x<-1 \\ \dfrac{\sin x+\cos x}{\cos x}, -1\le x<3 \\ e^{-\sin x}+\sin x, x>3 \end{cases}$

б) $y=\begin{cases} 5\sin y+\cos y, y<1 \\ \dfrac{y^2-2y-5}{e^y}, 1\le y<4 \\ \sqrt{y^2+5}+\lg y, y\ge 4 \end{cases}$

Мұндағы $y=\begin{cases} \lg x+\sqrt{x}, x\le 2 \\ \lg 2x-\sqrt{3x}, x>2 \end{cases}$

в) $y=\begin{cases} \sqrt{x}+\sin x, x>0{,}4 \\ \sqrt{2-x}+\cos x, -1<x<0{,}4 \\ e^{5x-1}+\lg x, x\le -1 \end{cases}$

4.3. Кездейсоқ жағдайларды өңдеу. Try-except операторы

Программаны жазу процесінде немесе оны тестілеу процесінде кодта қате болатын болса, оны программист түзетеді. Практикада пайдаланушының әрекетінен программада кездейсоқ жағдайлар туындап жатады. Мысалы, программада сан енгізілуі керек, бірақ пайдаланушы санның орнына байқамай әріптер енгізіп қойды. Программадағы түрлендіру функциясы оны **ValueError** деп аталатын кездейсоқ жағдайдың туындауына әкеліп, программа жұмысын апатты түрде аяқтайды.

Мұндай жағдайлардың алдын алу үшін программалау тілдерінде, оның ішінде, біз қарастырып отырған Python тілінде де осындай кездейсоқ жағдайларды болдырмау жолдары қарастырылған. Дәлірек айтқанда, кездейсоқ жағдайларды дер кезінде тоқтатып, программа оқиғаларды өңдейді немесе программа жұмысын қалыпты жағдайда аяқтайды.

Python тілінде мұндай әрекетті **try-except** операторы орындайды. Try сөзі тырысып көру, ал except – кездейсоқтық деп аударылады. Бұл операторлың орындалуын былай сипаттауға болады: берілген операторды орындауға тырысу, егер кездейсоқ жағдай пайда болса, онда келесі операторды орындау керек. Бұл операторлың құрылымы шартты операторлың «әйтпесе» тармағына ұқсас.

Енді мысалдар қарастырайық.

1-мысал. Енгізілген бүтін санның дұрыстығын тексеретін программа құрайық.

```
n = input("Бүтін сан енгізіңіз: ")
```

```
try:
    n = int(n)
    print("дұрыс")
except:
    print("Мән дұрыс енгізілген жоқ!")
```

Кездейсоқ жағдай программа кодының 3-жолында, n айнымалысының мәнін бүтін санға түрлендіру кезінде туындауы мүмкін. Операторды орындау мүмкін болмаса, онда ары қарай **try** операторының денесі орындалмай, программа жұмысын тоқтатады. Бұл жағдайда print операторы орындалмайды. Бұдан программаның орындалуы **except** тармағына отіп, оның денесін орындайды. Егер **try** денесінде кездейсоқ оқиға болмаса, **except** тармағы орындалмайды.

Программа орындалғанда пайдаланушы бүтін сан енгізсе,

```
RESTART: C:/Ergalym/Project School/phyton/15 май
_2 часть/While  loop/C level/тексеру обработка.py
Бүтін сан енгізіңіз: 15
дұрыс
>>>
```

«дұрыс», ал егер әріптер енгізілсе, онда "Мән дұрыс енгізілген жоқ!" деген хабарлама шығады.

```
RESTART: C:/Ergalym/Project School/phyton/15
май _2 часть/While  loop/C level/тексеру
обработка.py
Бүтін сан енгізіңіз: FF
Мән дұрыс енгізілген жоқ!
>>>
```

Мұндағы ескерілуге тиісті тағы бір мәселе жоғарыда келтірілген программа кез келген кездейсоқ оқиғаны өңдейді. **Try** денесінде пайда болатын кездейсоқ оқиғаға

байланысты әр оқиғаның өңдеушісі бар. Мысалы, **except** сөзінен кейін кездейсоқ оқиғаның типін көрсету қажет және жоғарыдағы программаны былай жазған дұрыс:

```
try:
   n = input('Бүтін сан енгізіңіз: ')
   n = int(n)
   print("Дұрыс. Бүтін сан енгізілді", n)
except ValueError:
   print("Бүтін сан енгізілген жоқ!")
```

Егер денесінде одан да басқа кездейсоқ оқиға болатын болса, ол өңделмейді, ол үшін басқа **except** тармағын жазуға тура келеді.

2-мысал. Екі нақты сан берілген. Осы сандардың бөліндісін табатын программа құрайық.

Программада бөлгіш дұрыс енгізілмеген жағдайда санды нольге бөлуге болмайды деген хабарлама шығуы тиіс.

```
try:
   a = float(input("Бөлінгішті енгізіңіз: "))
   b = float(input("Бөлгішті енгізіңіз: "))
   c = a / b
   print("Бөлінді: %.2f" % c)
except ValueError:
   print("жолдық шама енгізуге болмайды")
except ZeroDivisionError:
   print("Нольге бөлуге болмайды!")
```

```
RESTART: C:/Ergalym/Project School/phyton/15
МАЙ/Example for book (1)/While  loop/A level/try
except ex.py
Бөлінгішті енгізіңіз: 15
Бөлгішті енгізіңіз: 0
Нольге бөлуге болмайды!
>>>
```

Программа орындалғанда үш жолда кездейсоқтық жағдай болуы мүмкін, олар: енгізілген санды нақты санға түрлендіргенде және бөлуді орындағанда. Бірінші жағдайда - **ValueError**, ал, екінші жағдайда – **ZeroDivisionError**. Кездейсоқ жағдайдың әрбір типі **except** тармағы арқылы өңделеді. Бірнеше кездейсоқ оқиғаны бір тармаққа топтастыруға болады:

```
try:
    a = float(input("Бөлінгішті енгізіңіз: "))
    b = float(input("Бөлгішті енгізіңіз: "))
    c = a / b
    print("Частное: %.2f" % c)
except (ValueError, ZeroDivisionError):
    print("жолдық шама енгізуге болмайды !")
    print("Нольге бөлуге болмайды!")
```

Кездейсоқ оқиғаларды өңдеу операторының **except** тармағынан бөлек **finally** және **else** деп аталатын тармақтары бар. **except** бөлігіндегі кездейсоқ жағдайларға жауап берудің орындалуына тәуелсіз **finally** тармағы барлық уақытта орындала береді. Егер **try** бөлігінде кездейсоқ жағдайлар болмаса, **else** денесі жұмыс істейді. Дәлірек айтқанда, **except** бөлігіне көшу болмаған жағдайда жұмыс істейді.

3-мысал. Енгізілген бүтін санды квадраттап, экранға шығаратын программа құрайық.

```
s = 1
try:
    n = input(' Бүтін сан енгізіңіз: ')
    n = int(n)
except ValueError:
    print("мән дұрыс енгізілген жоқ")
else: # try бөлігінде кездейсоқ жағдай болмаған жағдайда
    print("Сан енгізілді, бұл дұрыс:", n)
finally: # кез келген жағдайда орындалады
```

```
    s = s * n*n
print(n,"санының квадраты=",s)
print("программаның соңы")
```

```
RESTART: C:\Ergalym\Project School\phyton\15
МАЙ\Example for book (1)\While  loop\A
level\except 3.py
 Бүтін сан енгізіңіз: 5
Сан енгізілді, бұл дұрыс: 5
5 санының квадраты = 25
программаның соңы
>>>
```

Бақылау сұрақтары мен жаттығулар

1. Программа орындалуындағы кездейсоқ жағдайлар деп нені түсінеміз?
2. **try-except** операторының қызметіне сипаттама беріңдер.
3. **finally** және **else** тармақтары қандай жағдайларда пайдаланылады?
4. Енгізілген екі мәннің біреуі жолдық шама болса, жалғастыру операторын орындайтын, кері жағдайда екі санның қосындысын табатын программа құрыңдар.

$$Y = \begin{cases} X^2, \text{ егер } X > 0 \\ |X|, X < 0 \\ X + 5, X = 0 \end{cases}$$

Функциясының мәнін есептейтін программа құрыңдар. Енгізілген мәннің сан екені тексеріліп, программаның аяқталғандығы экранға басылып шығуы тиіс.

5. ҚАЙТАЛАНУ ОПЕРАТОРЛАРЫ

Көп жағдайларда аргументтердің әр түрлі мәндері бойынша алгоритмнің белгілі бір бөліктерін немесе алгоритмдік тілдің бірнеше операторлар тобын бірнеше рет қайталауға тура келеді. Осындай процестерді ұйымдастыру үшін циклдік құрылымы бар алгоритмдер, дәлірек айтқанда, қайталанатын алгоритмдер (операторлар) пайдаланылады. Қайталанатын бөліктер әр түрлі заңдылықтар, ережелер бойынша құрастырылады.

Python тілінде қайталау процесін екі түрлі жолмен ұйымдастыруға болады:

- Алдын-ала берілген шарт бойынша қайталауды ұйымдастыру;
- параметрдің мәні бойынша қайталауды ұйымдастыру.

Бұл қайталаудың қай-қайсысын алсақ та, олар қайталану операторының көмегімен жүзеге асырылады. Қайталану операторлары құрама операторлардың қатарына жатады.

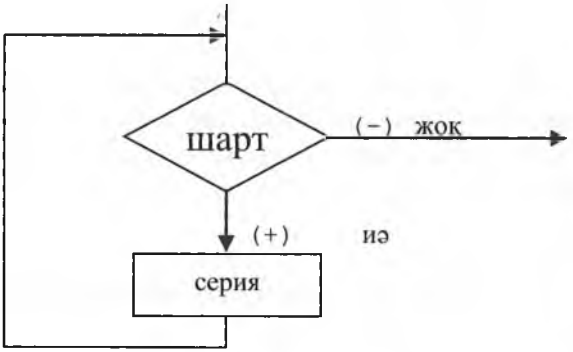
5.1. «Әзір» қайталану операторы (While)

Қайталану операторының бұл түрі алдын-ала берілген шарт бойынша жүзеге асырылады. Сондықтан бұл қайталау көп жағдайларда қайталану саны белгісіз болған жағдайда пайдаланылады. Бұл оператор WHILE (оқылуы-уаил) - әзір қызметші сөзінің көмегімен орындалады да, алгоритмдік тілдегі "әзір" қайталану командасына сәйкес келеді. Оның жазылу түрі мынадай:

Әзір шарт	WHILE логикалық өрнек:
Цб	
1-өрнек	1-өрнек
2-өрнек	2-өрнек
...	...
n-өрнек	n-өрнек
Цс	

Бұл оператордың орындалуы: алдымен шарт тексеріледі, егер шарт ақиқат болса (сақталса), онда циклдың денесі орындалады, одан соң қайтадан шарт тексеріледі және тағы сол сияқты, бұл процесс қашан шарт сақталудан қалғанша, дәлірек айтқанда, шарт жалған шартқа айналғанша қайталанады. Егер шарт бірінші тексеруден сақталмаса, шарт жалған болса, басқару негізгі программадағы қайталану операторынан кейінгі орналасқан операторға беріледі. Басқаша айтқанда, логикалық өрнектің мәні жалған болғанша, қайталану денесі қайталанып орындала береді.

Блок схема түрінде:



Сурет 5.1. «Әзір» қайталану командасының блок-схемасы.

Енді мысалдар қарастырайық.

1-мысал. 1-ден 5-ға дейінгі сандарды экранға шығаратын программа құрайық.

```
i = 1
while i <= 5:
    print(i,end=' ')
    i = i+1
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/15
МАЙ/Example for book (1)/While  loop/A lev-
el/сумма 5 чисел.py
1 2 3 4 5
>>>
```

1. Мұндағы **i** айнымалысы қайталану параметрі деп аталады, оның бастапқы мәні 1-ге тең. Алдымен цикл тақырыбындағы **i <= 5** шарты тексеріледі және оның нәтижесі ақиқат. Қайталану денесі орындалады, қайталану параметрінің мәні 1-ге артады.

2. Енді қайталану параметрінің мәні 2-ге тең. Шарт қайтадан тексеріледі, оның мәні ақиқат, цикл денесі екінші рет орындалады. Қайталану параметрінің мәні тағы да 1-ге артады.

3. Шарт тексеріледі, 3<5 болғандықтан цикл денесі 3-ші рет орындалады. Қайталану параметрінің мәні тағы да 1-ге артады.

4. Шарт тексеріледі, 4<5 болғандықтан цикл денесі 4-ші рет орындалады. Қайталану параметрінің мәні тағы да 1-ге артады.

5. Шарт тексеріледі, 5<=5 болғандықтан цикл денесі 5-рет орындалады. Қайталану параметрінің мәні тағы да 1-ге артады.

6. Қайтадан шарт тексеріледі, 6<=5 шарты жалған болғандықтан цикл денесі орындалмайды, циклдың орындалуы тоқтайды.

2-мысал. 2A мәні 3N+1 мәнінен үлкен болғанша, A айнымалысының мәнін 2-ге арттырып отыратын программа құрайық.

```
#2A мәні 3N+1 мәнінен үлкен болғанша, A-ға 2 санын қосу
A=1
N=1
while 2*A<=3*N+1:
```

```
A=A+2
N=N+1
print("A=", A, "N=", N)
```

```
RESTART::\Ergalym\Project School\phyton\15 МАЙ\Example for book (1)\While loop\A level\2A мәні 3N+1 мәнінен үлкен болғанша 2 санын қоса беру.py
A= 3 N= 2
A= 5 N= 3
A= 7 N= 4
>>>
```

3-мысал. Мәтінді экранға қайталап шығаратын программа құрайық.

```
#мәтінді экранға қайталап шығару
i=0
while i<10:
    print("Сәлем досым, сенімен танысқаныма қуаныштымын!")
    i+=1
```

4-мысал. 1-ден 5-ке дейінгі сандардың қосындысын табатын программа құрайық.

```
i = 1
s = 0
while i <= 5:
    s += i
    print(s,end=' ')
    i = i+1
```

5-мысал. Берілген бүтін санның [1..N] аралығындағы дәрежелерін есептеп шығаратын программа құрайық.

```
#Кез келген оң санның [1..N] аралығындағы дәрежелерін есептеу
a=int(input("a="))
N=int(input("N="))
```

```
i = 1
while i <= N:
    print(a**i)
    i =i + 1
```

6-мысал. Берілген бүтін санның факториалын есептейтін программа құрайық.

```
# N! есептеу
N=int(input("N="))
p=1
i=0
while i<N:
    i=i+1
    p=p*i
print("N!=", p)
```

7-мысал. Бүтін оң N саны берілген. N санына дейінгі жұп сандардың қосындысын есептейтін программа құрайық.

```
#жұп сандарды қосу
N=int(input("N="))
sum=0
i=0
while i<=N:
  if i%2==0:
    sum=sum+i
  i+=1
print("жұп сандардың қосындысы=", sum)
```

Құрама операторларды пайдаланғанда сол жақтан оң жаққа қарай қойылатын бос орынға мұқият болу керектігін айтқанбыз. Мына келтірілген мысалда екі құрама оператор пайдаланылып отыр. Қайталану операторының денесінде екі оператор, ал тармақталу (шартты) операторының денесінде бір ғана оператор бар. Бұл мысалдан қайталану денесіндегі операторлардың бір

бағанда және тармақталу денесіндегі оператордың шартпен бір бағанда орналасқанын байқауға болады.

8-мысал. Енгізілген сандардың ішіндегі ең үлкенін табатын программа құруды қарастырайық.

Программа қалай орындалады? Алғашқы енгізілген санды ең үлкен сан деп есептеп, оны max1 айнымалысына меншіктейміз. Қайталану шартын «әзір n != 0» деп аламыз. Цикл денесінде енгізілген сан біз үлкен деп есептеген санмен салыстырылады. Егер ол сан max1 шамасының мәнінен үлкен болса, max1 айнымалысына осы мәнді меншіктейміз. Осы процесс енгізілген мән 0-ге тең болғанша жалғастырылады.

```
n = int(input())
max1 = n
while n != 0:
    n = int(input())
    if n>max1:
        max1 = n
print(max1)
```

Егер енгізілген мәндер тек теріс сандар болса, онда программа қандай нәтиже береді?

```
    RESTART:
C:/Users/Asus/AppData/Local/Programs/Python
/Python36-32/Gulira Python/ex2 while.py
    -1
    -4
    -5
    -2
    0
    0
    >>>
```

Программа енгізілген сандардың ең үлкені 0 екенін көрсетіп тұр. Біз 0-ді мәндер енгізудің аяқталғандығын көрсету үшін алып отырмыз. Мұны қалай шешуге болады?

Бұл мәселені шешудің екі тәсілі бар. Бірінші тәсіл, циклдың орындалуын тоқтататын break операторын қосайық. Бұл if операторымен бірге орындалады. Ол сан енгізілгеннен кейін, бірден тексереді. Егер 0 саны кездессе, цикл орындалуын тоқтатып, программаны басқаруды келесі операторға береді.

```
n = int(input())
max1 = n
while n != 0:
    n = int(input())
    if n == 0:
        break
    if n>max1:
        max1 = n
print(max1)
print('All right')
```

```
        RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/Gulira Python/ex2 while.py
        2
        5
        6
        9
        -1
        7
        8
        0
        9
        All right
        >>>
```

Екінші тәсіл. Цикл ішіндегі сан енгізілгеннен кейінгі тұрған шартты күрделі шартқа алмастыру қажет.

```
n = int(input())
max1 = n
while n != 0:
    n = int(input())
    if n != 0 and n>max1:
        max1 = n
print(max1)
print('All right')
```

```
    RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/Gulira Python/ex3 comp cond while.py
    5
    8
    9
    10
    4
    3
    0
    10
    All right
    >>>
```

9-мысал. n элементтен тұратын тізбек берілген. Тізбек элементтерінің қосындысын табатын программа құрайық.

```
s = 0
n = int(input())
while n != 0:
    s = s + n
    n = int(input())
print(s)
print('All right')
```

Программаға өзгеріс енгізейік. Есептегішке ноль емес, енгізілген санды меншіктейік. Алынатын нәтиже алдыңғы программамен бірдей болады, ал циклдың шарты өзгеріссіз қалады.

```
n = int(input())
s = n
while n != 0:
    n = int(input())
    s = s + n
print(s)
print('All right')
```

Программадағы тізбек элементтерінің қосындысын табатын меншіктеу операторын ыңғайлы түрде жазуға болады: `s += n`

Бұл жазу форматы айнымалының ағымдағы мәніне n саны қосылады дегенді білдіреді.

10-мысал. Берілген тізбектегі тек оң элементтерді экранға шығаратын программа мысалын қарастырайық.

```
n = int(input())
while n != 0:
  if n > 0:
    print(n)
  n = int(input())
```

Енді осы мысалды **continue** операторының көмегімен жазып көрейік. Бұл оператор цикл қадамын жаңа мәннен бастап орындайды.

Қайталану параметріне теріс мән – (-1) мәнін меншіктейміз, қайталану параметрінің мәні әзір 0-ге тең емес. Егер қайталау параметрінің мәні 0-ден кіші немесе 0-ге тең болса, онда жалғастыру (**continue**) орындалады. Дәлірек айтқанда, бұл оператор шартты қайта тексеруге жібереді. Егер **break** операторымен салыстырсақ, **break** операторы циклды тоқтатып, программаны басқаруды келесі операторға беретін.

Бұл **continue** операторы циклдің ішіне кіретін операторлардың орындалуын ғана тоқтатып, қайталану шартын тексеруге жібереді.

```
n = -1
while n != 0:
    n = int(input())
    if n <= 0:
      continue
    print(n)
```

```
    RESTART:
C:\Users\Asus\AppData\Local\Programs\Python\Pyth
on36-32\Gulira Python\ex7 continue while.py
    -5
    5
    5
    4
    4
    -1
    0
    >>>
```

11-мысал. Сандарды кему ретімен экранға шығаратын программа құрайық.

```
#сандарды кему ретімен экранға шығару
i = 8
while i > 1:
 i -= 1
 if i == 3:
   continue
 print(i, end="  ")
```

```
RESTART: C:\Ergalym\Project School\phyton\15
МАЙ\Example for book (1)\While  loop\A
level\While-да continue қолдану.py
7  6  5  4  2  1
>>>
```

12-мысал. Көбейту кестесін экранға шығаратын программа құрайық.

```
n=2
while 1:
   i=1;
   while i<=10:
      print("%d X %d = %d\n"%(n,i,n*i));
      i = i+1;
   choice = int(input("Do you want to continue printing the table, press 0 for no?"))
   if choice == 0:
      break;
   n=n+1
```

5.2. «Әзір» (While) қайталану операторын пайдаланудағы тығырықты жағдайлар

While операторын пайдалануда төмендегідей ерекше (тығырықты) жағдайлар болуы мүмкін:

1) егер **While** тақырыбындағы шарт бірінші тексеруден жалған болса, қайталану денесі бірде-бір рет орындалмайды. Бұл программалауда қалыпты жағдай болып есептеледі. Өйткені программада қайталану параметрінің программаның орындалуы барысындағы өзгерісіне байланысты қайталану денесін орындамай, басқаруды келесі операторға беруі қажет болуы мүмкін.

2) егер **While** тақырыбындағы шарт ешқашан жалған мәнге айналмайтын болса, қайталау процесінің барлық кезеңінде ақиқат болып отырса, қайталану процесі ешқашан аяқталмайды. Егер прогаммадағы қайталану шексіз орындалатын болса, бұл процесс программадағы **түйықталу** деп аталады. Бұл жағдайда программа (қатып) өздігінен аяқтала алмайды. Қайталану процесі аяқталу үшін қайталану денесінде циклдан мәжбүрлі түрде қайталану процесін тоқтататын **break** операторының

немесе программадан шығуды жүзеге асыратын **exit** функциясының болуы міндетті.

Сонымен бірге, ерекше (тығырықты) жағдайлар тағы қандай кездерде болуы мүмкін?

Пайдаланушы программада сипатталғанындай, бүтін сан енгізуі тиіс. Программада **input ()** функциясы арқылы енгізілген жол int() функциясының көмегімен бүтін типке түрлендіріледі. Егер сан емес, жолдық символдар енгізілсе **ValueError** деп аталатын ерекше жағдай туындайды, ол программада **except** деген тармақпен өңделеді. Осымен программаның жұмысы аяқталады.

Егер программаға бүтін сан енгізіліп, программа ары қарай санның жұп немесе тақ екенін тексеруі керек еді, бірақ бүтін сан енгізілмеді, нәтижесінде программа жұмысын уақытынан бұрын тоқтатады.

Пайдаланушы дұрыс мән енгізгенше енгізуді талап ететін программа кодын жазайық.

1-мысал. Енгізілген санның жұп немесе тақ екенін анықтайтын программа құрыңыз.

```
n = input("бүтін сан енгізіңіз: ")
while type(n) != int:
    try:
        n = int(n)
    except ValueError:
        print("бүтін сан енгізілген жоқ!")
        n = input("Бүтін сан енгізіңіз: ")

if n % 2 == 0:
    print("жұп сан")
else:
    print("тақ сан")
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/ex while 2.py
бүтін сан енгізіңіз: ert
бүтін сан енгізілген жоқ!
Бүтін сан енгізіңіз: 15
тақ сан
>>>
```

Сонымен, Python тілінде құрама операторларда күрделі құрылым тақырыбының соңына қос нүкте (:) қойылады.

1-мысалдағы **type(n) != int** өрнегінде **type()** функциясының көмегімен **n** айнымалысының типі тексеріледі. Егер ол **type()** функциясына тең болмаса, онда **n** мәні бүтін сан емес, жолдық шама болса, онда өрнектің мәні ақиқат болады. Егер **n** шамасының типі **int()** бүтін болса, логикалық өрнектің мәні жалған болады.

Python тілінде **%** белгісі бүтін санды бүтін санға бөлгендегі қалдықты табу үшін пайдаланылады. Егер сан жұп болса, онда ол 2-ге қалдықсыз бөлінеді де, қалдық нольге тең болады. Егер сан тақ болса, қалдық 1-ге тең.

Енді программа кодының орындалуын талдап көрейік.

Пайдаланушының енгізген мәні жолдық тип, ол **n** айнымалысына меншіктеледі. **whiie** операторының тақырыбында **n** айнымалысының типі тексеріледі. Қайталанудың бастапқы қадамында **n** жолдық тип, бүтін тип емес. Бұдан логикалық өрнектің мәні ақиқат, ары қарай қайталану денесіне өтуге мүмкіндік береді. **try** тармағында жолдық шама бүтін сан типіне түрлендіріледі. Егер түрлендіру сәтті орындалатын болса, **except** тармағы орындалмайды да, оператордың орындалуы қайтадан **while** тақырыбына келеді. Енді **n** бүтін сан, оның типі **int()**. Осылайша **type(n) != int** логикалық өрнегінің мәні жалған болып, қайталану операторы өзінің жұмысын тоқтатады. Ары қарай программаның орындалуы

программаның негізгі тармағында орналасқан **if – else** операторына беріледі.

Мұнда ескеретін тағы бір жағдай бар. Егер **try** операторының денесінде жолды санға түрлендіру сәтті орындалмаса, ерекше жағдай **ValueError** орындалмай, операторлардағы басқару **except** тармағына жіберіледі де, осындағы өрнек орындалады, оның соңғысы пайдаланушыдан мәнді қайта енгізуді талап етеді. Нәтижесінде **n** айнымалысына жаңа мән енгізіледі. **Except** операторы орындалғаннан кейін қайталану тақырыбындағы логикалық өрнек қайтадан тексеріледі, ол қайтадан ақиқат мәнін береді, өйткені **n** айнымалысының мәні жолдық шама.

Сонымен, циклдан шығу **n** айнымалысының мәні бүтін санға сәтті түрлендірілген жағдайда ғана мүмкін болады.

2-мысал. Енгізілген номерге сәйкес келетін Фибоначчи санын экранға шығаратын программа құрайық.

Фибоначчи тізбегінің әрбір мүшесі $U_{n+2} = U_{n+1} + U_n$ заңдылығымен анықталады, тізбектің алғашқы екі мүшесінің мәні: $U_0 = U_1 = 1$ тең.

```
# Фибоначчи есебі
fib1 =fib2 = 1  #фибоначчидің бірінші және екінші элементі

n = input("Фибоначчи қатарының элемент номірі:")
n = int(n)
i = 0
while i < n - 2:
   fib_sum = fib1 + fib2 #нәтижені уақытша сақтау үшін айнымалыға fib_sum тағайындадық
   fib1 = fib2
   fib2 = fib_sum
   print(" ",fib2)
   i = i + 1
print(fib2)
```

```
RESTART: C:\Ergalym\Project School\phyton\15 май
_2 часть\While  loop\C level\фибоначи есебі.py
Фибоначчи қатарының элемент нөмірі:8
        2
        3
        5
        8
        13
        21
        21
>>>
```

5.3. Параметрлі қайталану операторы (for)

Қайталанатын объектінің барлық элементтерімен жұмыс істеуге «үшін» **for** қайталану операторын пайдалануға болады. Оның жазылу форматы:

for <айнымалы атауы> **in** <объектілер тізімі>
оператор

Мұндағы айнымалы атауы қайталану параметрі деп аталады.

Мысалы, color деп аталатын айнымалының мәндері 'red', 'green', 'blue' болсын.

```
for in ('red', 'green', 'blue'):
   print(color,'ball')
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/for1.py
red ball
green ball
blue ball
>>>
```

print(color,'ball') операторы арқылы айнымалының мәндері үлестіріледі.

Сонымен, мәліметтердің аралығы өзара байланысы бар символдар тізімі де болуы мүмкін.

Практикада **for** операторы **range** функциясының көмегімен іріктелетін сандық мәндер үшін және кортеждермен жұмыс істеуде де пайдаланылады.

Мысалы, экранға 1-ден 15-ке дейінгі сандарды басып шығарып көрейік.

```
for k in range(15):
    print(k)
```

Элементтердің номері 0-ден басталатындықтан 15-элемент 14 болып табылады, қайталану параметрінің соңғы мәні кірмейді. Барлық элементтер бір жолға орналасу үшін **end** сөзін пайдалануға болатынын алдыңғы тақырыптарда айтқан болатынбыз:

```
for k in range(15):
    print(k, end=' ')
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/for1.py
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
>>>
```

for қайталану операторындағы қайталану параметрінің бастапқы және соңғы мәнін, қайталау қадамын көрсетуге болады. Мысалы, 1-ден 16-ға дейінгі тақ сандарды экранға шығарып көрейік. Ол үшін қадам санын 2-ге тең деп аламыз.

```
for k in range(1, 16, 2):
    print(k, end=' ')
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/Tuple1.py
1 3 5 7 9 11 13 15
>>>
```

for қайталану операторындағы қадам мәні теріс сан болуы мүмкін. Бұл жағдайда оператордың жазылу форматындағы қайталану параметрінің бастапқы және соңғы мәндерінің орны алмасады және қадам саны теріс таңбалы сан болады. Мысалы, 1-ден 10-ға дейінгі сандарды кері ретпен шығарып көрейік.

```
for k in range(10, -1, -1):
    print(k,end=' ')
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/Gulira Python/for1.py
10 9 8 7 6 5 4 3 2 1 0
>>>
```

Мұнда қайталану параметрінің соңғы мәні -1-ге тең деп алынған, себебі, соңғы мән кірмейді.

1-мысал. N натурал саны берілген. Берілген N санның қосындысын табатын программа құрайық.

```
#Берілген сандарды қосу
N=int(input("N="))
sum=0
for i in range(N):
  a=int(input())
  sum=sum+a
print("sum=",sum)
```

Программаның орындалуы:

```
RESTART: C:\Ergalym\Project School\phyton\15
МАЙ\Example for book (1)\For loop\C lev-
el\Берілген сандарды қосу.py
N=5
5
8
7
6
4
sum= 30
>>>
```

2-мысал. Пифагор көбейту кестесін экранға басып шығаратын программа жазайық.

```
for i in range(1,11):
    for j in range(1,11):
        print(i*j, end='  ')
    print()
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/for1.py
 1   2   3   4   5   6   7   8   9   10
 2   4   6   8  10  12  14  16  18   20
 3   6   9  12  15  18  21  24  27   30
 4   8  12  16  20  24  28  32  36   40
 5  10  15  20  25  30  35  40  45   50
 6  12  18  24  30  36  42  48  54   60
 7  14  21  28  35  42  49  56  63   70
 8  16  24  32  40  48  56  64  72   80
 9  18  27  36  45  54  63  72  81   90
10  20  30  40  50  60  70  80  90  100
>>>
```

Бақылау сұрақтары мен жаттығулар

1. Қайталану операторының программадағы қызметі қандай?

2. WHILE операторында қай кезде циклдің денесі орындалады, қай кезде орындалмайды?

3. WHILE операторын пайдалануда қандай кездейсоқ жағдайлар болуы мүмкін?

4. P-натурал сан болса, a^p есептеуге программа құрыңдар.

5. Төмендегі өрнектердің мәндерін табуға программа құрыңдар:

а) $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}, n \geq 2;$

б) $\frac{1}{1*2}+\frac{1}{2*3}+...+\frac{1}{(n-1)n}, n \geq 2;$

в) $1-\frac{1}{2}-\frac{1}{4}-...+\frac{(-1)^n}{n}, n \geq 2;$

г) $\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)...\left(1+\frac{1}{n^2}\right); n \geq 2;$

6. Берілген санның барлық бөлгіштерін өсу тәртібімен жазуға программа құрыңдар.

7. Берілген n санның ішіндегі оң және теріс сандарының қосындысын, қосындылардың абсолют шамасы бойынша үлкенін табыңдар.

8. Трапеция (немесе тік тортбұрыштар) формуласын пайдаланып, мына интегралдардың мәндерін жуықтап есептеуге программа құрыңдар:

а) $\int_0^n \lg(2+\sin x)dx$

б) $\int_2^7 \lg(2+\sin x)dx$

9. Берілген нақты тізбектің элементтерінің ең үлкені мен ең кішісінің айырмасын табуға программа құрыңдар.

6. ҚОСАЛҚЫ АЛГОРИТМДЕРДІ ПРОГРАММАЛАУ

Практикалық есептерге программа құрғанда үлкен программаның белгілі бір бөлігін әр түрлі мәндер бойынша бірнеше рет қайталап пайдалануға тура келеді. Мұндай бір типтес программа бөлігін программаның әр бөлігінде қайталап жаза бермеу үшін оны жеке қосалқы программа ретінде бөліп жазған ыңғайлы.

Жеке программа түрінде бөлек жазылған, қажет кезінде оған оралып, оны пайдаланып отыруға болатын негізгі программаның арнайы бөлігін қосалқы программа (подпрограмма) дейді. Қосалқы программаны автоматты түрде шақыруға және одан шығуға болады.

Қосалқы программаны пайдалану төмендегідей мүмкіндіктер береді:

- негізгі программаның көлемі кішірейеді;
- негізгі программада пайдаланылған айнымалыларды қосалқы программада да пайдалануға болады;
- қосалқы программаға берілген жады ұяшықтарын ол орындалмай тұрғанда бос ұяшық ретінде (бос айнымалы ретінде) пайдалануға болады;
- қосалқы программаны пайдалану құрылымдық программалауға мүмкіндік береді.

Python программалау тілінде функция типті қосалқы программа пайдаланылады.

Сонымен, функциялар – әртүрлі параметрлер арқылы шақырылатын программаның бір бөлігі болып табылады.

Есеп шығару барысында белгілі бір әрекеттер қайталанып отырады, осындай параметрлері әртүрлі қайталанатын әрекеттерді бірнеше қайталап жазбас үшін программалау тілдерінде оны шағын, жеке программа ретінде жазып, қажетіне қарай шақырып отырады. Сонымен қатар, тіпті бір рет қана пайдаланылатын болса да, функциялар қандай да бір күрделі процесті әртүрлі логикалық бөліктерге бөлу үшін пайдаланылады. Мысалы, үлкен есепті қарапайым есептерге бөліп, олардың әрқайсысына атау беріп, оларды шығару үшін қандай мәліметтерді енгізіп, одан қандай мән алынатынын түсіну

керек. Дәлірек айтқанда, функцияға енгізілетін параметрлер мен функцияның беретін нәтижелерінің форматын анықтау керек. Есеп шығарудың мұндай тәсілін декомпозициялық әдіс деп атайды.

Функцияны құруда ескерілетін бірінші мәселе – ол ұқсалған болуы және басқа есептерді шығаруда пайдаланылатын болуы тиіс. Функцияға атау бергенде ол атқаратын қызметіне сай болуы керек. Сонымен бірге, фикция – программаның колемі шағын және экраннан коруге ыңғайлы болғаны дұрыс.

Біз осы уақытқа дейін дайын функцияларды пайдаланып келдік: print, input, len және math тәрізді библиотекалық функциялар. Бұл функциялар жүйеге орнатылған функциялар деп аталады. Орнатылған функцияларды шартты түрде екі топқа бөлуге болады:

- символдармен жұмыс істеуге арналған – ord(), chr(), len()
- математикалық функциялар – abs(), round(), divmod(), pow(), max(), min(), sum() және т.б. (Бірқатар жүйенің орнатылған функциялары 1.3.-тақырыпта қарастырылған).

ord() функциясы Unicode кестесіндегі символдың номерін береді. Бұл функция аргумент ретінде апострофқа алынған бір ғана символды қабылдайды. Мысалы,

```
>>> ord('m')
109
>>> ord('M')
77
>>> ord('Қ')
1178
>>>  ord('≤')
8804
>>>
```

Бұл функцияға кері мән беретін функция **chr()** деп аталады. Дәлірек айтқанда, символдың кодына сәйкес символды шығарады:

```
>>> chr(77)
'M'
>>> chr(109)
'm'
>>> chr(1178)
'Қ'
>>> chr(8804)
'≤'
```

Бұл аталған функцияларды бірі – бірінен шатастырмау үшін олардың қандай сөздерден қысқартылып алынғанын білу жеткілікті:

- **ord()** функциясы order сөзінен қысқартылып алынған, ол «реті» дегенді білдіреді, ол элементтің реттік номерін алу үшін пайдаланылады;
- **chr()** функциясы character сөзінен қысқартылып алынған, ол символ дегенді сипаттайды, кодқа, санға сәйкес символды алу дегенді білдіреді.

len() функциясының аргументі символдар болғандықтан ол апостроф ішіне алынып жазылады және сол символдардың санын есептеп береді. Мысалы,

```
>>> len('computer')
8
>>> m1 = 'computer'
>>> m2 = 'count'
>>> len(m1) > len(m2)
True
>>> len(s1)
8
>>> len(s2)
5
>>>
```

len() функциясы сандық мәндерді аргумент ретінде қабылдамайды. Мысалы,

```
>>> len('12345')
5
```

Сандар апостроф ішіне алынып жазылғандықтан, оны символдар ретінде қабылдап, оның ұзындығын, дәлірек айтқанда, символдар санын есептеп шығарып береді.

Бұл тақырыпта біз пайдаланушының құратын функцияларын қарастырамыз.

Математикалық фукциялардан python тіліндегі функциялардың айырмашылығы мән беруі міндетті емес, print, input функциялары мән қайтарып, шығармайды, олар белгілі бір әрекеттерді орындайды. Мысалы, print нәтижені экранға басып шығарады. Сондай-ақ, функция параметр қабылдамайды, ал input функциясы мән енгізеді.

6.1. Функциялар. Функцияның анықталуы

Енді пайдаланушының функциясы қалай құрылатынын қарастырайық.

Функцияны анықтау үшін **define** – анықтау сөзінен қысқартылып алынған **def** сөзі пайдаланылады. Одан кейін функцияның аты, дөңгелек жақшаның ішіне функция параметрлері үтір арқылы жазылады. Мысалы, тік бұрышты үшбұрыштың гипотенузасының мәнін екі катеті арқылы есептеуді қарастырайық.

1-мысал. Тік бұрышты үшбұрыштың гипотенузасының ұзындығын оның катеттері арқылы есептеп табатын программа құру.

```
def hhh(a,b):
    ss = a**2+b**2
    return ss**(1/2)
print(hhh(3,4))
```

Мұндағы **hhh** – функцияның аты, ал дөңгелек жақшаның ішіндегі a,b – функция параметрлері, дәлірек айтқанда функцияда есептелетін аргументтер; ss айнымалысына екі катеттің квадраттарының қосындысы есептеліп меншіктеледі. Функцияның есептелген соңғы мәні **return** командасы арқылы беріледі.

Функция қосалқы программа болғандықтан, оны негізгі программада пайдаланғанға дейін анықтау керек. Функцияны анықтағаннан кейін оны программада стандартты функция тәрізді шақыра беруге болады. Функция негізгі программада атауы бойынша шақырылады, функция атынан кейін оның параметрлері міндетті түрде көрсетілуі тиіс. Жоғарыда келтірілген мысалда функция атауы: **hhh(a,b)**. Есептелген функцияның мәні негізгі программада қай жерге шақырылса, функция атауы сол жерге орналасады. Функция программа есепті шығару алгоритмін құрған кезде анықталады және оны негізгі программаның кез келген жеріне орналастыруға болады. Ыңғайлы болуы үшін алдымен функцияларды анықтап, содан кейін негізгі программаны жазуға болады.

2-мысал. Функцияның көмегімен екі санның қосындысын есептейтін программа құрайық.

```
def add_two_nums(a, b):
    c = a + b
    return c
print(add_two_nums(3, 5))
```

Мұндағы add_two_nums – функцияның аты, a және b – функцияның параметрлері, функцияның мәні c айнымалысына меншіктеліп тұр.

Негізгі программада бір ғана оператор орналасқан: print(add_two_nums (3, 5)), ол функцияның параметрлеріндегі көрсетілген мәндер бойынша екі санның қосындысын есептеп, экранға шығарады.

3-мысал. Функцияның көмегімен факториалды есептейтін программа құрайық.

```
def fakt(n):
    f = 1
    while n>1:
        f *=n
        n -=1
    return f
print(fakt(5))
```

Енді осы жазылған функцияны екінші функцияның ішінде қарастырайық.

```
def fakt(n):
    f = 1
    while n>1:
        f *=n
        n -=1
    return f
def teru(n,k):
    return fakt(n)//(fakt(k)*fakt(n-k))
print(teru(5,2))
```

4-мысал. Берілген екі санның үлкенін табатын программа құрайық.

Программа бірінші сан үлкен болса, бірінші санды, кері жағдайда екінші санды экранға басып шығаруы тиіс.

```
def max2(x,y):
    if x>y:
        return x
    else:
        return y
print(max2(3,7))
print(max2(9,7))
print(max2(3,3))
```

Программада мүмкін болатын үш жағдайды тексеру қарастырылған.

```
        RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/exam_max2.py
        7
        9
        3
        >>>
```

Енді алдыңғы программа мысалына өзгеріс енгізейік. Программадан қайталанатын операторлардың бар екенін көруге болады. Атап айтқанда, return операторы екі рет жазылып тұр. Бұл функция мәнді қайтарып бергеннен кейін жұмысын тоқтатады дегенді білдіреді. Егер программадан else сөзін алып тастасақ ешнәрсе өзгермейді, өйткені, егер бірінші сан екінші саннан үлкен болатын болса, return орындалады. Функция ішіндегі одан кейінгі жазылған оператор орындалмайды. Бұл қайталану операторындағы **break** командасына ұқсас. Программа орындалып тұр.

```
def max2(x,y):
   if x>y:
      return x
   return y
print(max2(3,7))
print(max2(9,7))
print(max2(3,3))
```

5-мысал. Үш санның үлкенін табатын программа құрайық.

Ол үшін алдыңғы мысалдағы екі санның үлкенін табатын функцияны пайдаланайық.

```
def max2(x,y):
   if x>y:
      return x
   return y
```

```
def max3(x,y,z):
    return max2(max2(x,y),z)

print(max3(3,7,2))
```

Бұл үш санның үлкенін табатын функция деп аталады. Осы функциядан кейін орналасады және үш санның үлкенін табу функциясының ішінде атауымен (max2(max2(x,y),z)) шақырылады. Оның орындалу нәтижесі:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/exam_max3.py
7
```

Енді функциядан бірнеше мән алатын мысалдар қарастырайық.

6-мысал. Функцияның көмегімен сандарды сұрыптау. Программа берілген екі санды сұрыптап, алдымен кішісін, одан кейін үлкенін экранға басып шығаратын болсын. Бұл функцияны sort2 деп алайық. Егер бірінші сан екінші саннан кіші болса, онда алдымен бірінші санды, одан кейін екінші санды басып шығаратын болсын. Оны шығару үшін return операторынан кейін сандар үтір арқылы жазылады. Функцияның мәнін алу үшін min1 және max1 деп аталатын екі айнымалы енгіземіз де, осы айнымалыларға функциядан алынатын нәтижені меншіктейміз және нәтижеге осы айнымалылардың аттары шығарылады.

```
def sort2(x,y):
    if x<y:
        return x,y
    return y,x
min1,max1 = sort2(9,7)
```

```
print(min1, max1)
```

Программаның орындалу нәтижесінде алдымен кіші сан, одан кейін үлкен сан шығарылады:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/exam_sort2.py
7 9
>>>
```

Функцияның пайдаланылуына келесі мысал – логикалық мәнді қайтарып беру мысалын қарастырайық.

7-мысал. Берілген санның жұп екенін тексеру.

Функцияның атын **Even** деп алайық, функция атынан кейін оның параметрі жазылады. Функция денесінде функцияның жұп екенін тексеру шарты жазылады (сан жұп болу үшін оны 2-ге бөлгендегі қалдық нольге тең болуы тиіс).

Функциядан кейін функция параметрінің мәні енгізіледі. Одан кейін шарт жазылады. Егер **Even** функциясының денесіндегі шарт сақталатын болса, ***even*** сөзін, кері жағдайда ***odd*** сөзін басып шығару қажет.

```
def Even(k):
    return k % 2 == 0
k = int(input())
if Even(k):
    print('even')
else:
    print('Odd')
```

Программаның орындалу нәтижесін екі: жұп және тақ жағдайлар үшін тексеріп көреміз.

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/exam_fun_even.py
15
Odd
>>>
 RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/exam_fun_even.py
12
even
>>>
```

6.2. Функциядағы ортақ және жергілікті айнымалылар

Басқа программалау тілдеріндегі тәрізді Python тілінде функцияда пайдаланылатын айнымалылар ауқымды және жергілікті айнымалылар болып бөлінеді.

Негізгі программада сипатталған (бейненленген) атау, негізгі және қосалқы программада да (функцияларда да) пайдаланыла алады. Мұндай атаулар ортақ (глобальды) айнымалылар деп аталады. Қосалқы программада жарияланған айнымалылар тек сол қосалқы программада ғана пайдаланылады. Мұндай айнымалыларды жергілікті (локальды) айнымалылар деп атайды.

Айнымалыда сақталған мәнді экранға басып шығаратын қарапайым функция мысалын қарастырайық. Функцияның атын **k** деп атайық.

```
def k():
    print(n)
n = 5
k()
```

Программаның орындалу нәтижесінде 5 санын экранға басып шығарады.

Программада пайдаланылған айнымалылар - ортақ (глобальды) айнымалылар. Себебі, бұл айнымалы фунцияда да, негізгі программада да пайдалынылып отыр.

1-мысал. Функцияны пайдаланып,

$$z = \frac{\sin(x) + \cos(y)}{x + y}$$

функциясының мәнін есептейтін программа құру.

Функцияның мәнін есептеу үшін библиотекадан math функциясы шақырылады. Мұндағы x және y айнымалылары ортақ (глобальды) айнымалылар, өйткені ол негізгі программада да, функцияда да пайдаланылып отыр.

```
from math import*
x = int(input("x = "))
y = int(input("y = "))
def function(x,y):
    for i in range(-6, 6):
        z = (sin(x)+cos(y))/(x+y)
    return z
print("rez = %.2f" %(function(x, y)))
```

Шығару операторындағы тырнақшаның ішіндегі символдардың экранға басылып шығатынын білеміз, **%.2f" %** - символдары шығарылатын мәннің нақты екенін және бөлшек болігі 2 орыннан тұратынын білдіреді.

```
RESTART: C:\Ergalym\Project
School\phyton\functions\функция\B level\теңдеуді
шешу функ.py
x = 5
y = 4
rez = -0.18
>>>
```

Сонымен, **ортақ** (глобальды) айнымалылар дегеніміз – функциядан тыс, негізгі программада сипатталған айнымалылар.

Жергілікті айнымалылар тек функцияның ішінде ғана пайдаланылады, ол негізгі программада немесе басқа функцияларда шақырылмайды.

Функцияның параметрлері дегеніміз – жергілікті айнымалылар, ол тек сол функцияның шеңберінде ғана шақырылады.

2-мысал. Үшбұрыштың үш қабырғасының өлшемдері белгілі. Үшбұрыштың периметрі мен ауданын есептейтін программа құрайық.

Үшбұрыштың периметрін есептейтін функция ауданды есептейтін функцияда шақырылады. Есептеулерді жүргізуде түбірді есептеу кездесетіндіктен библиотекадан math import функциясы шақырылады.

```
# үшбұрыштың периметрі мен ауданын табу
from math import*
a = float(input("Бірінші қабырғасы: "))
b = float(input("Екінші қабырғасы: "))
c = float(input("үшінші қабырғасы: "))
def area_of_triangle(a, b, c, p):
    p = p / 2
    s=sqrt(p * (p - a) * (p - b) * (p - c))
    return s
def perimeter_of_triangle(a, b, c):
    p = a + b + c
    return p
perimeter = perimeter_of_triangle(a, b, c)
area = area_of_triangle(a, b, c, perimeter)
print("perimeter of triangle is %.0f" % (perimeter))
print("area of triangle is %.2f" % (area))
```

Программаның орындалуы:

```
RESTART: C:\Ergalym\Project
School\phyton\functions\функция\B lev-
el\үшбұрыштың периметр, ауданын табу функ.py
Бірінші қабырғасы: 9
Екінші қабырғасы: 7
үшінші қабырғасы: 4
perimeter of triangle is 20
area of triangle is 13.42
>>>
```

Python программалау тілінің ерекшеліктерінің бірі – функцияның бірнеше мән шығара алатындығында болып табылады. Ол үшін **return** сөзінен кейін шығарылатын айнымалылар үтір арқылы жазылады.

3-мысал. Цилиндрдің бүйір бетінің ауданы мен толық бетінің ауданын есептейтін программа құрайық.

```
def cylinder():
    r = float(input("r= "))
    h = float(input("h= "))
    side = 2 * 3.14 * r * h
    circle = 3.14 * r**2
    full = side + 2 * circle
    return side, full
sCyl, fCyl = cylinder()
print("Бүйір бетінің ауданы %.2f" % sCyl)
print("Толық бетінің ауданы %.2f" % fCyl)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/functions/buc 2 param.py =====
r= 3
h= 5
Бүйір бетінің ауданы 94.20
Толық бетінің ауданы 150.72
>>>
```

cylinder() функциясы екі мән береді, олардың бірінші – цилиндрдің бүйір бетінің ауданы sCyl айнымалысына, екіншісі – толық бетінің ауданы fCyl айнымалысына меншіктеледі.

4-мысал. Функцияны пайдаланып,

$$C_n^m = \frac{n!}{m!(n-m)!}$$

есептейтін программа құру.

```
# теруді есептеу
m = int(input("m = "))
k = int(input("k = "))

def factorial(x):
    fact = 1
    for i in range(1, x + 1):
        fact = fact * i
    return fact
f = factorial(m)/(factorial(k)*(factorial(m - k)))
print(f)
```

```
RESTART:C:/Ergalym/Project
School/phyton/functions/функция/B level/teru.py
m = 5
k = 3
10.0
>>>
```

5-мысал. Енгізілген күннің сол жылдың қанша-сыншы күні екенін анықтайтын программа құрайық.

Жылдың қаншасыншы күні екенін есептеу үшін Python тілінің уақыт пен күнмен жұмыс істеуге арналған datetime деп аталатын арнайы модульдері библиотекадан шақырылады.

```
#жылдың қаншасыншы күн екенін анықтау
import datetime
a = int(input("Year = "))
b = int(input("Month = "))
c = int(input("Day = "))

def leap_year(a, b, c):
    days = datetime.datetime(a, b, c)
    return days.strftime("%j")

print(leap_year(a, b, c))
```

Программаның орындалуы:

```
RESTART: C:\Ergalym\Project
School\phyton\functions\функция\C level\days.py
Year = 2021
Month = 8
Day = 21
233>>>
```

Datetime модулі арқылы ұсынылатын бірнеше кластар бар. Осы кластардың ішінде күн мен уақыттың комбинациясын беретін класс datetime.datetime деп аталады. Оның форматы:

datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None)

Оның міндетті аргументтеріне:

- **datetime.MINYEAR (1)** ≤ year ≤ **datetime. MAXYEAR (9999)**
- **1** ≤ month ≤ **12**
- **1** ≤ day ≤ **осы жыл мен айдағы күндер саны** жатады.

6-мысал. Енгізілген реттік номеріне сәйкес келетін Фибоначчи санын экранға шығаратын программа құру.

```
# Фибоначчи есебі
n = int(input("n = "))


def fibonacci(n):
    if n in (1, 2):
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)
print(fibonacci(n))
```

```
RESTART: C:\Ergalym\Project
School\phyton\functions\функция\C lev-
el\фибоначи есебі.py
n = 7
13
>>>
```

Программаның орындалуынан реттік саны 7 болатын Фибоначчи тізбегінің мүшесі 13 екенін көреміз.

7-мысал. Берілген бүтін санның цифрларының көбейтіндісін параметрлі фунцияны пайдаланып, табатын программа құрайық.

```
print(' төрт орынды бүтін сан енгізіңіз:')
def sumD(n):
    sumD = 1
    while n!= 0:
        sumD *= n % 10
        n = n // 10
    return sumD
print('санның цифрларының көбейтіндісі
',sumD(int(input())))
```

```
RESTART: C:/Ergalym/Project
School/phyton/functions/digit fun.py ======
 төрт орынды бүтін сан енгізіңіз:
1234
санның цифрларының көбейтіндісі = 24
>>>
```

Бақылау сұрақтары мен жаттығулар

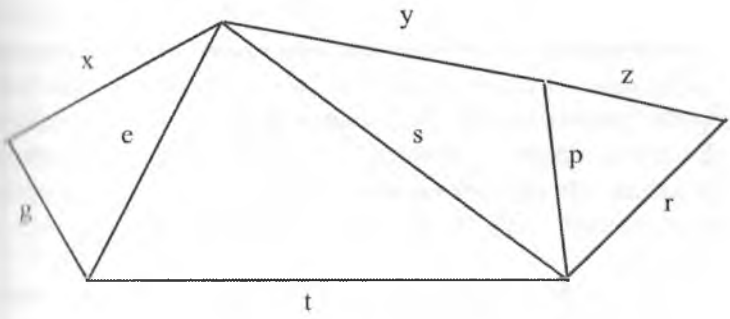
1. Қосалқы программа деп қандай программаны айтады?
2. Функция түрінде жазылған қосалқы программаның ерекшелігіне сипаттама беріңдер.
3. Ауқымды параметрлер мен жергілікті параметрлер арасында қандай айырмашылық бар?
4. Функцияны параметрсіз жазуға бола ма?
5. Функцияны қосалқы программа деп атауға бола ма?
6. Қосалқы программалар қалай шақырылады?

Мына төмендегі есептерді функцияны пайдаланып шешіңдер:

7. $$C_n^m = \frac{n!}{m!(n-m)!}$$

8. $$F = m!+(m-k)!+k!$$

9. Герон формуласы бойынша үшбұрыштың ауданын есептейтін программаны пайдаланып, функция арқылы төмендегі фигураның (6.1-сурет) ауданын есептеңдер.



Сурет 6.1. Дөңес төртбұрыш

Төмендегі функциялардың мәндерінің кестесін құрыңдар:

10. $y = \sin x + \dfrac{3x}{6 + 0{,}3x^2}; \qquad x \in [-3{,}8], n = 20$

11. $y = z \ln x + \dfrac{z(\sin x + \cos x)}{z + \cos x}; \qquad x \in [-1{,}1];$

12. $z = \begin{cases} -x-2, & x<0 \\ e^{x}-4 & x \geq 0 \end{cases}$

13. $z = \dfrac{\sin x + \cos y}{x + y}; \quad x \in [-6{,}6];$

14. $Y = \begin{cases} 2x-6, & x \leq 0 \\ e^{x}-1 & x>0 \end{cases}$

6.3. Рекурсия

Функцияның өзін-өзі шақыруы рекурсия деп аталады. Рекурсияны түсіну үшін екі ұғымды ажыратып алған дұрыс: командалар, программа мәтіні және жергілікті айнымалылардың мәндері туралы ақпарат, дәлірек айтқанда, программа кодының орындалып жатқан орнын, функцияның қай бөлігінде орындалудың тоқтағанын көрсетеді.

Осындай ақпараттардың жиынтығын функция экземпляры деп атайды.

Рекурсияны пайдалануға мысалдар қарастырайық.

1-мысал. Рекурсияны пайдалана отырып, алдымен берілген тізбектегі жұп сандарды, одан кейін тақ сандарды кері ретпен шығаратын программа құрайық.

Программада функция ешқандай параметр қабылдамайды. Функцияда алдымен сандар енгізіледі, әрбір функция аяқталуы тиіс, мысалы, қайталану операторы тәрізді. Оны шартты тексеру арылы жүзеге асыруға болады, егер енгізілген сан нольге тең болмаса, онда функция орындала береді, ал нольге тең болса, ол тізбектің аяқталғандығын білдіреді.

```
def rec():
    n = int(input())
    if n!=0:
        if n % 2 == 0:
            print(n)
        rec()
        if n % 2 != 0:
            print(n)
rec()
```

Программа орындалғанда егер енгізілген сан жұп сан болса, оны бірден экранға басып шығарады, ал тақ сан болса, рекурсиялық функцияның экземпляры **rec()** шақырылады да, тақ сандарды уақытша сақтап қояды. **rec()** функциясын шақырғаннан кейін санды тексеру шарты жазылады, егер шарт сақталатын болса, онда ол экранға кері ретпен шығарылатын болады.

Негізгі программа бір ғана оператордан – функцияның шақырылуынан тұрады: **rec()**

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/exam_fun_recurse.py
9
8
8
7
4
4
3
0
3
7
9
>>>
```

2-мысал. Берілген бүтін санның тақ немесе жұп екенін анықтайтын программаны рекурсияны пайдаланып құрайық.

```
def check(n):
    if (n < 2):
        return (n % 2 == 0)
    return (check(n - 2))
n = int(input("Бүтін сан енгізіңіз:"))
if (check(n) == True):
    print("Жұп сан!")
else:
    print("Тақ сан!")
```

```
RESTART: C:/Ergalym/Project
School/phyton/functions/recursion even.py ===
Бүтін сан енгізіңіз:56
Жұп сан!
>>>
```

Программаның орындалуын қадам бойынша талдап көрейік.

1. Пайдаланушының енгізген саны **n** айнымалысына меншіктеледі.

2. **n** айнымалысына меншіктелген сан check(n) рекурсивті функциясына аргумент ретінде беріледі. Ол программада функцияның анықталуында көрсетіліп тұр.

3. Рекурсияның негізі n < 2 шарты болып табылады. Бұл шарт орындалғанда функция өзінің жұмысын тоқтатады да, ақиқат (True) немесе жалған (False) мәнінің бірін береді.

4. Кері жағдайда, функция n – 2 аргументі бойынша қайтадан шақырылады.

5. Функция өзінің жұмысын аяқтап, нәтиже берген кезде енгізілген санның тақ немесе жұп екенін анықтай аламыз. Егер функция ақиқат (True) мәнін берсе – жұп сан, ал жалған (False) мәнін берсе – тақ сан болады.

6. Нәтиже экранға шығарылады.

3-мысал. Тізімдегі элементтердің қосындысын рекурсияның көмегімен есептейтін программа құрайық.

```
def sum_arr(arr, size):
    if (size == 0):
        return 0
    else:
        return arr[size - 1] + sum_arr(arr, size - 1)
n= int(input("Тізімдегі элементтер санын енгізіңіз:"))
a = []
for i in range(0, n):
    print(i, "элемент")
    element = int(input(" - "))
    a.append(element)
print("Тізімнің барлық элементтері:")
print(a)
print("Тізім элементтерінің қосындысы:")
b = sum_arr(a, n)
print(b)
```

Программның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/functions/recursion number entr.py
Тізімдегі элементтер санын енгізіңіз:5
0 элемент  - 1
1 элемент  - 2
2 элемент  - 3
3 элемент  - 4
4 элемент  - 5
Тізімнің барлық элементтері:
[1, 2, 3, 4, 5]
Тізім элементтерінің қосындысы:
15
>>>
```

Программаның талдануы.

1. Пайдаланушы тізімдегі элементтер санын енгізеді, ол **n** айнымалысына меншіктеледі.

2. Одан кейін пайдаланушы қосындысы есептелінетін тізім элементтерін енгізеді.

3. Енгізілген мәндер **append()** функциясының көмегімен алдын-ала құрылған бос a тізіміне енгізіледі. Көрнекі болу үшін құрылған тізім экранға шығарылады.

4. Енгізілген тізім мен оның ұзындығы sum_arr(arr, size) деп аталатын алдын-ала жазылған рекурсивті функцияға аргумент ретінде беріледі.

5. Тізімнің ұзындығы нольге тең болғанда, функцияның жұмысы аяқталып, өзін-өзі шақыруын тоқтатады.

6. Тізімнің ұзындығы нольге тең болмағандықтан, функция соңғы элемент қосылған қосындыны береді және әрбір шақыруда тізімнің ұзындығы 1-ге кеми береді.

7. Функция жұмысының нәтижесі b айнымалысына меншіктеліп, нәтиже экранға шығарылады.

4-мысал. Рекурсияны пайдаланып, екі санның ең кіші ортақ еселігін табатын программа құрайық.

```
def lcm(a, b):
    lcm.multiple = lcm.multiple + b
    if ((lcm.multiple % a == 0) and (lcm.multiple % b == 
0)):
        return lcm.multiple;
    else:
        lcm(a, b)
    return lcm.multiple
lcm.multiple = 0
a = int(input("Бірінші санды енгізіңіз:"))
b = int(input("Екінші санды енгізіңіз:"))
if (a > b):
    LCM = lcm(b, a)
else:
    LCM = lcm(a, b)
print(a," және  ", b," сандарының ЕКОЕ:")
print(LCM)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/functions/НОК.py ==========
Бірінші санды енгізіңіз:15
Екінші санды енгізіңіз:25
15  және   25  сандарының ЕКОЕ:
75
>>>
```

Программаның талдануы:

1. Пайдаланушы екі бүтін сан енгізеді, ол сандар **a** және **b** деген айнымалыларға меншіктеледі.
2. **lcm.multiple** деген айнымалыға ноль меншіктейміз.

3. Енгізілген санның қайсысы үлкен екені тексеріліп, осу реті бойынша lcm() рекурсивті функциясына беріледі.

4. **lcm()** функциясының бірінші қадамында айнымалының шамасы пайдаланушының енгізген санының ең үлкенінің шамасына артады. Дәлірек айтқанда, функцияның бірінші шақырылуында айнымалының шамасы сол санға тең болады.

5. Одан кейін **lcm.multiple** айнымалысының мәні енгізілген екі санға қалдықсыз бөлінетіндігі тексеріледі. Егер қалдықсыз бөлінетін болса, функция өзінің жұмысын тоқтатып, **lcm.multiple** айнымалысының мәнін нәтиже ретінде шығарады.

6. Егер сандар қалдықсыз бөлінбейтін болса, **lcm()** функциясы қайтадан шақырылып, **lcm.multiple** айнымалысының мәні пайдаланушының енгізген санының ең үлкенінің шамасына қайтадан арттырылады. Осы процесс (шарт) екі санға қалдықсыз бөлінгенше жалғаса береді.

7. Функция жұмысын аяқтағаннан кейін ең кіші ортақ еселіктің мәні экранға шығарылады.

5-мысал. Енгізілген бүтін санды ондық санау жүйесінен екілік санау жүйесіне аударатын программа құрайық.

```
l = []
def conv(b):
    if (b == 0):
        return l
    dig = b % 2
    l.append(dig)
    conv(b // 2)
a = int(input("Бүтін сан енгізіңіз: "))
conv(a)
l.reverse()
print(a," санының екілік жүйедегі түрі:")
for i in l:
    print(i,end=' ')
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/functions/конвертация чисел.py ==
Бүтін сан енгізіңіз: 10
10  санының екілік жүйедегі түрі:
1 0 1 0
>>>
```

Программаның талдануы:

1. Алдымен **l** айнымалысын бос тізім ретінде құрамыз.

2. Аргументі сан болып табылатын **conv()** деп аталатын рекурсивті функция құрамыз.

3. Функцияның өзінде рекурсияның негізгі шарты анықталады. Дәлірек айтқанда, функция аргументі нольге тең. Бұл жағдайда функция рекурсивті түрде қалыптасқан **l** тізімді береді.

4. Кері жағдайда, санның әрбір цифры екілік жүйеге аударылып, тізімге 0 немесе 1 түрінде қосылады. Функция қайтадан шақырылады, алғашқы аргументті 2-ге бөлгеннен алынған мән оның аргументі ретінде қабылданады.

5. Одан кейін **reverse()** функцияның көмегімен тізім элементтері кері ретпен орналасады да, **for** операторының көмегімен экранға шығарылады.

6-мысал. Рекурсияны пайдаланып, кез келген бүтін санды бүтін санға дәрежелейтін программа құрайық.

```
def ppp(base, e):
    if (e == 1):
        return (base)
    if (e != 1):
        return (base * ppp(base, e - 1))
base = int(input("Бүтін сан енгізіңіз: "))
e = int(input("дәреже көрсеткішін енгізіңіз: "))
print(base, "санының ",e, "дәрежесі:  ", ppp(base, e))
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/functions/степень.py =======
Бүтін сан енгізіңіз: 7
дәреже көрсеткішін енгізіңіз: 3
7 санының  3 дәрежесі:   343
>>>
```

Программаның талдануы:

1. Алдымен сан мен оның дәреже көрсеткіші енгізіліп, олар **base** және **e** айнымалыларына сәйкес меншіктеледі.

2. Бұл айнымалалар **ppp()** рекурсивті функцнясына аргумент ретінде беріледі.

3. Рекурсияның негізгі шартын анықтаймыз, ол **e == 1** тең. Бұл жағдайда функция **base** айнымалысының мәні санның өзін шығарады, өйткені кез келген санның 1 дәрежесі сол санның өзіне тең.

4. Кері жағдайда, аргументі 1-ге кеміген функция мен санның көбейтіндісі шығарылады: **base * ppp(base, e - 1)**. Осылайша **base** айнымалысына **e** рет санның көйбейтіндісі қосыла береді, ол **base** санын **e** рет дәрежелеумен бірдей.

5. Нәтижені экранға шығарамыз.

7-мысал. Рекурсияны пайдаланып, екі санның ең үлкен ортақ бөлгішін табатын программа құрайық.

```
def eyob(a, b):
    if (b == 0):
        return a
    else:
        return eyob(b, a % b)
a = int(input("Бірінші санды енгізіңіз:"))
b = int(input("Екінші санды енгізіңіз:"))
EYOB = eyob(a, b)
print(a," және ", b, "сандарының EYOB: ")
print(EYOB)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/functions/NOD.py ========
Бірінші санды енгізіңіз:27
Екінші санды енгізіңіз:15
27  және  15 сандарының EYOB:
3
>>>
```

Программаның талдануы:

1. Пайдаланушы екі бүтін сан енгізеді, бұл сандар a және b айнымалыларына меншіктеледі.

2. Одан кейін бұл сандар eyob() рекурсивті функциясына аргумент ретінде беріледі.

3. Рекурсияның негізгі шарты ретінде екінші аргументтің мәнін нольге тең (b == 0) деп аламыз. Бұл жағдайда функция жұмысының нәтижесі бірінші аргумент a болады.

4. Кері жағдайда рекурсивті функцияны қайтадан eyob(b, a % b) түрінде шақырамыз. Дәлірек айтқанда, бірінші аргумент ретінде оған функцияның алдыңғы шақырылуындағы екінші аргументті (b) береміз, ал екінші аргумент – a-ны b-ға бөлгендегі қалдық болады.

5. Функция жұмысын аяқтағанда оның нәтижесі осы функцияның соңғы шақырылуындағы бірінші аргумент болады және ол екі санның ең үлкен ортақ болгіші болып табылады.

6. Функция жұмысының нәтижесін экранға шығарамыз.

Бақылау сұрақтары мен жаттығулар

1. Рекурсия деп нені түсінеміз?
2. Функция экземпляры дегеніміз не?
3. Программадағы рекурсиялық функцияның қызметі қандай?

4. Рекурсиялық функцияның қызметін мысалмен түсіндіріңіз.

5. Рекурсияны пайдаланып, берілген бүтін санның жай сан екенін анықтайтын программа құрыңдар.

6. Берілген екі санның көбейтіндісін табатын программаны рекурсияны пайдалана отырып, құрыңдар.

7. Жолдық элементтерді кері ретпен шығаратын программаны рекурсияны пайдалана отырып құрыңдар.

8. Жолдың ұзындығын есептейтін программаны рекурсияны пайдалана отырып құрыңдар.

9. Рекурсияны пайдаланып, сандарды бір жүйеден екінші жүйеге аударатын программа құрыңдар.

10. Рекурсияны пайдаланып, Фибоначчи қатарының алғашқы 9 мүшесін экранға шығаратын программа құрыңдар.

7. ЖОЛДАР

Жолдар дегеніміз – бұл тырнақшаға алынған символдар тізбегі. Python программалау тілінде символдық тип жоқ, дәлірек айтқанда, объектісі жеке символ болып табылатын мәліметтер типі. Дегенмен, Python тілі жолдарды бір немесе одан да көп символдардан тұратын объектілер ретінде қарастыруға мүмкіндік береді. Жолдардың тізімнен айырмашылығы ол құрылымданған мәліметтерге жатпайды. Мәліметтер құрылымы қарапайым мәліметтер типтерінен құралады, ал жолдар қарапайым символдардан құралады. Жолдық элементтер тізім элементтері тәрізді реттелген. Сондықтан жолдық элементтерден жеке символдарды іздеп тауып, қию әрекеттерін орындауға болады.

7.1. Жолдарға пайдаланылатын негізгі операциялар

Бізге s[i] жолы берілген. s[i] жолының бірінші символы – s, жолды номерлеу 0-ден басталады. Python тілінің басқа программалау тілдерінен ерекшелігі - жолдарды оңнан солға қарай теріс сандармен де номерлеуге болады.

0	1	2	3	4	5
s	**t**	**r**	**i**	**n**	**g**
-6	-5	-4	-3	-2	-1

Жолдық элементтерді екі параметрмен қиып алғанда оң жақ шегі кірмейді. Мысалы,

s[2:5] = ‘rin’.

Оң және теріс индекстерді аралас пайдалануға болады. Мысалы,

s[1:-1] = ‘trin’

Екі параметрмен қиып алудың тағы бір тәсілі бар. S айнымалысының мәні **abcdef** болсын.

Print(s[1:]). Мұның нәтижесі **bcdef** болады.

Жұп номермен тұрған барлық символдарды шығару үшін бастапқы және соңғы символдың номерін көрсетпей, тек қадамын корсетеміз:

Print(s[::2])

Нәтижесінде жұп номерлі символдарды аламыз:

Ace

а символының номері 0 және ол жұп номерге жатады.

Барлық тақ номерлі сиволдарды шығару үшін

Print(s[1::2])

bcf

Len функциясы жолдағы символдардың санын немесе жолдың ұзындығын есептеуге арналған. Мысалы,

```
s = 'computer internet'
print(len(s))
```

Программаның орындалуы нәтижесінде 17 санын аламыз. Бұл символдар санының 17 екенін көрсетеді (бос орын бір символ болып есептеледі).

Жолдармен жұмыс істеуге арналған бірнеше әдістер бар. Әдіс дегеніміз нақты бір объектіге, оның мазмұнына қарай бекітілген функциялар болып табылады.

Қандай да бір объектінің құрылымын сипаттағанда ол туралы мәліметтер сақталады, осы мәліметтерді қандай да бір әдістердің көмегімен өңдеуге (жазуға, оқуға және түрлендіруге) болады.

Жолдарға пайдаланылатын әдістерді қарастырайық.

Жолды тұрақты ретінде сипаттайық.

Жолдың ішінен қандай да бір бөлігінің орнын іздеуді қарастырайық. Ол үшін **find()** әдісі пайдаланылады. Мысалы,

```
s = 'computer'
print(s.find('put'))
```

Нәтижесінде 3 деген жауап аладық. Бұл оның 3-орыннан басталатынын көрсетеді. Айнымалының атын көрсетіп, жанына нүкте қойғанда сол объектіге

пайдалануға болатын барлық әдістердің тізімі шығады. Мысалы, берілген мәтіннің ішінен ter сөзінің қай орыннан басталатынын табатын программа жазайық. Ол үшін орындар санын есептейтін есептегішке, дәлірек айтқанда, қайталану параметріне (pos = 0) ноль меншіктейміз.

Тексеру цикл (қайталану) командасы арқылы жүзеге асырылады. Циклдың шартына s айнымалысынан табылған 'ter' сөзінің орны -1-ге тең болмаса қайталану жалғаса беретіндей шарт қоямыз және қайталану параметрінің мәнін 1-ге арттырамыз.

```
s = 'computer internet'
pos = 0
while s.find('ter',pos) != -1:
    print (s.find('ter',pos))
    pos = s.find('ter',pos) + 1
```

Программа орындалғанда 5 және 11 деген жауап аламыз. Бұл ізделінді шаманың 5-ші және 11-ші орыннан басталатынын көрсетеді.

```
#тізім элементтерін өзгерту
Rainbow = ['Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Indigo', 'Violet']
print(Rainbow[0])
Rainbow[0] = 'қызыл'
Rainbow[3] = 'жасыл'
print('кемпірқосақты шығарыңыз:')
for i in range(len(Rainbow)):
  print(Rainbow[i])
```

2-мысал. Клавиатурадан енгізілген мәтінді 3 рет қайталап, диалог жасайтын программа құрайық.

```
# Жолдардың қайталануы
s=str(input('Кез келген мәтін енгізіңіз:'))
R=(s + '    ')*3   #S-ке меншіктелген мәтінді 3 рет қайталап шығарады
```

```
D='How are you?'
L='Hi'+','+R+'! '+D
print(L)
```

Программаның орындалуы:

```
RESTART: C:\Ergalym\Project School\phyton\
жолдардың қайталануы.py

Кез келген мәтін енгізіңіз:I love Python
Hi,I love Python   I love Python   I love Python   !
How are you?
>>>
```

3-мысал. Клавиатурадан енгізілген мәтіндерді жалғастыратын программа құрайық.

```
# конкатенация (қосу)
L=input('Бірінші мәтін енгізіңіз:')
S=input('Екінші мәтін енгізіңіз:')
R=L+'  '+S+'   біздікі'
print(R)
```

```
RESTART: C:\Ergalym\Project
School\phyton\конкатенация (қосу).py

Бірінші мәтін енгізіңіз:Kolsay
Екінші мәтін енгізіңіз:lake
Kolsay  lake   біздікі
>>>
```

4-мысал. Берілген создің әрбір символын және сөздің бірінші бөлігі мен екінші бөлігін қиып алатын программа құрайық.

```
# Кесінділерді табу(Срезы)
s='python'
```

```
print(" бірінші әріп: ", s[0])
print(" екінші әріп: ",s[1])
print(" үшінші әріп: ",s[2])
print(" төртінші әріп: ",s[3])
print(" бесінші әріп: ",s[4])
print(" алтыншы әріп: ",s[5])
print(" сөздің бірінші бөлігі: ",s[:3])
print(" сөздің екінші бөлігі: ",s[3:])
```

```
RESTART: C:/Ergalym/Project School/phyton/my
Срезы.py

бірінші әріп:  p
екінші әріп:  y
үшінші әріп:  t
төртінші әріп:  h
бесінші әріп:  o
алтыншы әріп:  n
сөздің бірінші бөлігі:  pyt
сөздің екінші бөлігі:  hon
>>>
```

5-мысал. S_1 және S_2 бүтін сандары берілген ($S_1 < S_2 < 1000$). Берілген аралықтағы сандарға сәйкес символдарды және А..Я аралығындағы символдардың кодын экранға шығаратын прогрaмма құрайық.

```
# Санға сәйкес символдарды және керісінше кодын анықтау
S1=int(input('Бастапқы санды енгізіңіз:'))
Sn=int(input('Соңғы санды енгізіңіз:'))
print('Берілген сандардың сәйкес символдары:')
for i in range(S1,Sn):
    S=chr(i)
    print(str(S),end=' ')
print('\n Келесі А-дан Я-ға дейінгі символдардың коды: ')
```

```
for i in range(ord('А'),ord('Я')):
    print(i, end=' ')
```

```
RESTART: C:\Ergalym\Project School\phyton\str(s),
ord(n)  сан-әріп.py
Бастапқы санды енгізіңіз:100
Соңғы санды енгізіңіз:150
Берілген сандардың сәйкес символдары:
d e f g h i j k l m n o p q r s t u v w x y z { | } ~ □
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
Келесі А-дан Я-ға дейінгі символдардың коды:
1040 1041 1042 1043 1044 1045 1046 1047 1048
1049 1050 1051 1052 1053 1054 1055 1056 1057
1058 1059 1060 1061 1062 1063 1064 1065 1066
1067 1068 1069 1070
>>>
```

6-мысал. N натурал саны берілген. N санының жазылуынан 0 мен 5 цифрларын алып тастап, қалған цифрлардың орнын өзгертпей жазатын программа құрайық. Мысалы, 5090105060 санынан 916 саны алынуы тиіс.

```
#жолдың элементтерін өшіру
s = input('сандар енгізіңіз::')
s1=s.replace('5', '')
s1=s1.replace('0','')
print("Енгізілген сан:",s)
print("Нәтиже: ",s1)
```

```
RESTART: C:\Ergalym\Project School\phyton\жол
элементтерін өшіру.py

сандар енгізіңіз::985756084550110
Енгізілген сан: 985756084550110
Нәтиже:  98768411
>>>
```

7-мысал. 100-ге дейінгі символдардан тұратын латын тексіндегі дауысты дыбыстардың санын анықтайтын программа құрайық.

```
vowel = ['a', 'e', 'i', 'o', 'u']
Sentence = input("Enter a sentence: ")
count = 0
for letter in Sentence:
   if letter in vowel:
      count += 1
print(count)
```

```
RESTART: C:/Ergalym/Project School/phyton/count
str.py

Enter a phrase: Almaty is the beautiful city in the
world
12
>>>
```

7.2. Жолдарға пайдаланылатын функциялар мен әдістер

Python программалау тілінде жолдар өзгермейтін тізбектер категориясына жататын болғандықтан, барлық функциялар мен әдістер тек жаңа жол құрады. Тілдің жүйесіне орнатылған функциялар мен әдістерді: жолдың регистрін өзгерту, жолдың бөлігін іздеу және оны алмастыру, жолдарды жіктеу, жолдарды туралау, жолдарды тізімдерге түрлендіруге арналған деп бөлуге болады.

Жолдарға пайдаланылатын функциялар мен әдістердің бірқатарына мысалдар келтірейік.

s.isidentifier() - жолдың Python тілінің идентификаторы екенін анықтайды. Егер **s** жолы Python тілінің идентификаторы (айнымалы аты, функция, класс

және т.б.) болса, ақиқат мәнін, кері жағдайда жалған мәнін береді.

```
>>> 'print'.isidentifier()
True
>>> '3201жоқ'.isidentifier()
False
```

S.lower() - жолды төменгі регистрге көшіреді.

```
>>> 'КОМПЬЮТЕР МЕНІҢ ЖАҚСЫ
ДОСЫМ'.lower()
'компьютер менің жақсы досым'
>>>
```

S.isdigit() – жолдың цифрлардан тұратынын анықтайды, егер жол бос болмаса және цифрлардан тұратын болса, ақиқат мәнін береді, кері жағдайда жалған мәнін береді.

```
>>> '1973'.isdigit()
True
>>>
>>> '1973 computer'.isdigit()
False
```

Жолдың бөлігін шаблонмен алмастыруды қарастырайық. Оның форматы:

S.replace(шаблон, алмастыру)

Төмендегі мысалда «ұнатпаймын» сөзі «ұнатамын сөзіне алмасуы тиіс.

```
>>> 'Мен программалауды ұнатпаймын! Мен
программалауды
ұнатпаймын!'.replace('ұнатпаймын', 'ұнатамын')
'Мен программалауды ұнатамын! Мен
программалауды ұнатамын!'
>>>
```

Мұндағы шаблон деп отырғанымыз, мәтін ішіндегі кездесетін сөз, ал «алмастыру» шаблонды алмастыратын жаңа сөз.

S.zfill(width) функциясын қарастырайық. Мұндағы width айнымалысы жалпы символдар санын көрсетеді. Мысалы,

```
>>> '+65'.zfill(6)
'+00065'
>>>
```

Егер жолдың алдында таңба болатын болса, онда ол сол жақта қалады. Мысалдан көріп отырғанымыздай, 65 санының алдына үш ноль орналастырды.

Жолды толтыруға арналған келесі функция

S.ljust(width, fillchar=" ")

Мұндағы **l (left)** солға қарай дегенді білдіреді, дәлірек айтқанда, берілген символдар сол жақта қалады, ал қойылатын символдар оң жақта орналасады.

```
'Well done'.ljust(15, '!')
'Well done!!!!!!'
>>>
```

Көріп отырғанымыздай, 15 саны символдардың саны, 'Well done' тоғыз орын алып отыр, сондықтан алты леп белгісі қойылды.

S.strip([chars]) – жолдың оң және сол жақ шетіндегі символдарды өшіреді. **S.strip**() әдісі **S.lstrip**([chars]) және **S.rstrip**([chars]) шақыруымен бара – бар. Chars әдісі символдың басындағы және соңындағы бос орындарды өшіреді.

Мысалы,

```
>>>'  My best friend is a computer      '.strip()
'My best friend is a computer'
>>>
```

1-мысал. Берілген жолдағы әріптің кездесетін орнын анықтау қажет. Атап айтқанда, бастапқы және соңғы орнының индексін табатын программа құрайық. Ол үшін екі параметрлі функция жазу қажет, ол ізделінді символды және тұтас жолды қамтиды. Жолда әріп кездеспеген жағдайда кортежді (None, None) беру керек, егер жолда ізделінді символ бар болса, онда оның бастапқы және соңғы индексін экранға басып шығару керек.

```
def first_last(letter, st):
    first = st.find(letter)
    if first < 0:
        return None, None
    last = st.rfind(letter)
    return first, last
# Тексеру
print(first_last('a', 'Алматы қаласы'))
print(first_last('a', 'Айша – Бибі мазары'))
print(first_last('a', 'алпамыс'))
print(first_last('a', 'Бойы бұлғын'))
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/str3.py ==================
(3, 10)
(3, 15)
(3, 3)
(None, None)
>>>
```

2-мысал. Ұсынылған мәтін үзіндісіндегі ең жиі кездесетін үш символды анықтайтын программа құрайық. Есептеу барысында бос орындарды ескермеу қажет. Есептеу нәтижелерін шығару функция арқылы жүзеге асырылсын. Функция нәтижені жолдар түрінде шығарсын: символ – саны, символ – саны және т.с.с.

```
from collections import Counter
def top3(st):
    counter_top3 = Counter(st.replace(' ', '')).most_common(3)
    return ', '.join([f'{letter} - {cnt}' for letter, cnt in counter_top3])
# Тексеру
print(top3('Желсіз түнде жарық ай, сәулесі суда дірілдеп, ауылдың маңы терең сай, тасыған өзен гүрілдеп'))
print(top3('Алматы'))
print(top3(мен ойланбадым'))
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/str1.py ==================
е - 8, а - 8, с - 6
А - 1, л - 1, м - 1
м - 2, н - 2, а - 2
>>>
```

3-мысал. Мәтін берілген. Мәтіннің ішінде жақшаға алынған бөліктері бар. Осы жақшаға алынған мәтіннің бөліктерін өшіріп, жақша сыртындағы мәтінді ғана экранға шығаратын программа құрайық. Программада функция пайдаланылсын.

```
def shortener(st):
    while '(' in st or ')' in st:
        left = st.rfind('(')
        right = st.find(')', left)
        st = st.replace(st[left:right + 1], '')
    return st
# Тексеру
print(shortener('Биыл жаз(көзді ашып жұмғанша (кірпік) қаққандай болып) жылдам (лезде)өте шықты'))
print(shortener('(Әлдекімге(белгісіз)болып түр)Қыстың (алай-түлей(бөраны(бар ма)))қандай болары беймәлім'))
```

```
RESTART: C:/Users/Asus/Desktop/python
prog/str2.py ==================
Биыл жаз жылдам өте шықты
Қыстың  қандай болары беймәлім
>>>
```

4-мысал. Ішінде артық сиволдары бар мәтін бөлігі берілген. Артық символдарды өшіргенде жүйеге кіретін құпия сөз табылады. Жолдық функциялар мен әдістерді пайдалана отырып, сөздегі артық символдарды өшіріп, құпия сөзді ашатын программа құрайық.

```
def cleaned_str(st):
    clean_lst = []
    for symbol in st:
        if symbol == '0' and clean_lst:
            clean_lst.pop()
        elif symbol != '0':
```

```
            clean_lst.append(symbol)
    return ''.join(clean_lst)
# тексеру
print(cleaned_str('ком0пт0ьюю0тер'))
print(cleaned_str('мамакин00000шитош000на'))
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/str4.py ==========
компьютер
машина
>>>
```

7.1-кестеде осы аталған әрекеттерді қамтитын жолдарға пайдаланылатын функциялар мен әдістер келтірілген.

Кесте 7.1.

Жолдарға пайдаланылатын функцнялар мен әдістер

Функциялар мен әдістер	Қызметі
S = 'str'; S = "str"; S = ''"str"''; S = ""'"str"'"	литералдар
S = "s\np\ta\nbbb"	Экрандаға тізбектер
S = r"C:\temp\new"	Форматталмаған жолдар
S = b"byte"	Байттар жолы
S1 + S2	Жолдарды жалғастыру
S1 * 3	Жолдарды қайталап шығару
S[i]	Жолдың индексін көрсету
S[i:j:step]	Жолдық элементтерді қию
len(S)	Жолдың ұзындығы

Кесте 7.1.

Жолдарға пайдаланылатын функциялар мен әдістер

(жалғасы)

Функциялар мен әдістер	Қызметі
S.find(str, [start],[end])	Жолдың ішкі бөлігін іздеу. Алғашқы табылған номерді көрсетеді
S.rfind(str, [start],[end])	Жолдың ішкі бөлігін іздеу. Соңғы табылған номерді көрсетеді
S.index(str, [start],[end])	Жолдың ішкі бөлігін іздеу. Алғашқы кіру номерін көрсетеді немесе ValueError қателігін береді
S.rindex(str, [start],[end])	Жолдың ішкі бөлігін іздеу. Алғашқы кіру номерін көрсетеді немесе ValueError қателігін береді.
S.replace(шаблон, алмастыру)	Шаблонмен алмастыру
S.split(символ)	Ажырату белгісі бойынша жолдарды бөлу
S.isdigit()	Жолдың цифрлардан тұратынын тексеру
S.isalpha()	Жолдың әріптерден тұратынын тексеру
S.isalnum()	Жолдың цифрлардан немесе әріптерден тұратынын тексеру

Кесте 7.1.

Жолдарға пайдаланылатын функциялар мен әдістер

(жалғасы)

Функциялар мен әдістер	Қызметі
S.islower()	Жолдың төменгі регистр символдарынан тұратынын тексеру
S.isupper()	Жолдың жоғарғы регистр символдарынан тұратынын тексеру
S.isspace()	Жолдың көрінбейтін символдардан тұратынын тексеру (бос орын, жаңа бетке көшіру (‘\f’), келесі жолға көшіру (‘\n’), каретканы көшіру (‘\r’), горизонталь табуляция (‘\t’), вертикаль табуляция (‘\v’)
S.istitle()	Жолдың бас әріптен басталатынын тексеру
S.upper()	Жолды жоғарғы регистрге кошіру
S.lower()	Жолды төменгі регистрге көшіру
S.startswith(str)	S жолы str шаблонынан басталатынын тексеру
S.endswith(str)	S жолының str шаблонымен аяқталатынын тексеру
S.join("5)"&()	S бөлгіші бойынша тізімнен жол құру

Кесте 7.1.

Жолдарға пайдаланылатын функциялар мен әдістер

(жалғасы)

Функциялар мен әдістер	Қызметі
ord(")+'&3)	Символдың ASCII жүйесіндегі кодын береді
chr(8)"3&)	ASCII жүйсіндегі кодқа сәйкес символды береді.
S.capitalize()	Жолдың тек бірінші символын жоғаргы регистрге алмастырады, ал қалған стиволдары төменгі регистрде болады.
S.center(width, [fill])	Шетінде fill символы болатын жолды ортаға туралайды.
S.count(str, [start],[end])	Басы және соңы аралығындағы қайталанбайтын жолдың ішкі бөлігінің кездесу санын береді.
S.expandtabs([tabsize])	Табуляцияның барлық символдары бір немесе бірнеше бос орындармен алмасатын жолдың көшірмесін береді. Ол ағымдағы бағанға тәуелді. Егер TabSize олшемі көрсетілмесе, табуляция өлшемі 8 бос орынға тең.

Кесте 7.1.

Жолдарға пайдаланылатын функциялар мен әдістер

(жалғасы)

Функциялар мен әдістер	Қызметі
S.lstrip([chars])	Жолдың басындағы бос орындарды өшіру
S.rstrip([chars])	Жолдың соңындағы бос орындарды өшіру
S.strip([chars])	Жолдың басындағы және соңындағы бөс орындарды өшіру
S.rpartition(sep)	Бірінші шаблонның алдындағы бөлікті кортежге айналдырады, одан кейін шаблонның өзі, одан кейін қалған бөлігі орналасады. Егер шаблон табылмаса, жолдың өзінен және екі бос жолдан тұратын кортежді береді
S.swapcase()	Төменгі регистр символдарын жоғарғы регистрге, ал жоғарғы регистр символдарын төменгі регистрге алмастырады
S.title()	Әрбір сөздің тек бірінші әрібін ғана жоғарғы регистрге көшіріп, қалғандарын өзгеріссіз қалдырады.

Кесте 7.1.

Жолдарға пайдаланылатын функциялар мен әдістер

(жалғасы)

Функциялар мен әдістер	Қызметі
S.zfill(width)	Қажеттілігіне қарай алғашқы символдарды нольмен толтыра отырып, жолдың ұзындығын width параметріне теңестіреді (осы параметрден кем болмайды)
S.ljust(width, fillchar=" ")	Қажеттілігіне қарай соңғы символдарды fillchar символдарымен толтыра отырып, жолдың ұзындығын width параметріне теңестіреді (осы параметрден кем болмайды)
S.rjust(width, fillchar =" ")	Қажеттілігіне қарай алғашқы символдарды fillchar символдарымен толтыра отырып, жолдың ұзындығын width параметріне теңестіреді (осы параметрден кем болмайды)
s.isidentifier()	Жолдың Python тілінің идентификаторы екенін анықтайды.
S.format(*args, **kwargs)	Жолдарды форматтау

Туралау символдарды толықтырғыштардың көмегімен жүзеге асырылады. Туралаудың төмендегідей нұсқалары бар.

Кесте 7.2.

Символдарды толықтырғыштардың көмегімен туралау

Белгі	**Мәні**
'<'	Толықтауыш символдар тек оң жақтан қойылады (объект сол жаққа қарай тураланады)
'>'	объект оң жаққа қарай тураланады
'='	Толықтауыш тек таңбадан кейін, тек цифрдың алдынан қойылады. Тек сандық типтер үшін орындалады
'^'	Объект ортасына қарай тураланады

Кесте 7.3.

Format әдісінің көмегімен форматтау

Типі	**Мәні**
'd', 'i', 'u'	Ондық сан
'o'	Сегіздік санау жүйесіндегі сан
'x'	Он алтылық санау жүйесіндегі сан (төменгі регистр әріптері)
'X'	Он алтылық санау жүйесіндегі сан (жоғарғы регистр әріптері)
'e'	Экспоненциалды жылжымалы үтірлі сан (төменгі регистрдегі экспонента)
'E'	Экспоненциалды жылжымалы үтірлі сан (жоғарғы регистрдегі экспонента)
'f', 'F'	Жылжымалы үтірлі сан (әдеттегі формат)
'g'	Экспоненциалды жылжымалы үтірлі сан (егер ол -4-тен кіші болса, төменгі

	регистрдегі экспонента, кері жағдайда әдеттегі формат)
‘G’	Экспоненциалды жылжымалы үтірлі сан (егер ол -4-тен кіші болса, жогарғы регистрдегі экспонента, кері жағдайда әдеттегі формат)
‘c’	Символ (бір символдан немесе саннан тұратын жол – символ коды)
‘s’	Жол
‘%’	Сан 100-ге көбейтіледі, сан жылжымалы үтір арқылы бейнеленеді, одан кейін % белгісі қойылады.

Таңбалар тек сандарға пайдаланылуы мүмкін және оның төмендегідей нұсқалары бар:

Кесте 7.4.

Таңбалардың пайдаланылуы

Белгі	**Мәні**
‘+’	Таңба барлық сандар үшін пайдаланылады
‘-‘	Таңба тек теріс сандарға пайдаланылады
Бос орын	‘-‘ – тек теріс сандар үшін, бос орын – оң сандар үшін

Python тілінде экранға шығарылатын символдар тізбегін басқару мүмкіндігі бар.

Кесте 7.5.

Экранға шығарылатын символдар тізбегін басқару

Экрандағы символдардың орналасуын басқару	Қызметі
\n	Курсорды келесі жолға көшіру
\a	Қоңырау
\b	Арнайы белгі қою
\f	Келесі бетке көшіру
\r	Келесі жолға өту
\t	Көлденең табуляция
\v	Вертикаль табуляция
\N{id}	Юникод мәліметтер қорының ID идентификаторы
\uhhhh	16-лық түрдегі 16-биттік Юникод символы
\Uhhhh. . .	32-лік түрдегі 32-биттік Юникод символы
\xhh	Символдың 16-лық мәні
\ooo	Символдың 8-дік мәні
\0	Null символы (жолдың соңын білдіретін белгі болып табылмайды)

Бақылау сұрақтары мен жаттығулар

1. Python тілінде жолдар деп нені түсінеміз?
2. Жолдарға пайдаланылатын негізгі операциялар қандай?
3. Жұп номермен тұрған барлық символдарды шығаруға бола ма?
4. Жолдарға пайдаланылатын қандай негізгі әдістер бар?

5. Жолдарға пайдаланылатын әдіс пен функцияның айырмашылығы неде?

6. Жолдық әдістер қандай жағдайларда пайдаланылады?

7. Python программалау тілінде пайдаланылатын барлық функциялар мен әдістердің ерекшелігі неде?

8. Жолдың ішкі бөлігінің орнын қалай анықтауға болады?

9. Жолдарды туралау қалай жүзеге асырылады?

10. Жолда символдың қанша рет кездесетіні қалай анықталады?

11. Жолдың ішкі бөлігін іздеу қалай жүзеге асырылады?

12. Жолдың басындағы және соңындағы символдарды іздеуге бола ма?

13. Жолды тізімге қалай түрлендіруге болады?

14. Жолдағы бос орындарды қалай өшіруге болады?

15. Жолға ноль немесе жаңа символ қосуға бола ма?

16. Символдық берілгендерді енгізудің қандай ерекшеліктері бар?

17. Шығару операторындағы форматтың қызметі қандай?

18. Формат көрсетілмесе нәтиже қалай беріледі?

19. Символдық мәндер қалай енгізіледі?

20. Кез келген мәтін берілген, ал мәтіннің соңын Enter белгілейді. Енгізілген символдың санын және мәтіндегі Т әрпінің санын есептеуге программа құрыңдар.

21. Коды 1-ден 255-ке дейінгі сәйкес символдарды шығаруға, A-дан Z-ке дейінгі латынның бас әріптерінің кодын анықтауға программа құрыңдар.

22. 100-ге дейінгі символдардан тұратын латын тіліндегі берілген мәтіндегі дауысты дыбыстардың санын анықтайтын программа құрыңдар.

23. Латын әріптерімен мәтін берілген. Латынның кіші әріптерін оған сәйкес үлкен әріпке аударатын программа құрыңдар.

24. Count() функциясын пайдаланбай, берілген жолдағы жолдың ішкі бөлігін іздейтін программа құрыңдар.

25. Жолдардан құрылған массив берілген. Массивтегі ең ұзын жолды анықтап, оны нөмерімен (индексімен) экранға шығаратын проғрамма құрыңдар. Егер ең ұзын жол бірнешеу болатын болса, олардың бәрінің индексін экранға шығару керек.

26. Жолдан бос орындарды өшіріп, оның палиндром болып табылатынын анықтау керек. (Палиндром дегеніміз – оңнан солға қарай және солдан оңға қарай бірдей оқылатын сөз немесе фраза болып табылады).

27. Әріптер мен бос орындардан тұратын жол берілген. Осы әріптерді пайдалана отырып, әртүрлі ұзындықтағы бірнеше сөз тіркесін (сөз) алатын программа құрыңдар. Әріптерді бірнеше рет пайдалануға болады.

28. Берілген жолдағы қайталанатын символдар мен бос орындарды өшіретін программа құрыңдар.

29. Енгізілген жолдағы бас әріптермен кіші әріптердің санын есептейтін программа құрыңдар. (Мәтін ағылшын тілінде болуы тиіс).

30. Енгізілген санның идентификатор екенін, яғни сөз жоғарғы немесе төменгі регистрдегі ағылшын әрібінен бастала ма, әлде астын сызу таңбасынан тұра ма, ағылшын алфавиті немесе цифрлардан басқа символдарды қамтылмайтынын анықтайтын программа құрыңдар.

31. N натурал саны берілген. N санының жазылуынан 3 санының еселіктері болатын цифрларды алып тастап, қалған цифрлардың орнын өзгертпей жазатын программа құрыңдар. Мысалы, 5309615960 санынан 50150 саны алынуы тиіс.

32. Интеллектуалдық ойында Азаматтың жинаған ұпайларының санын көрсететін мәтін берілген. Жинаған ұпайларының жалпы санын есептеп, оны экранға шығаратын программа құрыңдар.

8. КОРТЕЖДЕР

8.1. Кортеж элементтерін құру және олармен жұмыс істеу

Тұрақты шамаларды сақтауға арналған объект кортеж деп аталады. Кортежде сақталған элементтер индекстеледі, дәлірек айтқанда, номерленеді. Номерлеу жолдық элементтер номері тәрізді нольден басталады.

Кортеж элементтеріне қандай операциялар қолдануға болады?

Бірінші қолданылатын операция жалғастыру.

Алдымен екі кортежді сипаттап, оған жалғастыру операциясын қолданайық.

```
Tuple1 = (7, 8 ,9)
Tuple2 = (1, 2, 3)
Tuple3 = ('a','b','c')
print(Tuple1 + Tuple2)
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/Tuple1.py
(7, 8, 9, 1, 2, 3)
```

Кортеж элементтерінің санын **len** функциясын пайдалана отырып, есептеуге болады:

```
Tuple1 = (7, 8 ,9)
print(len(Tuple1))
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/Tuple1.py
3
```

Бұл бірінші кортеждегі элементтер санының 3-ке тең екені көрсетеді.

Сонымен қатар, бір кортеждің ішінде әртүрлі типті мәліметтер орналаса алады. Мысалы, кортеждің элементтері бүтін сандардан, нақты сандардан және символдар тізімінен тұратын болсын. Осындай элементтерден тұратын кортеждің мазмұнын экранға шығарып көрейік.

```
Tuple1 = (7, 8 ,9 , 5.3, -6.5, 'computer')
print(Tuple1)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/Tuple1.py
(7, 8, 9, 5.3, -6.5, 'computer')
```

Кортеждің элементтері кортеж болуы мүмкін. Төмендегі кортеждегі 4 санының индексін былай сипаттап жазамыз:

```
Tuple4 = (5, (4,3),(6,))
print(Tuple4 [1][0])
```

Кортеж элементтерін индекстеу (номерлеу) 0-ден басталатынын айттық. Оның номері квадрат жақшаның ішіне алынып жазылады, алдымен сырткы индекс, одан кейін ішкі индекс квадрат жақша арқылы көрсетіледі. Біздің мысалда индексі 1 болатын элемент кортеж – оның индексі 1, осы кортеждің 0-ші элементі – 4.

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/Tuple1.py
4
```

Сонымен, кортеждер көбінесе объектілерді сипаттау үшін пайдаланылады. Кортеж элементтері ретінде кестедегі бір жолдың элементтерін алуға болады.

Мысалы, кестенің бір жолында студенттің аты-жөні, телефоны, туған жылы туралы мәліметтер орналасуы мүмкін.

Кортеж элементтерін сипаттағанда үтір арқылы жазылған элементтердің оң жағында арифметикалық операциялар пайдаланылмаған жағдайда жақшаны жазбауға болады.

Кортеждерге қолданылатын келесі операция – кортеж элементтерін жинақтау және ашу деп аталады.

Үтір арқылы жазылған элементтер бір объектіге жинақталады. Мұны кортежді жинақтау (упаковка) деп атайды. Бұл кортеж элементтерін құру. Сонымен, кортеж элементтерін жинақтау үшін кортеж атауынан кейін меншіктеу белгісін қойып, үтір арқылы кортеж элементтерін жазамыз. Осыған кері әрекет кортеж элменттерін ашу (распаковка) деп аталады, дәлірек айтқанда, кортеждің әрбір элементтерін жеке айнымалыларға меншіктеп басып, шығаруға және операциялар қолдануға болады. Мысалы, Tuple5 деп аталатын кортежге үш элемент жинақталған. Осы элементтерді жеке айнымалыларға меншіктеп, олардың екеуіне қосу амалын қолданып, нәтижені экранға шығарып көрейік.

Tuple5 = 11,12,15	Жинақтау (құру)
a,b,c = Tuple5	ашу
print(a + c)	Ашылған элементтерге амал қолдану

Оның орындалу нәтижесі:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/Tuple1.py
26
```

Кортеж элементтерін жіктегенде қанша элемент болса, сонша айнылмалы үтір арқылы жазылуы тиіс, дәлірек айтқанда, элементтер санына айнымалылар саны тең болуы тиіс.

Кортежбен жұмыс істегенде алдымен элементтер құрылады, одан кейін оны ашуға болады. Кортеж элменттерін құрғаннан кейін оған қолданылатын операциялардың бірі - алмастыру, айнымалылардың мәндерін алмастыруға болады.

```
l,m,n = 10,15,16
print(l,m,n)
l,m,n = n,m,n
print(n,m,l)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/Tuple1.py
10 15 16
16 15 16
```

Кортеждерге пайдаланылатын бірқатар функциялар бар. Мысалы, белгілі бір аралықтағы сан мәндерін оқу үшін **range** деп аталатын функцияны пайдалануға болады. Ол үшін шығару операторында оның кортеж екенін **tuple** сөзі арқылы көрсету қажет (**tuple** – өзара байланысқан шамалар жиынтығы деген мағынаны білдіреді).

```
ran1 = range(10)
print(tuple(ran1))
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python
36-32/Gulira Python/Tuple1.py
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Мұндағы **tuple** функциясының негізгі қызметі – алдымен бастапқы мәнді алып, одан кейінгі мәндерді бірден қабылдап отырады. Бұл жерде қайталану операциясының бар екенін көруге болады. Қайталанудың әрбір қадамында қандай әрекет орындалады? Ағымдағы мәнді алғаннан кейін қайталану ережесіне сай, келесі объектіге өтеді. Осылайша **tuple** функциясы барлық объектіні бірінен кейін бірін шығарып отырады. Дәл осы процесті жолдық объектілер үшін де орындауға болады.

1-мысал. Кездейсоқ бүтін сандардан тұратын кортеж құратын программа жазайық.

```
# 0-ден 10 аралығындағы бүтін сандардан тұратын
# 5 элементті қамтитын кортеж құру
# Кездейсоқ сандар гееараторын шақыру үшін
#  random модулін қосамыз
import random
# 1. Сандардан тұратын тізім құру
lst = [] # алдымен бос тізім құрамыз
i = 0
while i < 5:
   num = random.randint(0,10)
   lst = lst + [num]
   i = i+1
# 2. Тізім элементтерінен кортеж құрамыз.
a = tuple(lst)
print("a = ", a) # a = (3, 0, 2, 6, 2)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/tuple 1.py ==========
a =  (3, 1, 10, 4, 8)
```

2-мысал. Жолдардан тұратын кортеж берілген. Берілген элементті іздеп табатын программа құрайық.

```
# Берілген элементті іздеу
# 1. Жолдардан тұратын кортеж берілген
a = ('ab', 'abcd', 'cde', 'abc', 'def')
# 2. Ізделінді жолды енгізу
s = str(input("s = "))
# 3. Циклды пайдалану
result = False # нәтиже
i = 0
while i < len(a): # len(a) – кортеждегі элементтер саны
   if (a[i]==s):
      print('ізделінді элемент: ', a[i])
      result = True
       break
   i = i+1
if (result):
   print("Yes")
else:
   print("No")
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/tuple2.py ==============
s = abc
ізделінді элемент:  abc
Yes
>>>
```

3-мысал. Бүтін сандардан құрылған кортеж элементтерін өсу ретімен функция арқылы сұрыпталатын программа жазайық. Егер кортеждің бір элементі бүтін сан болмаса, бастапқы кортеж өзгеріссіз шығарылсын.

```
def tpl_sort(tpl):
   for element in tpl:
      if not isinstance(element, int):
         return tpl
   return tuple(sorted(tpl))
# Тексеру
```

```
print(tpl_sort((7, 5, 6, 3, 10)))
print(tpl_sort((3, 3, 1.1, 'a', 11)))
```

```
RESTART: C:/Users/Asus/Desktop/python
prog/tuple4.py ==============
(3, 5, 6, 7, 10)
(3, 3, 1.1, 'a', 11)
>>>
```

Бақылау сұрақтары мен жаттығулар

1. Кортеж деп нені түсінеміз?
2. Кортежді пайдаланудың маңыздылығы қандай?
3. Кортеждің қандай қасиеттері бар?
4. Тізім мен кортеждің арасындағы айырмашылықтары қандай?
5. Тізім мен кортеждің арасындағы ұқсастықтары қандай?
6. Кортеж құрудың қандай тәсілдері бар?
7. Кортеж элементтерінің санын қалай есептеуге болады?
8. Кортеж элементтеріне қандай операциялар қолдануға болады?
9. Бүтін сандардан құрылған кортеж элементтерін кері ретиен орналастыратын функция – программа жазыңдар.
10. 10 атаулы студенттер кортежінен тұратын кортеж құрыңдар. Атаулы кортеждің төмендегідей өрістері болуы тиіс: студенттің аты-жөні, курс, семестрдегі бағасы, оқу орны. Функция осы кортежді қабылдап, студенттердің орташа бағасын есептеп, тиісті хабарлама шығаруы тиіс.

9. ТІЗІМДЕР

Басқа программалау тілдеріндегі «массив» деген атпен танымал мәліметтер типі Python тілінде **тізім** деп аталады. Массив дегеніміз не? Бір атауға біріктірілген айнымалылардың реттелген тізімін массив деп атайтынбыз. Python тілінде **тізім** дегеніміз атауға біріктірілген айнымалылардың реттелген тізбегі. Реттелген деген сөз элементтердің реттік санының бар екенін білдіреді, дәлірек айтқанда, әрбір элементтің реттік номері, индексі бар.

Тұрақты элементтерен тұратын тізімді қалай құруға болады?

Айнымалы атына квадрат жақшаға алынған мәндер меншіктеледі. Мұнда квадрат жақшаны қалдырып кетуге болмайды. Егер жақшасыз жазылса, оны жүйе кортеж деп түсінуі мүмкін.

Тізімнің жазылу форматы:

$a = [a_1, a_2, \ldots, a_n]$

Мұндағы a – тізімнің аты, ал a_1, a_2, …, a_n – тізімнің (массив) элементтері.

Бір тізімнің элементтерін екінші тізімге беру үшін тізім атауына екінші тізімнің атауын меншіктеу жеткілікті. Мысалы, 3 элементтен тұратын a тізімі берілген. a тізімінің элементтерін b тізіміне меншіктеп, b тізімінің элементтерін экранға басып шығарайық.

```
a = [4,5,6]
b = a
print('b=',b)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list1.py
b= [4, 5, 6]
>>>
```

Тізім элементтерімен жұмыс істеу элементтің реттік номері арқылы жүзеге асырылады. Үшінші тұрған элементті 1 санымен алмастырайық. Ол үшін 3-элементтің реттік номерін анықтаймыз, номерлеу 0-ден басталатындықтан 3-элементтің реттік саны 2. Олай болса, тізімнің реттік саны 2 болатын элементіне 1-ді меншіктейміз және ол екінші тізімде өзгеретінін немесе өзгермейтінін тексерейік.

```
a = [4,5,6]
b = a
a[2] = 1
print('b=',b)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list1.py
        b= [4, 5, 1]
>>>
```

Бұдан b тізімінің элементтерінің де өзгергенін байқауға болады. b тізімінің элементтерін өзгертпей қалдыруға бола ма? Ол үшін екінші b тізімі жаңа объект ретінде құрылуы тиіс және a тізімін меншіктегенде «қию» (:) операциясын қою керек, сонда бастапқы a тізімінің элементтері меншіктеледі де, одан кейінгі өзгерістер b тізіміне енгізілмейді.

```
a = [4,5,6]
b = a [:]
a[2] = 1
print('b = ',b)
print('a = ',a)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python
36-32/Gulira Python/list1.py
b =  [4, 5, 6]
a =  [4, 5, 1]
>>>
```

Сонымен, бізде екі тізім объектісі бар, олардың біреуіне өзгеріс енгізуге болады, екіншісіне болмайды.

Тізімнің элементтері әртүрлі мәліметтер типінен құрылуы мүмкін. Мысалы,

```
c = [2, (1,(2, 'abc',3)),5, (6,7)]
print(c)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python36-32/Gulira Python/list1.py
[2, (1, (2, 'abc', 3)), 5, (6, 7)]
>>>
```

Бұдан тізімнің элементтері ретінде кортеж элементтерін алуға болатынын көруге болады.

Тізімнің элементтерін тағы да қандай жолдармен алмастыруға болады?

Функцияны пайдаланып, тізімнің реттік саны 3 болатын элементін 0-мен алмастырайық.

```
def rp(a):
    a[2] = 0
a = [4,5,6,7,8]
rp(a)
print(a)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python36-32/Gulira Python/list fun.py
[4, 5, 0, 7, 8]
>>>
```

Бұдан функцияда объектіге сілтеме параметр арқылы берілетінін көруге болады. Сонымен, тізім өзгеретін объектіге жатады.

9.1. Тізімдерге пайдаланылатын әдістер

Алдымен, жолдардан қалай тізім құруға болатынын қарастырайық. Ол үшін тізім құруға арналған **list** деп аталатын функция пайдаланылады.

```
a = list('computer')
print(a)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list2.py
['c', 'o', 'm', 'p', 'u', 't', 'e', 'r']
>>>
```

Жолдан тізім құрып, оған қандай да бір өзгерістер енгізіп, одан кейін жолды қайтадан жинауға болады. Ол үшін жолдармен жұмыс істеуге арналған **join** (ағылшын тілінен аударғанда *біріктіру*) әдісін пайдаланамыз. Бұл функцияның қызметі жолдық элементтерді бір-бірімен жалғастырып шығарады. Алдыңғы мысалдағы 'computer' сөзіне the артиклін қосып шығарайық.

```
a = list(' computer')
a[0] = 'the '
s = ' '.join(a)
print(s)
```

Реттік саны 0-ші а тізіміне **the** артиклін меншіктейміз де, ал 'computer' сөзінің алдына – 0-ші орынға - бос орын қалдырамыз. s жолына бос орын (апостроф) қойып, join функциясын жазамыз. Программаның орындалу нәтижесі:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list2.py
the computer
>>>
```

Мысалдан көріп отырғанымыздай, 'the' сөзі a тізіміндегі бос орынға, дәлірек айтқанда, 0-орынға барып орналасты.

Практикалық есептерде сөздер тізімін оқуға тура келеді, ал сөздер бос орындармен бөлінген. Бұл жағдайда жолдарға пайдаланылатын **split** әдісін пайдалануға болады. Аталған әдіс бос орынсыз жазылған сөздерден тізім құрайды.

```
w = input().split()
print(w)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/list split.py
my computer is          best
['my', 'computer', 'is', 'best']
>>>
```

Программаның орындалуынан енгізілген бос орындар қалдырылып, тек символдардан құралған сөздердің алынғанын көруге болады.

split әдісін параметрмен де пайдалануға болады. Мысалы, енгізілген символдардың ішінен «о» әрібін қиып алып тастау қажет болсын.

```
w = input().split('o')
print(w)
```

Программаның орындалу нәтижесінен «о» әрібі қиылған жерден олардың элементтерге жіктелетінін көруге болады.

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/list split.py
component
['c', 'mp', 'nent']
>>>
```

Біз сөздерден құралған тізіммен қалай жұмыс істеуге болатынын көрдік. Енді сандардан құралған тізіммен жұмыс істеуді қарастырайық. Бір жолға жазылған сандар берілсін. Осы сандарды тізім элементтері ретінде шығарып көрейік.

Мұны **map** функциясы арқылы орындауға болады, ол тізімнің әрбір объектісіне қандай да бір функцияны пайдалануға мүмкіндік береді. Сонымен қатар, тізім элементтері сандар болуына байланысты **int** функциясын және сандар тізімін алу қажет болғандықтан **list** функциясын пайдаланамыз, ол объектінің элементтерін бір-бірлеп алып, содан тізім құрады. Сандарды енгізгенде арасы бос орынмен ажыратылуы тиіс.

```
num = list(map(int, input().split()))
print(num)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list number  split.py
1 2 3 8 7 9
[1, 2, 3, 8, 7, 9]
>>>
```

Сандарды қалай экранға басып шығаруға болады?

Алынған тізім элементтерін экранға үтірсіз басып шығару үшін **join** әдісін пайдалануға болады, бірақ дәл қазір тізім элементтері сандар, сондықтан **str** функциясын қосамыз және ол барлық элементтер үшін орындалуы қажет, сол себепті **map** функциясын бірге пайдаланамыз.

```
num = list(map(int, input().split()))
print(' '.join(map(str,num)))
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list number  split.py
7 8 9 45 2
7 8 9 45 2
>>>
```

Әдетте тізім элементтерін экранға басып шығару үшін **print(*num)** арқылы шығаруға да болады.

9.2. Тізім элементтерін өңдеу

Тізім элементтерін өңдеу үшін оны енгізу қажет. Ол енгізу операторы арқылы жүзеге асырылады, сонымен бірге, бірқатар қосымша функциялар пайдаланылады. Біз алдыңғы тақырыпта қарастырғанымыздай, тізім элементтерін енгізу:

a = list(map(int, input().split()))

input() - жолды оқу үшін;
split() - жолды сөздерге бөлу үшін;
map() - осы тізімге **int** функциясын пайдалану, дәлірек айтқанда, әрбір сөз санға түрлендіріледі;
list() - осы итерацияланатын объектіден тізім алу үшін пайдаланылады.

1-мысал. Берілген тізімдегі тек жұп элементтерді экранға шығаратын программа құрайық.

```
a = list(map(int, input().split()))
i = 0
while i <len(a):
   if a[i] % 2 != 0:
      a.pop(i)
   else:
      i +=1
print(*a)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list 3.py
1 5 4 2 7 6
4 2 6
>>>
```

Қайталану параметрінің **i** бастапқы мәніне 0 меншіктейміз, өйткені тізім элементтерінің реттік саны 0-ден басталады. **Әзір** қайталану параметрі тізім элементтерінің ұзындығынан кіші болса, онда тізімнің ағымдағы элементін 2-ге болгендегі қалдық 0-ге тең болмаса, сол элемент өшіріледі. Әйтпесе қайталану параметрінің қадам саны 1-ге арттырылады. Осы әрекеттер қайталану операторының ішінде орындалады.

Ұсынылып отырған программада **pop()** функцнясы тізімде қанша элемент болса, сонша рет шақырылады да, программа баяу жұмыс істейді.

Бұдан да тиімдірек тәсілді қарастырайық.

2-тәсіл.

```
a = list(map(int, input().split()))
b = []
for n in a:
    if n % 2 == 0:
        b.append(n)
print(*b)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/list even.py
7 8 9 5 4 6
8 4 6
>>>
```

Бұл нұсқада жаңа **b** деген бос тізім құрамыз, осы тізімге тек қажетті элементтерді салып отырамыз. Енді «**үшін**» қайталану операторын пайдаланамыз: қайталану параметрі **n** тізімнің ағымдағы элементін қабылдайды, ол **a** тізіміне кіретін болса, онда шартты тексереміз. Егер тізімнің ағымдағы элементін 2-ге бөлгендегі қалдық 0-ге тең болса, онда сол элементті **b** тізіміне тіркейміз. Қайталану аяқталғаннан кейін b тізімінің элементтерін шығарамыз.

9.3. Тізім элементтерін сұрыптау

Тізім элементтерін сұрыптауға арналған арнайы әдістер бар. Сол әдісердің бірі sort() деп аталады. Ол үшін алдымен тізім құру қажет. Біз тізім құрудың екі әдісін қарастырдық. Тізім элементтерін тұрақты ретінде енгізу:

a = [9,1,7,3,5,2,6]

немесе

клавиатурадан енгізу:

a = list(map(int, input().split()))

```
a = [9,1,7,3,5,2,6]
a.sort()
print(*a)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list sort1.py
1 2 3 5 6 7 9
>>> >
```

Бұдан тізім элементтерінің өсу ретімен сұрыпталғанын көруге болады.

Екінші әдіс тізімнің сұрыпталған көшірмесін құрады. Ол үшін **sorted()** функциясы пайдаланылады.

```
a = [9,1,7,3,5,2,6]
b = sorted(a)
print('b= ',*b)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list sort1.py
b= 1 2 3 5 6 7 9
>>> >
```

sorted() функциясының нәтижесі бастапқы енгізілгендерге тәуелсіз барлық уақытта тізім болады.

Біз тізім элементтерін өсу ретімен қалай сұрыптауға болатынын көрдік. Енді тізімнің элементтерін кему ретімен қалай сұрыптауға болады?

```
a = [9,1,7,3,5,2,6]
b = sorted(a, reverse=True)
print('b= ',*b)
```

Бұл мысалда **reverse** функциясы кері шаманы алу үшін пайдаланылады, біздің жағдайда тізім элементтерін өсу реті бойынша емес, керісінше, кему реті бойынша шығару үшін оған ақиқат (**True**) мәнін меншіктейміз.

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list sort1.py
b=  9 7 6 5 3 2 1
>>>
```

Тек қарапайым объектілерді ғана емес, күрделі объектілерді де сұрыптауға болады. Алдымен екі кортеждің элементтерін салыстырып, нәтижесін экранға шығарып көрейік.

```
print((1,2)<(3,4))
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list sort2.py
True
>>>
```

Программаның орындалуынан нәтиженің ақиқат екенін көруге болады.

Салыстыру элементтердің реттік номері бойынша жүргізіледі.

1-мысал. Массив элементтерін клавиатурадан енгізіп, кему реті бойынша сқрыптайтын программа құрайық.

```
n = int(input("n = "))
a = [ int(input('Введите число A[%d] = '%(i+1))) for i in range(n)]
a = sorted(a, reverse = True)
print(a)
```

2-мысал. Массив элементтерін кездейсоқ сандар генераторы арқылы енгізе отырып, «көпіршікті» сұрыптау әдісімен массив элементтерін сұрыптайтын программа құайық.

```
from random import randint
 N = 10
a = []
for i in range(N):
   a.append(randint(1, 99))
print(a)
for i in range(N-1):
   for j in range(N-i-1):
      if a[j] > a[j+1]:
         a[j], a[j+1] = a[j+1], a[j]
print(a)
```

Бұл мысалда кездейсоқ сандар генераторын шақыру **randint** модулін импорттау арқылы жүзеге асырылды.

```
RESTART: C:/Users/Asus/Desktop/python
prog/сортировка пузыр.py ========
[93, 28, 51, 67, 52, 12, 41, 98, 68, 97]
[12, 28, 41, 51, 52, 67, 68, 93, 97, 98]
>>>
```

9.4. Екі өлшемді массивтер (тізімдер)

9.4.1. Екі өлшемді массивтерді (тізімдерді) енгізу және өңдеу

Практикада тікбұрышты кесте түрінде берілген мәліметтерді сақтап, өңдеу жұмыстары жиі кездеседі. Мұндай кестелерді матрица немесе екі өлшемді массивтер деп атайды. Python программалау тілінде кесте түрінде берілген мәліметтерді жолдар тізімі түрінде ұсынуға болады, мұнда әрбір элемент өз кезегінде жол болып табылады. Енді екі өлшемді массив элементтерімен жұмыс істеуге арналған мысалдар қарастырайық.

1-мысал. Екі жол, үш бағаннан тұратын сандық кесте құратын программа құрайық.

```
a = [[1, 2, 3], [4, 5, 6]]
print(a[0])
print(a[1])
b = a[0]
print(b)

print(a[0][2])
a[0][1] = 7
print(a)
print(b)
b[2] = 9
print(a[0])
print(b)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/2mas.py ==
[1, 2, 3]
[4, 5, 6]
[1, 2, 3]
3
[[1, 7, 3], [4, 5, 6]]
[1, 7, 3]
[1, 7, 9]
[1, 7, 9]
>>>
```

Мұндағы тізімнің бірінші жолы a[0] – бұл [1, 2, 3] сандарынан тұратын тізім болып табылады. Дәлірек айтқанда,

a[0][0]==1
a[0][1]==2
a[0][2]==3

a[1][0]==4
a[1][1]==5
a[1][2]==6

Тізім элементтерін енгізіп, өңдеу үшін бірінің ішіне бірі орналасқан екі цикл пайдаланылады. Бірінші цикл жол номерімен, екінші цикл баған номерімен жұмыс істеуге арналған.

2-мысал. Екі өлшемді сандық тізімді экранға жолмен шығаратын программа құрайық (1-тәсіл).

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for i in range(len(a)):
    for j in range(len(a[i])):
        print(a[i][j], end=' ')
    print()
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/2mas.py ==
1 2 3 4
5 6
7 8 9
>>>
```

Python тілінде for параметрлі қайталану командасы range функциясының көмегімен құрылатын аралықпен ғана емес, кез келген тізбектің кез келген элементін таңдауға мүмкіндік береді. Python тіліндегі тізбек – тізімдер, жолдар және т.б. кейбір объектілер болуы мүмкін.

Ернді **for** параметрлі қайталану командасының қасиеттерін пайдаланып, екі өлшемді массив элементтерін шығаруды қарастырайық.

2-мысал. Екі өлшемді сандық тізімді экранға жолмен шығаратын программа құрайық (2-тәсіл).

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for row in a:
    for elem in row:
        print(elem, end=' ')
    print()
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/2mas.py ==
1 2 3 4
5 6
7 8 9
>>>
```

3-мысал. Екі өлшемді сандық тізімдегі барлық элементтердің қосындысын табатын программа құрайық (1-тәсіл).

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j]
print('Екі өлшемді массив элементтерінің қосындысы:
',s)
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/2mas.py ==
Екі өлшемді массив элементтерінің қосындысы:  45
>>>
```

Осы тізім элементтерінің қосындысын табуды индекс бойынша емес, жолдың мәні бойынша да есептеуге болады.

3-мысал. Екі олшемді сандық тізімдегі барлық элементтердің қосындысын табатын программа құрайық (2-тәсіл).

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    for elem in row:
        s += elem
print('Екі өлшемді массив элементтерінің қосындысы:
',s)
```

Екі өлшемді массив элементтерін енгізу. Біз массив элементтерін енгізу мысалдарын келтірдік. Сонымен, екі өлшемді массив (квадрат матрица) элементтерін енгізудің бірнеше жолы бар екенін байқадық. Массив элементтері

клавиатурдан енгізілгенде, алдымен, жол және баған элементтернің саны енгізіледі. Одан кейін, жол және баған элементтері үшін екі бос тізім құрылады. Енгізілген элементтер append() функциясының көмегімен көрсетілген тізімге тіркеледі. Мысалы,

```
n = int(input("n = "))
m = int(input("m = "))
a = []
for i in range(n):
    b = []
    for i in range(m):
        b.append(int(input()))
    a.append(b)
```

Екі өлшемді массив элементтеріне қолданылатын барлық операциялар бірінің ішіне бірі орналасқан циклдар арқылы жүзеге асырылады. Мысалы, екі өлшемді массивтегі ең үлкен элементті іздеу:

```
max1 = a[0][0]
for i in range(n):
    for j in range(m):
        if a[i][j] > max1:
            max1 = a[i][j]
```

Екі өлшемді массив элементтерін шығару. Екі олшемді квадрат матрица элементтерін шығару да бірінің ішіне бірі орналасқан циклдармен орындалады. Екі олшемді массив элементтерін матрица түрінде шығару үшін **end** функциясы пайдаланылады.

```
for i in range(n):
    for j in range(n):
        print(a[i][j], end=' ')
    print()
```

Енді мысалдар келтірейік.

4-мысал. Бүтін сандардан құралған екі өлшемді массив берілген. Масивтегі ең үлкен элементті тауып, оны жол және баған номерімен экранға шығаратын программа құрайық.

```
n = int(input("n = "))
```

```
m = int(input("m = "))
a = []
for i in range(n):
    b = []
    for i in range(m):
        b.append(int(input()))
    a.append(b)
l = 0
k = 0
max1 = a[0][0]
for i in range(n):
    for j in range(m):
        if a[i][j] > max1:
            max1 = a[i][j]
            l = i
            k = j
for i in range(n):
    for j in range(n):
        print(a[i][j], end=' ')
    print()
print(' ең үлкен элемент: ',max1)
print(' жол номері: ',l, 'баған номері: ', k)
```

Программаның орындалуы:

```
n = 3
m = 3
4
...
65
4
9
4 5 6
3 4 5
65 4 9
 ең үлкен элемент:  65
 жол номері:  2 баған номері:  0
>>>
```

5-мысал. Квадрат матрицаның бас диагоналында орналасқан элементтерді нольмен алмастыратын программа құрайық.

```
n = int(input("n = "))
m = int(input("m = "))
a = []
for i in range(n):
    b = []
    for i in range(m):
        b.append(int(input()))
    a.append(b)
for i in range(n):
    for j in range(m):
        if i==j:
            a[i][j] = 0

for i in range(n):
    for j in range(n):
        print(a[i][j], end=' ')
    print()
```

```
RESTART: C:/Ergalym/Project
School/phyton/Library prog/max mass.py =====
n = 3
m = 3
1
2
...
8
9
0 2 3
4 0 6
7 8 0
>>>
```

6-мысал. Квадрат матрицаның бас диагоналының бойында орналасқан элементтерді бірмен, бас диагональ үстіндегі элементтерді нольмен, ал бас диагональдың астында орналасқан элементтерді екі санымен алмастыратын программа құрайық.

```
n = int(input("n = "))
m = int(input("m = "))
a = []
for i in range(n):
    b = []
    for i in range(m):
        b.append(int(input()))
    a.append(b)
for i in range(n):
    for j in range(m):
        if i < j:
            a[i][j] = 0
        elif i > j:
            a[i][j] = 2
        else:
            a[i][j] = 1
 for i in range(n):
    for j in range(n):
        print(a[i][j], end=' ')
    print()
```

```
RESTART: C:/Ergalym/Project
School/phyton/Library prog/max mass.py =====
n = 3
m = 3
1
...
8
9
1 0 0
2 1 0
2 2 1
>>> >>>
```

9.5. Массив (тізім) элементтерін сұрыптау

Массив элементтерін олардың реттік номері бойынша сұрыптауды қарастырайық.

Алдымен сандар тізімін оқып, одан кейін әрбір санға реттік номер беру қажет. Мұны кортеж арқылы жүзеге асыруға болады. Ол үшін бос тізім құрамыз да, тізімнің элементтері мен номерінен тұратын жаңа тізім кортежін оған қосамыз. Тізім элементтері тұтасымен сұрыпталатын болғандықтан, оны циклдың ішінде қарастырамыз. Есептің шартына сәйкес, алдымен элементтер сұрыпталатын болғандықтан, бірінші элемент, одан кейін оның индексі жазылады. Келесі қадамда кортежден тұратын тізім элементтеріне сұрыптау әдісін пайдаланамыз да, тізім элементтерін экранға шығарамыз.

```
mt = list(map(int, input().split()))
n = []
for i in range(len(mt)):
   n.append((mt[i], i))
n.sort()
print(n)
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/sort list 1.py
9 2 7 5 6 3
[(2, 1), (3, 5), (5, 3), (6, 4), (7, 2), (9, 0)]
>>>
```

Программаның орындалу нәтижесінен массив элементтерінің өсу ретімен сұрыпталып, әрбір элементтің жанында реттік саны шығарылғанын көруге болады. Тізім элементтерінің реттік саны нольден басталатынын білеміз. Мысалы, ең кіші элемент 2, ал оның реттік саны 1 болса,

тізімдегі ең үлкен элемент 9, оның реттік саны 0, яғни бірінші тұрған элемент.

Егер элементтің өзі емес, тек реттік санын ғана элементтердің өсу ретімен экранға шығару үшін программаға төмендегідей өзгеріс енгіземіз.

Тізімдегі ең кіші элементті табу үшін сұрыпталған тізімдегі реттік саны 0-ші элементті шығару жеткілікті.

```
mt = list(map(int, input().split()))
mt.sort()
print(mt[0])
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python
36-32/Gulira Python/min max list .py
9 8 5 7 1 6
1
>>>
```

Python тілінде ең үлкен және ең кіші элементті іздеуге арналған арнайы функциялар бар.

```
mt = list(map(int, input().split()))
print(min(mt))
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python
36-32/Gulira Python/min max list .py
9 8 5 7 1 6
1
```

Дәл осылай ең үлкен элементті де табуға болады:

```
mt = list(map(int, input().split()))
print(max(mt))
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/max list .py
5 7 9 1 3 4
9
>>>
```

Бақылау сұрақтары мен жаттығулар

1. Массив деп нені түсінеміз?
2. Массив элементтері программада қалай белгіленеді?
3. Программада массив элементтерін енгізу қалай ұйымдастырылады?
4. Программада массив элементтерін шығару қалай ұйымдастырылады?
5. Массив элементтеріне қандай амалдар қолданылады?
6. Бүтін сандардан құрылған A(n) массивіндегі берілген X санына тең болатын элементтер санын табыңдар.
7. Бүтін сандардан құрылған сызықтық массив берілген. Массивтің k-шы элементін өшіретін программа құрыңдар.
8. Бүтін оң және теріс сандардан құрылған бір өлшемді масив берілген. Осы массив элементтерінен екі жаңа массив құратын программа жазыңдар. Бірінші массивте оң элементтер мен нольдер, екінші массивте теріс элементтер орналасуы тиіс.
9. Бүтін сандардан құрылған сызықтық массив берілген. Массивтің k-шы элементін бірінші элементпен алмастыратын программа құрыңдар.
10. Бүтін сандардан құрылған элементтері әртүрлі A(n) және B(n) массивтері берілген. Осы массивтердің

ортақ элементтерін экранға шығаратын программа құрыңдар.

11. P және n натурал сандары мен $a_1,\ldots,a_n$ бүтін сандары берілген. $a_1,\ldots,a_n$ тізбегінің P санына еселі болатын мүшелерінің көбейтіндісін табыңдар.

12. P, q және $a_1,\ldots,a_n$ бүтін сандары берілген (P>q>=0). $a_1,\ldots,a_n$ тізбегіндегі модулі P-ға бөлгенде q қалдық беретін мүшелерін нольмен алмастырыңдар.

13. $a_1,\ldots,a_n$ тізбегіндегі

а) 5-ке еселі;

б) тақ және теріс;

в) 5-ке бөлінетін және 7-ге бөлінбейтін;

г) $|a_i|<i^2$ шартын қанағаттандыратын мүшелерінің қосындысын және санын табатын программа құрыңдар.

14. A(m,n) матрицасының әр жол элементтерінің ең үлкенін экранға шығаратын программа құрыңдар.

15. A(m,n) матрицасының диагональдарындағы ең кіші элементті экранға шығаратын программа құрыңдар.

16. A(m,n) матрицасының диагоналындағы және одан жоғары жатқан элементтерін нольмен алмастырыңдар.

17. A(m,n) матрицасы берілген. Осы матрицаға кері матрицаны алатын программа құрыңдар.

10. ҚҰРЫЛЫМДАР

Басқа программалау тілдеріндегідей, Python тілінде жазу немесе құрылым деген ұғым жоқ. Бірақ құрылымға ұқсас объектілерді құруға болады және оның бірнеше тәсілдері бар.

Бұл жаңа объектілер типі ретінде құрылады. Ол класс деп аталады. Класс атауы барлық уақытта бас әріптен басталып жазылады.

Сұрыптауға арналған мысалды қарастырайық. Студенттерді жасы мен аты бойынша сұрыптап көрейік. Ол үшін екі өріс енгіземіз: жасы (age) және аты (name). Кортежде бұл өрістер реттік санмен белгіленген болатын, мұнда өрістерге мағынасына қарай атау береміз.

```
class Stud:
    age = 0
    name = ''
```

Тізімге студенттер туралы мәліметтер қалай қосылады? Ең алдымен, студенттер тізімін құрамыз да, студенттер санын оқимыз:

```
s = []
n = int(input())
```

Студенттер санын енгізгеннен кейін әрбір студентке қатысты мәміеттерді енгізу «үшін» қайталану операторы арқылы жүзеге асырылады. Әрбір студент үшін екі параметр енгізіледі: жасы – санмен, ал аты – жолдық шамамен. Жасы мен аты енгізіленнен кейін оған split() әдісін пайдаланамыз. Мұнда map функциясын пайдалана алмаймыз, себебі, жасы – санмен, ал аты – жолдық шамамен берілген.

```
for i in range(n):
    a,n = input().split()
```

Студенттің жасы шамасын санға түрлендіреміз.

```
a = int(a)
```

Stud() типті stud айнымалысын құрамыз, ешқандай параметрлер берілмейтіндіктен бос жақша жазылады:

```
stud = Stud()
```

Кластың, дәлірек айтқанда, құрылымның өрістері туралы мәлімет аламыз:

```
stud.age = a
stud.name = n
```

Сәйкес өрістер туралы мәліметтерді алғаннан кейін, оны тізімнің соңына жалғаймыз.

```
s.append(stud)
```

Сонымен, класс түрінде құрылған студенттер тізімін алдық.

Енді класстың, құрылымның элементтерін салыстыруды функция түрінде жазамыз.

```
def mTuple(stud):
   return (stud.age, stud.name)
def mTuple(stud):
   return (stud.age, stud.name)
s.sort(key=mTuple)
for k in s:
   print(k.age, k.name)
```

Студенттер жасы мен аты бойынша сұрыпталады. Егер тізімдегі студенттердің жастары бірдей болса, онда олар аттары бойынша сұрыпталады.

Программаның толық мәтіні:

```
class Stud:
   age = 0
   name = ''
s = []
n = int(input())
for i in range(n):
   a,n = input().split()
   a = int(a)
   stud = Stud()
   stud.age = a
   stud.name = n
   s.append(stud)
def mTuple(stud):
   return (stud.age, stud.name)
s.sort(key=mTuple)
```

```
for k in s:
    print(k.age, k.name)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/struct 1.py
5
25 Dana
23 khan
26 Liza
21 Mars
24 Gul
21 Mars
23 khan
24 Gul
25 Dana
26 Liza
>>>
```

Тізімде сақталған жолды ұзындығы бойынша сұрыптауды қарастырайық.

Тізімнің элементтерін тұрақты ретінде сипаттайық. Тізім элементтерін кему реті бойынша емес, өсу ретімен сұрыптайық. Ол үшін sort деп аталатын әдісті шақырып, оған key деп аталатын атаулы параметр берейік. Бұл функция тізімнің әрбір объектісіне қолданылып, одан қандай да бір мән алынады да, алынған мән бойынша салыстырылады. Егер key = len деп жазатын болсақ, жолдың ұзындығын есептеу үшін len функциясы шақырылады (Бұл функцияның қызметі туралы алдыңғы тақырыптарда айтылған). Дәлірек айтқанда, тізімнің әрбір элементі үшін len функциясы шақырылып орындалады және одан алынған мән сақталып, сол мән бойынша сұрыптау жүргізіледі.

```
m = ['com', 'puter', 'exit', 'on']
m.sort(key=len)
print(*m)
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list key.py
on com exit puter
>>>
```

Программаның орындалу нәтижесінен тізім элементтеріндегі символдардың саны бойынша өсу ретімен сұрыпталғанын көруге болады.

Егер символдар саны бірдей элементтер болса, онда олардың қайсысы бірінші тұрса, соны алдымен шығарады. Сұрыптаудың бұл қасиетін «орнықтылық» деп атайды. Python тілінде стандартты сұрыптау орнықты сұрыптау болып табылады.

Біз алдыңғы мысалда студенттерді жасы мен аты бойынша өсу ретімен сұрыптағанбыз. Енді оларды кему ретімен сұрыптауды қарастырайық.

Программаның бастапқы берілгендері өзгеріссіз қалады. Өзгерген кортеждің мәнін беретін функция енгізу қажет. Функцияға студенттің жасы (сан) және аты (жолдық шама) беріледі. Функцияның орындалуы нәтижесінде жасы үлкен студент бірінші шығуы тиіс. Ол үшін студенттің жасын сипаттайтын өріс теріс таңбамен алынады, егер жастары бірдей студент болатын болса, онда олардың аттары бойынша сұрыпталады. Одан кейін кортеж sTuple кілті бойынша сұрыпталады.

```
k = [(21, 'Mars'),
    (25, 'Dana'),
    (24, 'Gul'),
    (26, 'Liza'),
    (24, 'Khan')]
def sTuple(stud):
   return (-stud[0], stud[1])
k.sort(key=sTuple)
print(*k)
```

Программаның орындалу нәтижесінен жасы бірдей студенттер аты бойынша сұрыпталғанын көруге болады.

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list key2.py
(26, 'Liza') (25, 'Dana') (24, 'Gul') (24, 'Khan') (21,
'Mars')
>>>
```

Сонымен, **key** параметрінің көмегімен мәліметтерді түрлендіретін функцияны беруге, пайдалануға болады.

1-мысал. Қызметкер класын құрып, олардың аты-жөні мен жалақысы туралы мәліметті экранға шығаратын программа құрайық.

```
class Employee:
   """Қызметкерлер"""
   emp_count = 0
   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.emp_count += 1
   def display_count(self):
      print('Барлық қызметкерлер: %d' %
Employee.emp_count)
   def display_employee(self):
      print('Аты: {}. Жалақысы: {}'.format(self.name,
self.salary))
 # Employee класының 1-объектісі құрылады
emp1 = Employee("Алмас", 200000)
# Employee класының екінші объектісі құрылады
emp2 = Employee("Майра ", 150000)
emp1.display_employee()
emp2.display_employee()
```

```
print("Барлық қызметкерлер: %d" % Employee.emp_count)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python prog/struct1.py ==========
Аты: Алмас. Жалақысы: 200000
Аты: Майра . Жалақысы: 150000
Барлық қызметкерлер: 2
>>>
```

Бақылау сұрақтары мен жаттығулар

1. Python тілінде жазу деп нені түсінеміз?
2. Құрылымға мысалдар келтіріңдер.
3. Прогрaмма тақырыбында жазулар қалай бейнеленеді?
4. Жазу элементтеріне қандай операцияларды қолдануға болады?
5. Жазу мәліметтерін енгізу мен шығаруды қалай ұйымдастыруға болады?
6. Жазудың массивтен айырмашылығы неде?
7. Жазуға бір типті элементтерді пайдалануға бола ма?
8. Жазуға қандай оиерациялар қолдануға болады?
9. Жазу типті мәліметтерді пайдалана отырып, төмендегідей есептерді шығаруға программа құрыңдар:

А) Жазықтықта n нүкте берілген. Координаталар басына жақын орналасқан нүктені табыңыз.

Ә) Нүкте мен түзудің, екі түзудің және екі жазықтықтың арасындағы қашықтықты табыңыз.

Б) Дене бір қалыпты үдемелі қозғалады. Дененің Т уақыттан кейінгі жылдамдығы мен үдеуін табыңыз.

10. Нүкте мен түзудің, екі түзудің және екі жазықтықтың арасындағы қашықтықты табыңыз.

11. Дене бір қалыпты үдемелі қозғалады. Дененің Т уақыттан кейінгі жылдамдығы мен жолын табыңыз.

12. 4A+7B-3A+8A-2K+1P-5R-2B көпмүшелігі берілген. А үшін ұқсас мүшелерді тауып қосынды коэффициентті есептеңдер.

13. Группада 30 оқушы бар. Студенттердің берілген аты жөні және 6 сабақтан тапсырған емтихан бағасы бойынша әрбір оқушының орта балын, аты-жөнін және ең жақсы оқушының орта балын есептеп экранға шығаратын программа құрыңдар.

14. Ағымдағы күннен Х күн өткеннен кейін болатын күнді есептейтін программа құрыңдар.

15. Мекеменің автомобильдері жөніндегі мәліметтерді (марка, түсі, шыққан жылы, сапасы, құны) сақтап, жасалған сұрау бойынша мәліметтерді беретін программа құрыңдар.

16. Қала аралық автостанциядағы автобустар қозғалысының кестесін сақтайтын программа құрыңдар: рейс нөмірі, маршрут (бастапқы және соңғы пункттері), жүру мезгілі.

17. "Студент" деп аталатын жазу типті мәлімет берілген. Төмендегідей ақпаратты өңдейтін программа құрыңдар.

А) Қалада және жатаханада қанша студент тұрады;

Ә) Бірінші семестрдегі қанша студенттің балы 12-ден жоғары;

В) Екінші семестрдегі қанша студенттің балы 13-тен төмен;

Г) Оқу жылы бойынша орташа оқу үлгерімін есептеу;

Д) Үлгермейтін студенттердің тізімін шығару;

Е) Тізімді балдың кемуі бойынша реттеу;

Ж) Тізімді алфавиттік ретпен реттеу.

11. ЖИЫНДАР

Python программалау тілінде жиындар – бұл реттелмеген элементтерден тұратын мәліметтер құрылымы болып табылады. Жиын элементтері номерленбейді (индекстелмейді). Тізім элементтері тәрізді, жиын элементтері де өшіріп, өзгерістер енгізуге мүмкіндік береді. Дегенмен, жиын элементтерінің басқа мәліметтер құрылымымен салыстырғанда бірқатар ерекшеліктері бар:

- жиында қайталанатын элементтер болмайды;
- жиын элементтері өзгермейді, бірақ жиынның өзін өзгертуге болады (мысалы, жиын элементтерінен list функциясының көмегімен тізім құруға болады);
- жиын элементтері индекстелмейтіндіктен, ешқандай қию және индекстеу операциялары орындалмайды.

11.1. Жиын элементтерін құру

Python программалау тілінде жиын элементтерін құру басқа құрылымдық типтерді құруға ұқсас. Барлық жиын элементтері фигуралық жақшаның ішіне алынып жазылады және элементтер үтір арқылы ажыратылады. Жиын кез келген мөлшердегі элементтерді қамтуы және жиын элементтері әртүрлі болуы мүмкін. Мысалы, бүтін сандар, жолдар, кортеждер және т.б. бір жиынның элементтері болуы мүмкін. Бірақ жиын өзгеретін элементтерді, тізімдер, сөздіктер және т.б. қамти алмайды.

Енді Python программалау тілінде жиын құру мысалын қарастырайық.

1-мысал. Бес элементтен тұратын жиынды элементтерімен экранға шығаратын программа құрайық.

```
myset = {5,7,9,4,3}
print(myset)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
create.py =========
{3, 4, 5, 7, 9}
>>>
```

Дәл осылайша жолдардан тұратын жиын құруға болады.

2-мысал. Элементтері апта күндері болатын жиынды экранға шығаратын программа құрайық.

```
my_str_set = {"дүйсенбі", "сейсенбі", "сәрсенбі",
"бейсенбі", "жұма", "сенбі", "жексенбі"}
print(my_str_set)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
create.py =========
{'сейсенбі', 'сенбі', 'дүйсенбі', 'сәрсенбі', 'жұма',
'бейсенбі', 'жексенбі'}
>>>
```

Программаның орындалуынан жиын элементтерінің орындарының ауысып экранға шығарылғанын көруге болады. Бұл жиын элементтерінің реттелмегендігін көрсетеді. Программаның әрбір орындалуында жиын элементтерінің орналасу реті өзгере беруі мүмкін.

Сонымен бірге, жиын әртүрлі типті элементтерді қамти береді.

```
mixed_set = {5, "Жұмабек", (7, 9, 3)}
print(mixed_set)
```

Python программалау тілінде тізімнен тұратын жиын құруға болады. Ол үшін жүйеде орнатылған **set()** функциясы пайдаланылады.

3-мысал. Қайталанатын элементтері бар тізімнен жиын құрыңдар.

```
num_set = set([1, 9, 5, 5, 9])
print(num_set)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
create.py ========
{1, 5, 9}
>>>
```

Бұл мысалдан жиында қайталанатын элементтер болмайтындығын көруге болады.

Python программалау тілінде бос жиын құру үшін **set()** функциясы пайдаланылады.

Мысалы,

```
x = set()
print(type(x))
```

Программа орындалғанда x айнымалысының типі жиын екенін көрсетеді.

```
<class 'set'>
```

11.2. Жиын элементтерімен жұмыс істеу

Жиынның барлық элементтерімен жұмыс істеу үшін цикл пайдаланылады. Мысалы,

```
season = set(["summer", "autumn", "winter", "spring"])
for i in season:
    print(i)
```

Программаның орындалуы нәтижесі:

```
RESTART: C:/Ergalym/Project
School/phyton/sets/set create.py ========
        spring
        winter
        autumn
        summer
>>>
```

Сонымен бірге, ізделінді элементтің жиында бар екенін де тексеруге болады. Оны **in** көмегімен тексеруге болады:

```
season = set(["summer", "autumn", "winter", "spring"])
print("spring" in season)
```

Нәтижесінде **True** мәні экранға басылып шыгады.

11.3. Жиынға элементтер қосу

Python тілінде жиынға жаңа элементтер қосу **add()** функциясының көмегімен жүзеге асырылады.

Мысалы,

```
season = set(["autumn", "winter", "spring"])
season.add("summer")
print( season)
```

Программаның орындалу нәтижесі:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
create.py
{'summer', 'spring', 'autumn', 'winter'}
>>>
```

11.4. Жиындағы элементтерді өшіру

Python тілінде жиындағы элементтерді өшіру мүмкіндігі бар, бірақ жиын элементтері номерленбейтінін айтқанбыз. Сондықтан жиын элементтерін өшіруде индекс пайдаланылмайды. Жиын элементтерін өшіру үшін **discard**(), **remove**() әдістерінің бірін пайдалануға болады. Бұл әдістердің пайдаланылуында өзгешеліктер бар.

discard() әдісін пайдаланғанда, жиында ізделінді элемент болмаса, онда қателік шығармайды. Ал, **remove**() әдісі пайдаланылғанда элемент табылмаса, қателік шығады.

```
num_set = {1, 2, 3, 4, 5, 6}
num_set.discard(3)
print(num_set)
```

```
RESTART: C:/Ergalym/Project
School/phyton/sets/set create.py
{1, 2, 4, 5, 6}
>>>
```

Программаның орындалу нәтижесінен 3-элементінің өшірілгенін көруге болады.

Енді **remove**() әдісі пайдаланалайық.

```
num_set = {1, 2, 3, 4, 5, 6}
num_set.remove(9)
print(num_set)
```

```
Traceback (most recent call last):
  File "C:/Ergalym/Project School/phyton/sets/set cre-
ate.py", line 2, in <module>
    num_set.remove(9)
KeyError: 9
>>>
```

Программаның орындалуынан модульдің екінші жолында қате бар екендігі көрсетілген, **remove()** әдісі **KeyError** типті қатені, «9» деген элементтің жиында жоқ екенін шығарып тұр.

Python тілінде жиынның барлық элементтерін өшіруге мүмкіндік беретін әдіс бар. Ол **clear()** деп аталатын әдіс. Бұл әдіс орындалғанда бос жиын аламыз.

Мысалы,

```
myset = {5,7,9,4,3}
myset.clear()
print(myset)
```

```
RESTART: C:/Ergalym/Project
School/phyton/sets/set 2.py
set()
>>>
```

11.5. Жиындардың бірігуі

Бізге X және У жиындары берілсін. Екі жиынның бірігуі дегеніміз – бұл екі жиындағы барлық элементтердің жиынтығы. Python тілінде мұндай операция **join()** функциясының көмегімен орындалады.

```
weeks_x = set(["дүйсенбі", "сейсенбі", "жұма", "жексенбі"])
weeks_y = set(["сәрсенбі", "бейсенбі", "сенбі"])
wholeweek = weeks_x.union(weeks_y)
print(wholeweek)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
union.py

{'сенбі', 'дүйсенбі', 'бейсенбі', 'сәрсенбі', 'сейсенбі',
'жұма', 'жексенбі'}
>>>
```

Жиын элементтері реттелмейтіндіктен, апта күндерінің ретсіз орналасқанын көріп отырмыз.

Бірнеше жиынның элементтерін бір жиынға біріктіруге болады. Мысалы,

```
myset = {5,7,9,4,3}
weeks_x = set(["дүйсенбі", "сейсенбі", "жұма", "жексенбі"])
weeks_y = set(["сәрсенбі", "бейсенбі", "сенбі",3])
season = set(["жаз", "күз", "қыс", "көктем"])
allsets = myset.union(weeks_y, weeks_x, season)
print(allsets)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set union.py

{'сенбі', 'жексенбі', 'жаз', 3, 4, 5, 'сейсенбі', 7, 9, 'сәрсенбі', 'бейсенбі', 'жұма', 'күз', 'қыс', 'көктем', 'дүйсенбі'}
>>>
```

Программаның орындалуынан төрт жиын элементтерінің бір жиынға біріккенін және жиындағы қайталанатын элементтердің бір рет қана шығарылатынын көрдік.

Жиындардың бірігуін жүзеге асыратын келесі оператор - | белгісі болып табылады.

```
n1 = set(["Қанат","Санат", "Майра", "Алма"])
n2 = set(["Дәрия", "Майра", "Алмас" ])
n3 = set(["Дана", "Марат", "Алмас", "Жанат"])
print("жиындардың бірігуі: ", n1|n2|n3)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/sets/set 3.py ==========

жиындардың бірігуі: {'Алмас', 'Алма', 'Қанат',
'Дана', 'Майра', 'Дәрия', 'Санат', 'Марат', 'Жанат'}
>>>
```

Программаның орындалуынан қайталанатын жиын элементтерінің кездеспейтінін көруге болады.

11.6. Жиындардың қиылысуы

Жиындардың ортақ элементтері олардың қиылысуын береді. Бізге X және У жиындары берілсін. Жиын элементтерінің қиылысуы **&** операторының және **intersection()** функциясының көмегімен жүзеге асырылады.

```
myset1 = {5,7,9,4,3}
myset2 = {5,11,6,1,3}
print("екі жиынның қиылысуы:" myset1& myset2)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/sets/set4.py ==========
екі жиынның қиылысуы:  {3, 5}
>>>
```

Бұдан екі жиындағы ортақ элементтер 3 және 5 екенін көреміз.

Осы мысалды **intersection()** функциясының көмегімен тексеріп көрейік.

```
myset1 = {5,7,9,4,3}
myset2 = {5,11,6,1,3}
k = myset1. intersection(myset2)
print("екі жиынның қиылысуы:", k)
```

```
RESTART: C:/Ergalym/Project
School/phyton/sets/set4.py ==========
екі жиынның қиылысуы:  {3, 5}
>>>
```

11.7. Жиындардың айырымы

Жиындардың айырымы дегеніміз – бұл бірінші жиынға кіретін, ал екінші жиынға кірмейтін элементтердің жиынтығы болып табылады.

Мысалы, Х және У жиындары берілсін. Х және У жиындарының айырымы Х жиынына кіретін барлық элементтерді қамтиды, оған У жиынының элементтері кірмейді.

Python тілінде жиындардың айырымын табу үшін difference() немесе «—» операторы пайдаланылады.

```
set1 = {11,14, 17, 21,28}
set2 = {11,16, 17, 23,24}
diffset = set1.difference(set2)
print("екі жиынның айырымы:", diffset)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
diff.py

екі жиынның айырымы: {28, 21, 14}
>>> >>>
```

Программаның орындалуынан бірінші жиынның тек үш элементі гана екінші жиынға кірмейді және осы

элементтер екі жиынның айырымы болады (дәлірек айтқанда, бірінші жиынның екінші жиыннан айырмашылығы).

Сондай-ақ, жиындардың айырымын «алу» (минус) операциясы арқылы да табуға болады.

```
set1 = {11,14, 17, 21,28}
set2 = {11,16, 17, 23,24}
print("екі жиынның айырымы:", set1-set2)
```

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
diff.py

екі жиынның айырымы: {28, 21, 14}
>>>
```

Python тілінде жиын элементтері арасындағы симметриялық айырымды табу мүмкіндігі бар. Жиын элементтерінің симметриялық айырымы дегеніміз – екі жиынға да ортақ элементтерден басқа элементтерді қамтитын жиын элементтері болып табылады. Жиындардың симметриялық айырымын анықтау үшін **symmetric_difference()** әдісін немесе ^ операторы пайдаланылады.

```
set1 = {11,14, 17, 21,28}
set2 = {11,16, 17, 23,24}
sdiff = set1.symmetric_difference(set2)
print("екі жиынның симметриялық айырымы:", sdiff)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
diff.py

екі жиынның симметриялық айырымы: {16, 21, 23,
24, 28, 14}
```

Программаның орындалуынан екі жиынға ортақ 11 және 17 элементтерінің кірмегенін көруге болады.

Дәл осы операцияны ^ операторының көмегімен де табуға болады:

```
set1 = {11,14, 17, 21,28}
set2 = {11,16, 17, 23,24}
print("екі жиынның симметриялық айырымы:", set1^set2)
```

```
RESTART: C:/Ergalym/Project School/phyton/sets/set diff.py

екі жиынның симметриялық айырымы: {16, 21, 23, 24, 28, 14}
>>>
```

11.8. Жиындарды салыстыру

Python тілінде жиын элементтерін салыстыру мүмкіндігі қарастырылған. Жиын элементтерін салыстырғанда жиынның аталық немесе басқа жиынның ұрпағы болып табылатындығы туралы айтылады. Мұндай салыстырудың нәтижесі ақиқат (true) немесе жалған (false) болып табылады.

X жиыны У жиынының ұрпағы болып табылатындығын тексеру үшін төмендегідей операция орындалады:

X <= У

Ал, У жиыны X жиынының ұрпағы болып табылатындығын тексеру үшін төмендегідей шарт тексеріледі:

X >=У

Мысалы,

```
x = set(["дүйсенбі", "сейсенбі", "жұма", "жексенбі", "сәрсенбі", "бейсенбі", "сенбі"])
y = set(["сәрсенбі", "бейсенбі", "сенбі"])
subset = x >=y
bigset = y<=x
```

```
print("X = ",x)
print(" X  сыртқы жиын =", subset)
print("y= ",y)
print("Y ішкі  жиын =", bigset)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project
School/phyton/sets/compare set.py

X =  {'жексенбі', 'сәрсенбі', 'дүйсенбі', 'сейсенбі',
'бейсенбі', 'жұма', 'сенбі'}
 X  сыртқы жиын = True
y=  {'сәрсенбі', 'бейсенбі', 'сенбі'}
Y ішкі  жиын = True
>>>
```

11.9. Жиындарға пайдаланылатын әдістер

Python программалау тілінде жиындарға пайдаланылатын бірқатар орнатылған әдістер қарастырылған. Сол әдістердің негізгілерін қарастырайық.

1. copy() әдісі. **copy()** әдісі жиынның көшірмесін алу үшін пайдаланылады.

```
season = set(["summer", "autumn", "winter", "spring"])
k = season.copy()
print("k=",k)
```

Программаның орындалуы

```
RESTART: C:/Ergalym/Project School/phyton/sets/set
copy.py

k= {'winter', 'autumn', 'summer', 'spring'}
>>>
```

Программаның орындалу нәтижесінен k жиыны season жиынының көшірмесі екенін көруге болады.

Келесі әдіс жиындардың қиылысатынын тексеруге арналған. Ол **isdisjoin()** деп аталады. Егер жиындардың ортақ элементтері болмаса, бұл әдіс ақиқат, кері жағдайда жалған мәнін береді. .

```
season = {"жаз", "күз", "қыс", "көктем"}
weeks_x = {"дүйсенбі", "сейсенбі", "жұма", "жексенбі"}
y = season. isdisjoint(weeks_x)
print("y=",y)
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set disjoin.py

y= True
```

Келтірілген екі жиынның ортақ элементтері болмағандақтан, ақиқат мәнін шығарып тұр.

Келесі әдіс жиындағы элементтердің санын есептеуге арналған. Ол **len()** әдісі. Мысалы,

```
season = {"жаз", "күз", "қыс", "көктем"}
print("жиындағы элементтер саны: ",len(season))
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project School/phyton/sets/set disjoin.py

жиындағы элементтер саны:  4
>>>
```

Сонымен, жиын математикада қандай мәнге ие болса, Python программалау тілінде де сондай мәнге ие.

Жиын – реттелмеген элементтердің жиынтығы. Жиынды өзгертуге, түрлендіруге болады, бірақ оның элементтері өзгермейді.

Жиынға элементтер қосуға, бар элементті өшіруге болады. Жиын элементтері индекстелмейтіндіктен, оның белгілі бір элементтеріне бағытталған операциялар орындалмайды.

Бақылау сұрақтары мен жаттығулар

1. Жиын деп нені түсінеміз?
2. Математикада жиындарға қандай амалдар қолданылады?
3. Python программалау тілінде жиындарға қолданылатын амалдар қалай жазылады?
4. Жиын элементтері қалай енгізіледі?
5. Бос жиын дегеніміз не және ол қалай белгіленеді?
6. Жиын элементтерін шығару қалай ұйымдастырылады?
7. Жиындардың бірігуі қалай жүзеге асырылады?
8. Жиындардың қиылысуы мен айырымы қалай табылады?
9. Жиын элементтері қалай салыстырылады?
10. Жиын элементтерін салыстырғанда қандай жағдайда жалған нәтиже алынады?
11. Жиын элементтерінің санын есептеудің қандай әдістері бар?
12. Жиындардың қиылысуы қалай тексеріледі?
13. Жиын элементтерінің симметриялық айырымы дегеніміз не?
14. & операторы мен intersection() функциясының айырмашылығы неде және қай уақытта пайдаланылады?
15. Жиындардың басқа мәліметтер типтерінен, мысалы, массивтер, кортеждерден айырмашылығы неде?
16. Жиын элементтерін массив, жолдық немесе кортеж элементтеріне түрлендіруге бола ма?
17. Python тілінде жиындарға қолданылатын қандай әдістер бар?

18. 1..60 аралығындағы сандардан құрылған X[1..n] және У[1..m] екі тізбегі берілген. Осы екі тізбектің барлық ортақ элементтерін табыңдар.

19. 100...150 аралығындағы сандардан құрылған X[1..n] және У[1..m] тізбектері берілген. X тізбегіне кіретін, ал У тізбегіне кірмейтін барлық элементтерін табыңдар.

20. 1...30 аралығына 40 сан берілген. Осы сандардың ішіндегі Фиббоначчи санының қанша екенін және бірінші мәнді цифры 1 немесе 2 болатын қанша цифр бар екенін анықтаңдар.

21. Символдардан құралған S жолындағы кездесетін "*","+","-" арифметикалық амал белгілерінің мөлшерін анықтаңдар.

22. 100…250 аралығындағы сандардан құралған X[1..n], У[1..m] және Z[1..k] тізбектері берілген. Алғашқы екі тізбекте кездесетін, ал, үшінші тізбекте кездеспейтін барлық элементтерді табыңдар.

23. 100…180 аралығындағы сандардан құралған X[1..n], У[1..m] және Z[1..k] тізбектері берілген. Осы тізбектерде тек бір реттен ғана кездесетін барлық элементтерді табыңдар.

24. Нақты сандардан құрылған A[1..n,1..n] матрицасының жолдары мен бағандарының нөмірлері сәйкес типі 1..n болатын S1 және S2 бос емес жиынында жататын элементтерінің қосындысын табыңдар.

25. Цифрлар мен латын әріптерінен құрылған мәтін берілген. Осы мәтіндегі дауысты немесе дауыссыз дыбыстардың қайсысы көп екенін анықтаңдар.

26. 100…250 аралығынан алынған X[1..n] тізбегі берілген. Тізбектегі қайталанбай, тек бір реттен кездесетін әрбір санды басып шығарыңдар.

27. 1..250 аралығынан алынған n^2+m^2 түрінде берілген барлық бүтін сандарды өсу ретімен басып шығарыңдар.

28. [1..200] аралығындағы барлық жай сандарды өсу ретімен басып шығаратын программа құрыңдар.

12. СӨЗДІК

12.1. Сөздік құру. Сөздікке элементтер қосу, өзгерту және өшіру

Python программалау тілінде тізімдер және кортеждермен бірге, сөздік деп аталатын мәліметтер типі пайдаланылады. Сөздік – бұл тізімдер тәрізді өзгеретін, реттелмеген (жолдар, тізімдер, кортеждерден өзгешелігі) элементтердің жиыны, оның элементтері «**түйінді сөз: мән**» түрінде сипатталады. Реттелмеген деген ұғым – элементтер жұбының орналасу реті маңызды емес екенін білдіреді және ол элементтердің индексі бойынша жұмыс істеу мүмкін емес екендігін корсетеді.

Сөздікке ұқсас мәліметтер құрылымы басқа программалау тілдерінде басқаша аталады.

Сөздік ұғымы түсінікті болу үшін оны әдеттегі аударма создіктермен салыстырып көрейік, ағылшын және қазақ тілі сөздігімен салысытырайық. Аударма сөздікте әрбір ағылшын создігінің аудармасы бар: cat - мысық, dog - ит, bird – құс және т.б. Егер ағылшын-қазақ тілі создігін Python тілінде сипаттайтын болсақ, онда ағылшын создерін түйінді сөз, ал қазақ тіліндегі сөздерді оның мәні ретінде сипаттауға болады:

{'cat': 'Мысық', 'dog': 'Ит', 'bird': 'Құс'}

Мұндағы фигуралық жақшалар олардың сөздік екенін анықтайды.

Python тіліндегі создіктің жазылу форматы төмендегідей:

{ түйінді сөз: мән, түйінді сөз: мән, түйінді сөз: мән, ... }

Сөздікте мән индекс бойынша емес, квадрат жақшаға алынған түйінді сөз бойынша ізделінеді. Мысалы,

```
>>> a['cat']
'Мысық'
>>> a['bird']
```

'Құс'
>>>

Сөздік тізім тәрізді, өзгермелі мәліметтер типіне жататындықтан элементтерді (**түйінді сөз: мән** жұбын) өзгертуге, қосуға және өшіруге болады. Алдымен сөздік бос болып құрылады, мысалы, а = {}, одан кейін элементтермен толтырылады. Элементтерді толтыру мен өзгертудің синтаксисі бірдей:

сөздік [түйінді сөз] = мән

Сөздіктің мазмұнын экранға шығару үшін **print()** функциясын пайдаланамыз.

Мысалы,

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020}
print(dict_mysal)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict2.py ==============
{'Company': 'Nissan', 'model': 'X-Trail', 'year': 2020}
>>> >>>
```

Сөздікке мән енгізгеннен кейін оны өзгертуге болады. Оны өзгерту үшін түйінді сөз пайдаланылады. Мысалы,

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020}
dict_mysal['year'] = 2021
print(dict_mysal)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict2.py ==============
{'Company': 'Nissan', 'model': 'X-Trail', 'year':
2021}
>>>
```

Создікке жаңа мәндер қосудың бірнеше тәсілдері бар. Жаңа түйінді сөз алып, оған мән меншіктеуге болады. Мысалы,

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020}
dict_mysal['Volume'] = '2.5'
print(dict_mysal)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict3.py ==============
{'Company': 'Nissan', 'model': 'X-Trail', 'year': 2020,
'Volume': '2.5'}
>>>
```

Программаның орындалуынан сөздікке қосылған түйінді сөз **'Volume'**, ал оның мәні **'2.5'** екенін көруге болады және ол сөздіктің ең соңғы элементі ретінде орналасты.

Егер түйінді сөз бұрыннан болатын болса, онда мән өзгереді, ал егер жаңа болса, онда жаңа мән қосылады. Создік элементтерін өшіру Python тілінің орнатылған del операторының көмегімен жүзеге асырылады. Мысалы,

```
b = {}
b[1] = 3.5
```

```
b[2] = 7.2
b[3] = 1.178
print(b)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict1.py ==============
        b= {1: 3.5, 2: 7.2, 3: 1.178}
        >>>
```

Del операторының комегімен бір элементті өшірейік. Мысалы,

```
b = {}
b[1] = 3.5
b[2] = 7.2
b[3] = 1.178

print('b=',b)
del b[2]
print('Екінші элемент өшірілгеннен кейінгі')
print('b=',b)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict1.py ==============
        b= {1: 3.5, 2: 7.2, 3: 1.178}
        Екінші элемент өшірілгеннен кейінгі
        b= {1: 3.5, 3: 1.178}
        >>>
```

Сөздіктегі элементтерді өшірудің екінші бір тәсілі pop() функциясын пайдалану болып табылады. Мұндағы түйінді сөз функцияның аргументі болып табылады. Мысалы,

```
dict_mysal = {
 "Company": "Nissan",
```

```
 "model": "X-Trail",
 "year": 2020}
dict_mysal.pop('year')
print(dict_mysal)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict2.py ==============
{'Company': 'Nissan', 'model': 'X-Trail'}
>>>
```

Программаның орындалуынан **'year'** түйінді сөзінің өшірілгенін көреміз.

Сөздіктегі соңғы элементті өшіруге болады. Ол үшін **popitem()** функциясы пайдаланылады.

```
dict_mysal = {
 "Company": "Nissan",
 "model": "X-Trail",
 "year": 2020,
 "volume": 2.5}
dict_mysal.popitem()
print(dict_mysal)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict3.py ==============
{'Company': 'Nissan', 'model': 'X-Trail', 'year':
2020}
>>>
```

Сөздіктің элементтерімен жұмыс істеуге get() функциясын пайдалануға болады. Ол сөздік атынан кейін жазылады және жақшаның ішіне түйінді сөз функцияның аргументі ретінде жазылады.

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020}
k = dict_mysal.get('model')
print(k)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict2.py ==============
X-Trail
>>>
```

Сөздікте екі элементті бірдей бір түйінді сөзбен сипаттауға болмайды, бірақ әртүрлі түйінді сөздердің мәндері бірдей болуы мүмкін. Түйінді сөз ретінде кез келген өзгермейтін мәліметтер типі алынады, ал оның мәні кез келген мәліметтер типі бола алады. Сөздіктің мәні басқа құрылымдар, мысалы, басқа создіктер немесе тізімдер болуы мүмкін. Мысалы,

```
nums = {'one': (1, 'I'), 'two': (2, 'II'), 'three': (3, 'III')}
```

Сөздік элементтері басқа күрделі типтердің элементтері тәрізді for операторымен таңдалады. Егер шығарылатын элемент нақты көрсетілмесе, тек түйінді сөздер ғана басылып шығады. Мысалы,

```
nums = {'one': (1, 'I'), 'two': (2, 'II'), 'three': (3, 'III')}
for i in nums:
   print(i)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict1.py ==============
        one
        two
        three
        >>>
```

Түйінді сөздер бойынша мәнді алуға болады. Мысалы,

```
a = {'cat': 'Мысық', 'dog': 'Ит', 'bird': 'Құс'}
for i in a:
   print(a[i])
```

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict1.py ==========
        Мысық
        Ит
        Құс
        >>>
```

Сөздікті барлық элементтерден тазалап, босату үшін clear() функциясы пайдаланылады.

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020}
dict_mysal.clear()
print(dict_mysal)
```

Программаның орындалуы:

```
RESTART:C:/Users/Asus/Desktop/python
prog/dict4.py ==========
{}
>>>
```

Программаның орындалуынан жиынның бос екенін көруге болады.

12.2. Сөздікте пайдаланылатын әдістер

Алдыңғы тақырыпта сөздікте пайдаланылатын бірқатар әдістер қарастырылды. Енді сөздікте пайдаланылатын кеңінен таралған әдістерді қарастырайық.

1. **len()** әдісі. Сөздіктегі элементтердің санын есептеу үшін **len()** әдісі пайдаланылады.

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020,
    "volume": 2.5}
print(len(dict_mysal))
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict4.py ==========
4
>>>
```

Программаның орындалуынан сөздікте 4 жазу бар екенін көреміз.

2. **copy()** әдісі. Сөздіктің көшірмесін алу үшін **copy()** әдісі пайдаланылады.

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020,
    "volume": 2.5}
x = dict_mysal.copy()
print(x)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict4.py ==============
{'Company': 'Nissan', 'model': 'X-Trail', 'year': 2020,
'volume': 2.5}
>>>
```

Программаның орындалуынан **dict_mysal** сөздігінің көшірмесі құрылып, ол x айнымалысына меншіктелгенін көреміз.

3. **items()** әдісі. Сөздіктегі өзгерістерді көрсету үшін **items()** объектісі пайдаланылады.

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020,
    "volume": 2.5}
x = dict_mysal.items()
print(x)
dict_mysal["model"] = "Qashqai"
print(x)
dict_mysal["model"] = "Patrol"
print(x)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict4.py ==============
dict_items([('Company', 'Nissan'), ('model', 'X-Trail'),
('year', 2020), ('volume', 2.5)])
dict_items([('Company', 'Nissan'), ('model', 'Qashqai'),
('year', 2020), ('volume', 2.5)])
dict_items([('Company', 'Nissan'), ('model', 'Patrol'),
('year', 2020), ('volume', 2.5)])
>>>
```

4. **fromkeys()** әдісі. Сөздікте пайдаланылатын келесі әдіс **fromkeys()** деп аталады. Бұл әдіс сөздікте көрсетілген түйінді сөздерді мәндерімен шығарады. Оның жазылу форматы:

dictionary.fromkeys(keys, value)

Мұндағы keys параметрінің мәні қайталанатын объектілер. Ол жаңа сөздіктің түйінді сөздеріне жауап береді.

Мәндері бірдей үш түйінді сөзден тұратын сөздік құрайық.

```
name = ('Ерғалым', 'Алмас', 'Асыл')
#age = 25
country = 'Қазақстан'
dict_mysal = dict.fromkeys(name, country)
print(dict_mysal)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict5.py ==========
{'Ерғалым': 'Қазақстан', 'Алмас': 'Қазақстан', 'Асыл':
'Қазақстан'}
>>>
```

Бұл мысалда түйінді сөздер анықталып, оларға бірдей мән меншіктелген. **fromkeys()** әдісі түйінді сөздерді мәндерімен біріктіріп, жаңа сөздік құрады.

Сөздіктегі **keys** параметрінің мәнін жазу міндетті болып есептеледі, ал егер **values** параметрі көрсетілмесе, онда мәні басылып шығады.

```
name = ('Ерғалым', 'Алмас', 'Асыл')
#age = 25
country = 'Қазақстан'
dict_mysal = dict.fromkeys(name)
print(dict_mysal)
```

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict5.py ==============
{'Ерғалым': None, 'Алмас': None, 'Асыл': None}
>>>
```

5. **setdefault()** әдісі. Бұл әдіс элементтің мәнін нақты түйінді созбен алу қажет болған жағдайда пайдаланылады. Егер түйінді сөз табылмаса, ол сөздікке көрсетілген мәнімен орналасатын болады. Әдістің жазылу форматы:

dictionary.setdefault(keyname, value)

Бұл әдісте **keyname** параметрінің жазылуы міндетті, ол мәні шығарылатын түйінді сөзді көрсетеді. Ал **value** параметрінің жазылуы міндетті емес. Егер түйінді сөз сөздікте болатын болса, онда ешқандай өзгеріс болмайды. Ал, түйінді сөз болмаса, функцияның мәні түйінді сөздің мәні ретінде қабылданады.

```
dict_mysal = {
  "Company": "Nissan",
  "model": "X-Trail",
  "year": 2020,
    "volume": 2.5}
x = dict_mysal.setdefault("color", "Blue")
#print(x)
print(dict_mysal)
```

Прогамманың орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict6.py ==============
{'Company': 'Nissan', 'model': 'X-Trail', 'year': 2020,
'volume': 2.5, 'color': 'Blue'}
>>>
```

Программаның орындалуынан color түйінді сөзінің мәнімен сөздікке қосылғанын көруге болады.

1-мысал. Берілген сөздік элементтерін пайдалана отырып, қойылған шартты қанағаттандыратындай жаңа сөздік құрыңдар.

```
a_dict = {'one': 1, 'two': 2, 'thee': 3, 'four': 4}
new_dict = {}  # Create a new empty dictionary
for key, value in a_dict.items():
    # If value satisfies the condition, then store it in new_dict
    if value <= 2:
        new_dict[key] = value
print(new_dict)
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python prog/dict7.py ==========
{'one': 1, 'two': 2}
>>>
```

2-мысал. Бизнесмен компаниясының айлық өнімін сөздікте сақтады. Бизнесмен компаниясының жылдық өнімін қалай есептейді?

```
mes = {'apple': 500000.00, 'orange': 90000.00, 'banana': 60000.00, 'pear': 85000.00}
total_income = 0.00
for value in incomes.values():
    total_income += value  # Accumulate the values in total_income
print( 'компанияның жылдық өнімі: ', total_income*12, 'tenge')
```

Программаның орындалуы:

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict8.py ==============
компанияның жылдық өнімі:  8820000.0 tenge
>>>
```

Сөздік реттелген мәліметтер құрылымына жататындықтан кез келген сөздіктің элементтерін **sorted()** әдісінің көмегімен сұрыптауға болады.

3-мысал. Компания өнімдерін товар аты бойынша сұрыптайтын программа құрайық.

```
incomes = {'apple': 500000.00, 'orange': 90000.00, 'banana': 60000.00, 'pear': 85000.00}
sorted_income = {k: incomes[k] for k in sorted(incomes)}
print('сұрыпталған товар: ', sorted_income)
```

```
RESTART: C:/Users/Asus/Desktop/python
prog/dict9.py ==============
сұрыпталған товар: {'apple': 500000.0, 'banana': 60000.0, 'orange': 90000.0, 'pear': 85000.0}
>>> >>>
```

Бақылау сұрақтары мен жаттығулар

1. Python программалау тілінде сөздік қалай анықталады?
2. Сөздік қандай мәліметтер типіне жатады?
3. Жолдар, тізімдер, кортеждер типті мәліметтерден өзгешелігі қандай? Ұқсастығы бар ма?
4. Python программалау тіліндегі сөздіктің жазылу форматы қандай?
5. Сөздікке жаңа мән қалай қосылады? Сөздіктергі мәндерді өзгертуге, өшіруге бола ма және ол қалай жүзеге асырылады?

6. Сөздікте екі элементті бірдей түйінді сөзбен сипаттауға бола ма?

7. Сөздікті элементтерден қалай тазалауға болады?

8. Сөздіктегі элементтер санын қалай есептеуге болады?

9. Сөздіктің көшірмесін алу қандай әдістің көмегімен орындалады?

10. Сөздік элементтеріндегі өзгерістерді қандай әдістің көмегімен білуге болады?

11. Қандай әдістің көмегімен сөздіктегі көрсетілген түйінді сөздерді мәндерімен шығаруға болады?

12. setdefault() әдісі қандай жағдайларда пайдаланылады?

13. Univer айнмалысымен байланысты сөздік құрып, бір факультеттегі мамандықтар бойынша әрбір курстағы студенттердің санын бейнелейтін мәліметтермен толтырыңдар. Сөздікке төмендегідей өзгерістер енгізілсін.

А) бір курстағы студенттердің санына өзгеріс енгізілсін;

Ә) бір курсқа жаңа топ қосылсын;

Б) бір курстағы бір мамандық жабылып, студенттер таратылсын (жойылды);

В) Факультеттегі студенттердің жалпы санын есептеңдер.

14. Түйінді сөздері сан, ал мәндері жолдар болып келетін сөздік құрыңдар. Оған items () әдісін пайдаланыңдар, алынған объектіні dict_items бастапқы сөздікке кері болатын жаңа сөздікті құратын функцияға беріңдер. Бастапқы сөздікке кері дегеніміз: түйінді сөздері жолдар, ал мәндері сандар болып табылады.

13. ФАЙЛДАР

Python тілінде файлдармен жұмыс істеу мүмкіндігі қарастырылған, ол орнатылған негізгі объектілер типіне жатады. Файл дегеніміз – тұрақты есте сақтау құрылғысының атау берілген бөлігі. Файлдармен жұмыс істеуді операциялық жүйе басқарады. Әрбір файл үшін объект құрылып, файлмен жұмыс істеуді қамтамасыз етеді, файл компьютердің есте сақтау құрылғыларының (винчестер, флеш диск және т.б.) кез келген бөлігіне орналасуы мүмкін. Кез келген файлдың атауы болады, ол толық немесе қысқартылған адреспен көрсетілуі мүмкін. Файлдың адресі толық болған жағдайда, ол файлға баратын толық маршрутты қамтиды:

- файл орналасқан есте сақтау құрылғысы (C немесе D дискісі болуы мүмкін);
- кері слэш (\) арқылы бөлінген каталогтар тізімі (жоғары деңгейден төменгі деңгейді қамтитын).

Қысқартылған адрес тікелей файлдың тек атын ғана көрсетеді. Python тілінде жазылған программалардың бәрі бір папкада болса, сол папка атауы мен файл аты көрсетіледі. Мысалы:

Python\students_score.py

Мұндағы, **Python** – папка аты, ал, **students_score.py** – файлдың аты.

Файлдың аты екі бөліктен тұрады: бірінші бөлік нүктеге дейін файлдың атын көрсетсе, нүктеден кейінгі екінші бөлігі – файлдың типін көрсетеді.

Файл кез келген ақпаратты қамтиды, ол әртүрлі программалық құралдардың көмегімен форматына байланысты құрылып, өңделеді. Файл форматы дегеніміз – бұл ақпаратты ары қарай тиімді пайдалану және сақтау мақсатында орындалатын оны кодтау тәсілі. Әртүрлі программалық құралдарда пайдаланылатын файлдардың форматы әртүрлі болады. Мысалы, жоғарыда келтірілген файлдың аты екі бөліктен тұрады: бірінші бөлік нүктеге дейін файлдың атын көрсетсе, нүктеден кейінгі екінші бөлігі – файлдың типін көрсетеді.

*. **py** – файлдың Python программалау тілінде жазылған программа екенін білдіреді, тіл атауының бастапқы екі әрібінен алынған. Мұны файлдың кеңейтілімі немесе форматы деп атайды. Мысалы, Ms Word мәтіндік процессорындағы құжаттың форматы - *.doc, MS PowerPoint құжаты - *.ppt және т.с.с. болуы мүмкін. Дәл осы тәрізді барлық программалық құралдарда пайдаланылатын файлдардың форматы әртүрлі болады және олар стандартталып, жарияланған.

Python программалау тілінде файлдармен жұмыс істеудің бірқатар ерекшеліктері бар.

Python тіліндегі файлдар сандарға, тізбектерге немесе бейнелерге жатпайды.

Файлда сақталған мәліметтер жолдар түрінде бейнеленеді. Сондықтан мәліметтерді объектілердің басқа да типтерінде ұсыну үшін оны түрлендіру (convert) жасау қажет. Бұл мәліметтерді файлдан оқығанда және файлға жазғанда ескеріледі. Егер файлға мәлімет жазу қажет болса, онда өңдеу әдістерінің көмегімен қалыптасқан жол түрінде беру керек.

Мәліметтерді енгізу және шығару кезеңінде файл буферге орналастыруды қамтамасыз етеді. Файлға шығару аралық буферлердің комегімен жүзеге асырылады. Егер мәліметтерді файлға жазу операциясы орындалатын болса, онда алдымен мәліметтер буферге орналасады да, одан кейін файлға жазылуы мүмкін. Сонымен бірге, мәліметтерді файлға жазғанда және файлдан оқығанда орынды өзгертуге болады.

13.1. Python тіліндегі файлдардың типтері

Python программалау тілінде басқа программалау тілдеріндегі тәрізді файлдардың екі типі ерекшеленеді: мәтіндік және бинарлық (екілік).

Мәтіндік тип. Бұл str типті жол түрінде ұсынылған файлдар. Python программалау тілінде мұндай жолдарды

Unicode символдарына автоматты түрде кодтау және одан шығару жүзеге асырылады және оған сәйкес жолдың соңы да өңделеді.

Бинарлық (екілік) тип. Бұл bytes типті жолдар түрінде ұсынылған файлдар. Мұндай жолдарды файлға жазғанда қосымша өңдеусіз жібереді. Сондай-ақ, файлдан оқығанда да жолдар өңделмейді.

Файлдармен орындалатын барлық операцияларды үш кезеңге бөлуге болады:

- Файлды ашу;
- Оқу және жазу операцияларын орындау;
- Файлды жабу.

13.2. Мәліметтерді файл арқылы енгізу

Мәліметтерді енгізіп, шығарудың екінші бір тәсілі – берілгендерді файлдан оқу және нәтижені файлға жазу болып табылады. Берілгендерді файлдан оқу үшін кез келген қарапайым мәтіндік редакторда мәтіндік файл құрып, берілгендерді сақтау қажет және ол файл программа мәтіні орналасқан бумада болуы тиіс. Мысалы, екі бүтін санның қосындысын және көбейтіндісін табуды қарастырайық. Ол үшін берілген екі санды «**input1.txt**» деп аталатын файл құрып, сол файлға сақтаймыз.

Жаңа файл ашып, программа кодын жазамыз. Ең алдымен **file** дескрипторынан тұратын айнымалы қажет. Файл **open ()** функциясының көмегімен ашылады. Файлдан оқу үшін r (**read**) режимі пайдаланылады. Келесі пайдаланылатын параметр - кодтау (**encoding**) параметрі, кодтаудың бірнеше түрі бар, біз utf8 кодтау түрін аламыз. Әдетте мәліметтерді енгізу үшін **input** функциясы пайдаланылса, мәліметтерді файлдан оқу үшін **readline** әдісі пайдаланылады. Программа нәтижесін экранға шығарайық.

1-мысал. Мәтіндік файлда екі бүтін сан сақталған. Осы сандарды жеке-жеке оқып, осы сандардың қосындысын экранға шығаратын программа құрайық.

```
fin = open('input1.txt', 'r', encoding='utf8')
x = int(fin.readline())
y = int(fin.readline())
print(x+y)
```

Сонымен, readline – input функциясына ұқсас, ол мәліметтерді файлдан оқуға арналған және әрбір жолдан бір элементтен оқиды. Барлық жолдагы элементтерді толық оқуға бола ма? Барлық жолдағы элементтерді толық оқып, оны тізімге орналастыруға болады, ол үшін **readlines** әдісі пайдаланылады. Алымен бір тізім құрып, сол тізімге барлық элементтерді орналастыруға болады.

2-мысал. Мәтіндік файлға сақталған элементтерді оқып, оны экранға шығаратын программа құрайық (1-тәсіл).

```
fin = open('input1.txt', 'r', encoding='utf8')
list = fin.readlines()
print(list)
```

Программаның орындалу нәтижесінен курсорды «келесі жолға көшіру» белгісінің (\n) де басылып шыққанын көруге болады:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pytho
n36-32/Gulira Python/list file2.py
['5\n', '7\n']
>>>
```

Бұл белгіні алып тастап үшін жолдарға арналған **strip** әдісін пайдалануға болады. Тізімнің әрбір элементі үшін **strip** әдісі жазылады.

2-мысал. Мәтіндік файлға сақталған элементтерді оқып, оны экранға шығаратын программа құрайық (2-тәсіл).

```
fin = open('input1.txt', 'r', encoding='utf8')
list = fin.readlines()
print(list[0].strip(), list[1].strip())
```

Программаның орындалуы:

```
RESTART:
C:/Uscrs/Asus/AppData/Local/Programs/Python/Pyt
hon36-32/Gulira Python/list file2.py
5  7
>>>
```

Сонымен, **strip()** әдісі барлық бос орын символдарын қиып алуға арналған. Бос орын символдарына: бос орын, табуляция, курсорды келесі жолға көшіру жатады.

readlines() әдісі мәліметтерді оқып, компьютердің жадысына орналастырады. Егер оқылатын файлдың көлемі үлкен, ал оперативті жадының шамасы аз болса, онда орын жетіспеу мәселесі туындайды.

Тізімнің барлық элементтерін компьютердің жадысына бірден сақтамау үшін for операторын пайдалану арқылы элементті оқығаннан кейін қажетті операция орындауды жүзеге асыруға болады. Бұл әдіс барлық тізім жолдарын оперативті жадыда сақтауды талап етпейді.

3-мысал. Мәтіндік файлдағы барлық мәліметтерді оқып, олардың мәнін екі еселеп экранға шығаратын программа құрайық.

```
fin = open('input1.txt', 'r', encoding='utf8')
for k in fin:
    print(int(k) * 2)
```

Программаның орындалуы:

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Pyth
on36-32/Gulira Python/list file2.py
10
14
>>>
```

Мәліметтерді файлдан оқып қана қоймай, нәтижені файлға жазуға болады. Алдымен файл **open** функциясының көмегімен ашылады. Файлға жазу үшін **w (write)** режимі пайдаланылады және кодтау тәсілі ретінде **utf8** тәсілін көрсетеміз (мысалы, мәтінді файлға қазақ тілінде жазып шығару үшін бұл кодтау тәсілін көрсету міндетті). Енгізу файлындағы сандарды оқып, олардың қосындысын тауып, нәтижені файлға жазайық. Файл үшін **file** атаулы параметр, **fout** дескриптор аты пайдаланылады. Егер файл бұрын құрылмаған болса, жаңа файл құрылады. Ал файл бұрын құрылған болса, файлдың бұрынғы мәні өшіріліп, оның орнына жаңа мән жазылады.

4-мысал. Мәтіндік файлда сақталған мәліметтерді файлдан оқып, олардың қосындысын тауып, оның нәтижесін файға сақтайтын программа құрайық.

```
fin = open('input1.txt', 'r', encoding='utf8')
fout = open('output.txt','w', encoding ='utf8')
print(sum(map(int, fin.readlines())), file=fout)
fout.close()
```

Операциялық жүйеде файлға жазу бірден орындалмайды. Ол буферге жиналып, уақыт өткен сайын босатылып отырады. Осы процесс нақты орындалу үшін **close**() әдісі пайдаланылады. Дәлірек айтқанда, файлды жабады. Программа кодынан көріп отырғанымыздай, нәтиже экранға шығарылмай, **'output.txt'** файлына жазылады.

Программаның орындалу нәтижесі:

'input1.txt'	'output.txt'
5 7 4	16

Сонымен, мәліметтерді тек клавиатурадан ғана енгізіп қоймай, сонымен бірге, файлдан оқып, нәтижені файлға жазуға болады екен.

Файлды ашу режимдерінің толық тізімі 12.1–кестеде келтірілген.

Кесте 12.1.

Файлдармен жұмыс істеу режимдері

Режим түрі	Сипаттамасы
r	Файлдан тек оқу үшін
w	Файлға тек жазу үшін. Егер көрсетілген атаумен файл табылмаса, сол атаумен жаңа файл құрады.
rb	Файлдан тек оқу үшін пайдаланылады (бинарлық файлға арналған)
wb	Файлға тек жазу үшін пайдаланылады және бинарлық файлға арналған. Егер көрсетілген атаумен файл табылмаса, сол атаумен жаңа файл құрады.
r+	Файлдан оқу және жазу үшін пайдаланылады.
rb+	Файлдан оқу және жазу үшін пайдаланылады (бинарлық файлға арналған)
w+	Файлды оқу және жазу үшін пайдаланылады. Егер көрсетілген атаумен файл табылмаса, сол атаумен жазу үшін жаңа файл құрады
wb+	Файлды оқу және жазу үшін пайдаланылады (бинарлық файл). Егер көрсетілген атаумен

	файл табылмаса, сол атаумен жазу үшін жаңа файл құрады
a	Файлды жаңа мазмұнды қосу үшін ашады. Егер көрсетілген атаумен файл табылмаса, сол атаумен жазу үшін жаңа файл құрады
a+	Файлды жаңа мазмұнды қосу үшін ашады. Егер көрсетілген атаумен файл табылмаса, сол атаумен жазу және оқу үшін жаңа файл құрады
ab	Файлды жаңа мазмұнды қосу үшін ашады (бинарлық файл үшін). Егер көрсетілген атаумен файл табылмаса, сол атаумен жазу үшін жаңа файл құрады
ab+	Файлды жаңа мазмұнды қосу үшін ашады (бинарлық файл үшін). Егер көрсетілген атаумен файл табылмаса, сол атаумен жазу және оқу үшін жаңа файл құрады

5-мысал. Мәтіндік файлдан үш бүтін санды оқып, олардың үлкенін тауып, нәтижені файлға жазатын программа құрайық.

```
# үш санның максимумын тауып файлға жазу
data=[]
f = open('in.txt', 'r')
data = f.readlines()
k = max(data)
f1=open('out1.txt','w')
f1.write(str(k))
f1.close()
f.close()
```

6-мысал. Мәтіндік файлды жазу үшін ашып, 10-нан 20-ға дейінгі сандардың квадраттарын осы ашылған файлға сақтайтын программа құрайық.

```
my_file = open("some1.txt", "w")
for i in range(10,21):
```

```
    my_file.write(str(i**2))
    my_file.write(' ')
my_file.close()
```

7-мысал. Мәтіндік файлда сақталған оқушылардың ұпайларына сәйкес олардың бағасын мәтіндік файлға жазатын программа құрайық.

```
#балл саны бойынша оқушының бағасын есептеу
f = open('in1.txt', 'r', encoding='utf8')
bal = f.readlines()
f1=open('out.txt','w', encoding='utf8')
for i in range(len(bal)):
    if int(bal[i])>=86 and int(bal[i])<=100:
        f1.write('Бағасы=5, өте жақсы; ')
    elif int(bal[i])>=64 and int(bal[i])<86:
        f1.write('Бағасы=4, жақсы;  ')
    elif int(bal[i])>=50 and int(bal[i])<64:
        f1.write('Бағасы=3, қанағаттанарлық; ')
    elif int(bal[i])>0 and int(bal[i])<50:
        f1.write('Бағасы=2, қанағаттанғысыз;  ')
    else:
        f1.write('қате!  ')
f1.close()
f.close()
```

Программаның орындалуы:

'in1.txt'	'out.txt'
85	Бағасы=4, жақсы;
75	Бағасы=4, жақсы;
55	Бағасы=3, қанағаттанарлық;
80	Бағасы=4, жақсы;
90	Бағасы=5, өте жақсы;
70	Бағасы=4, жақсы;
60	Бағасы=3, қанағаттанарлық;
45	Бағасы=2, қанағаттанғысыз;

Қазақ әріптерін файлға жазу үшін **encoding='utf8'** типін көрсету міндетті екенін жоғарыда айтқанбыз.

8-мысал. Берілген үш бүтін санды клавиатурадан енгізіп, файлға сақтайтын және олардың ең үлкенін тауып, нәтижені файлға жазатын программа құрайық.

```
#максимумды процедураны қолданып тауып, оны файлға жазу
def input1():
    a=int(input("a= "))
    b=int(input("b= "))
    c=int(input('c= '))
    max(a,b, c)
def save(a,b,c,max):
    fin=open('mfile1.txt','w')
    fin.write(str(a))
    fin.write(' ')
    fin.write(str(b))
    fin.write(' ')
    fin.write(str(c))
    fin.write(' ')
    fout=open('mfile2.txt','w')
    fout.write(str(max))
def max(a, b,c):
    if a > b and a>c:
        max=a
    elif b>a and b>c:
        max=b
    else:
        max=c
    save(a,b,c,max)
input1()
```

Программаның орындалуы:

```
RESTART: C:/Ergalym/Project level/max табу School/phyton/Files/файлдар/С проңедурамен.ру
a= 5
b= 7
c= 3
```

'mfile1.txt'	'mfile2.txt'
5 7 3	7

9-мысал. Екі өлшемді массивтегі баған элементтерінің қосындысын есептеп, нәтижені файлға жазатын программа құрайық.

```
# Массив бағандарының қосындысын есептеп, файлға жазу
n = int(input("n = "))
m = int(input("m = "))

a = []
for i in range(n):
    b = []
    for i in range(m):
        b.append(int(input()))
    a.append(b)

for i in range(n):
    for j in range(m):
        print(a[i][j], end = ' ')
    print()

c = []
for i in range(m):
    s = 0
    for j in range(n):
```

```
        s += a[j][i]
    c.append(s)
print(*c)
f1=open('mfile2.txt','w')
for i in range(len(c)):
    f1.write(str(c[i]))
    f1.write(' ')
f1.close()
```

Программаның орындалуы:

```
RESTART:
C:\Ergalym\Project School\phyton\Files\файлдар\C level\массив бағандарының  қосындысыны файлға жазу.py
n = 3
m = 2
7
8
9
4
5
6
7 8
9 4

5 6 21 18
```

'mfile2.txt'

```
21 18
```

10-мысал. Нақты сандардан тұратын кортеж берілген. Осы кортеж элементтерін мәтіндік файлға жазып және оны сол файлдан оқып, екінші кортеж құратын программа құрайық.

```
# 1. Нақты сандардан құралған бастапқы кортеж
T1 = ( 7.8, -3.65, 12.5, 21.05)

# 2. Кортежді файлға жазу
```

```
    # 2.2. кортеж элементтерін жазатын мәтіндік файл ашу
    f1 = open('myfile5.txt', 'wt')

    # 2.3. Кортеж элементтерін файлға жазу үшін циклды пайдалану
    for item in T1:
        # 2.3.1. item бүтін санды жолға түрлендіру
        s = str(item)

        # 2.3.2. символдардың арасынан бос орын қалдыру
        f1.write(s + ' ')

        # 2.4. Файлды жабу
    f1.close()

    # 3. Нақты сандардан тұратын файлдан мәліметтерді оқу

    # T2 жаңа кортежін құру
    f = open('myfile5.txt', 'rt')

    # 3.1. T2 кортежін құру
    T2 = ()

    # 3.2. Файлдан мәліметтерді оқып, кортеж құру
    for lines in f: # циклды пайдалану
        # 3.2.1. Разбить строку lines жолын strings бөліктеріне бөлу.
        # жолдың әрбір бөлігі strings[i] – бұл жолдық форматпен
        # берілген нақты сан
        strings = lines.split(' ')

    # 3.2.2. жолдың барлық бөлігін түрлендіріп, оны T2 кортежіне жазу
    for s in strings:
```

```
        if s != '': # егер бос жол болмаса, оны нақты санға түрлендіру
            num = float(s)

            # кортежге жалғау
            T2 += (num,)

    # 3.3. кортежді тексеру үшін экранға шығару
    print("T2 = ", T2) # Оның мәні: T2 = (3.8, 2.77, -11.23, 14.75)

    # 3.4. Файлды жабу
    f.close()
```

Бұл программада сөздердің арасына бос орын қою үшін split әдісі пайдаланылды.

11-мысал. Бүтін сандардан құрылған жиын элементтері берілген. Жиын элементтерін файлға жазып, оны қайтадан оқып, экранға шығаратын программа құрайық.

```
# Жиын элементтерін мәтіндік файлға жазу және оқу.
# 1. Бүтін сандар жиыны берілген.
M = { 2, 3, -12, 22, 38 }
# 2. Жиынды файлға жазу
# 2.1. Мәтіндік файлды жазу үшін ашу
f = open('myfile7.txt', 'w')
# 2.2. Жиын элементтерін фалға жазу цикл арқылы орындалады
for item in M:
    # 2.2.1. Жиын элементтерін жолға + '\n' - түрлендіру
    s = str(item) + '\n'
    # 2.2.2. Жолды файлға жазу
    f.write(s)
# 2.3. Файлды жабу
f.close()
# 3. Файлдан жиын элементтерін оқу
```

```
# 3.1. Файлды оқу үшін ашу
f = open('myfile7.txt', 'rt')
# 3.2. Босжиын құру
M2 = set()
# 3.3. Файлдан мәліметтерді оқу және жаңа жиын құру
for line in f: # Файлдан мәліметтерді оқу үшін циклды пайдалану
    # line жолын бүтін санға түрлендіру
    num = int(line)
    M2 = M2.union({num})
# 3.4. Жиын элементтерін экранға шығару
print("M2 = ", M2) # M2 = {2, 3, -12, 38, 22}
# 3.5. Файлды жабу
f.close()
```

Кесте 13.2.

Python тілінде файлдарға пайдаланылатын әдістер

Әдістер	Сипаттамасы
file.close()	Ашылған файлды жабады
file.fileno()	Файлдың бүтін типті дескрипторын береді
file.flush()	Ішкі буферді тазалайды
file.isatty()	Егер файл терминалмен жалғасқан болса, TRUE мәнін береді
file.next()	Файлдың келесі жолын береді
file.read(n)	Файлдың алғашқы n символын оқиды
file.readline()	Жолдардың немесе файлдың бірінші жолын оқиды
file.readlines()	Файлдағы барлық жолдар тізімін оқып, шығарады.
file.seek(offset[when])	Файлға ағымдағы орынды тағайындайды

file.seekable()	Файлдың кездейсоқ қатынауды сүйемелдеуін тексереді. Егер сүйемелдесе True мәнін береді
file.tell()	Файлда ағымдағы позицияны корсетеді
file.truncate(n)	Файлдың өлшемін кішірейтеді. Егер n көрсетілсе, n байтқа дейін қысқартылады. Егер n көрсетілмесе, ағымдағы позицияға дейін қысқартады.
file.write(str)	str жолын файлға қосады
file.writelines(sequence)	Жолдар тізбегін файлға қосады

Бақылау сұрақтары мен жаттығулар

1. Файл дегеніміз не?
2. Файлдың толық маршруты деген нені білдіреді?
3. Файлдың форматы дегенді қалай түсіндіңдер?
4. Python программалау тілінде файлдың қандай типтері пайдаланылады?
5. Программалау тілдерінде файл не үшін пайдаланылады?
6. Мәтіндік файл мен екілік файлдардың арасында қандай айырмашылық бар?
7. Python программалау тілінде файл қалай құрылады? Файл құрудың қандай тәсілдері бар?
8. Файл құруда кодтау тәсілін таңдаудың маңыздылығы қандай?
9. Файлдан бірнеше элементті оқу үшін қандай әдіс пайдаланылады?
10. **strip()** әдісінің қызметі қандай?
11. "text.txt" деп аталатын мәтіндік файл берілген. Файлдағы барлық мәліметті ошіріп, оның орнына «Сәлем

баршаңызға! Біз программалау тілін үйренудеміз!!!» деген мәтінді қазақ тілінде файлға сақтаңыз.

12. Дөңес төртбұрыш берілген, оның барлық қабырғалары мен бір диагоналінің ұзындығы белгілі, оның ауданын табатын программа құрыңдар. Барлық мәліметтер файлдан оқылып, нәтиже файлға сақталсын.

13. Тізім түрінде берілген апта күндерін файлға жазып, оны қайтадан оқып, нәтижесін экранға шығаратын программа құрыңдар.

14. Матрица берілген. Матрица элементтерін файлға жазып, оны қайтадан оқып, экранға шығаратын программа құрыңдар.

15. Фибоначчи санының n-ші реттік номерін файлдан оқып, сол n-ші мүшеге дейінгі Фибоначчи тізбегін сол файлға жазатын программа құрыңдар.

16. n элементтен тұратын тізі берілген. Тізім элементтерін файлдан оқып, олардың ең кішісін тауып, оны файлға жазатын программа құрыңдар.

17. Нақты сандардан тұратын матрица берілген. Матрицаның жолдық элементтерінің үлкенін тауып, оны тізім ретінде файлға сақтайтын программа құрыңдар.

18. Нақты сандардан тұратын матрица берілген. Матрицаның жанама диагоналінде орналасқан элементтерін тауып, экранға шығаратын және оны тізім ретінде файлға сақтайтын программа құрыңдар.

19. Мәтіндік файлда сақталған оқушылардың ұпайларын файлдан оқып, оны кему реті бойынша сұрыптап, нәтижені файлға жазатын программа құрыңдар.

20. Мәтіндік файлда сақталған оқушылардың ұпайларын файлдан оқып, ұпай саны бойынша оларға шәкіртақы тағайындайтын, оның нәтижесін файлға жазатын және экранға шығаратын программа құрыңдар.

14. PYTHON ТІЛІНДЕГІ ГРАФИКА

14.1. PyGame кітапханасы. Кітапхананы қосу

Pygame - бұл «ойын кітапханасы», программистерге ойын құруға көмектесетін құралдар жиынтығы болып табылады. Оның құрамына:

- Графика және анимация
- Дыбыс (музыканы қоса алғанда)
- Басқару элементтері (тінтуір, пернетақта, геймпад және т.б.) жатады.

Кітапхана (библиотека) – бұл функциялар мен кластар кодтарының жиынтығы. Python программалау тілінде код кітапханасын сипаттау үшін қолданылатын термин – модуль (module) деп аталады. Мысалы, pygame модулінде ішкі модульдер ретінде pygame.draw, pygame.image, pygame.mouse және т.б. модульдер пайдаланылады.

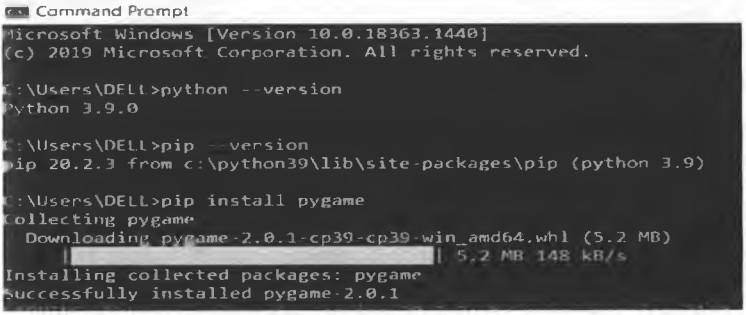
PyGame кітапханасы – 2D ойындарын және басқа мультимедиалық қосымшаларды әзірлеуге арналған Python программалау тілінің модулі болып есептеледі.

PyGame өте ыңғайлы болғанымен Python-ның стандартты кітапханасына кірмейді, дәлірек айтқанда, ол Python тілінің стандартты бумасынан бөлек болғандықтан, оны бөлек орнату қажет. Ол үшін пайдаланушының компьютерінде орнатылған Python тілінің нұсқасына сәйкес **PyGame** кітапханасы www.pygame.org сайтынан жүктеледі. Ол үшін операциялық жүйенің командалық жолына **cmd** командасын енгіземіз. Нәтижесінде командалық жол терезесі ашылады (14.1-сурет). Осы терезеге Python сөзін енгіземіз.

Сурет 14.1. Командалық жол терезесі

Нәтижесінде пайдаланушының компьютерінде орнатылған Python программалау тілінің нұсқасы туралы ақпарат шығады. Бұл жүйенің компьютерге дұрыс орнатылғанын көрсетеді (14.2-а-сурет). Бұдан **<CTRL+Z>** командасын орындау арқылы программалау жүйесінен шығамыз.

Келесі қадамда командалық жолға **pip install pygame**_командасын енгізіп, **Enter** клавишын басамыз.



Сурет 14.2-а. Pygame кітапханасын орнату терезесі

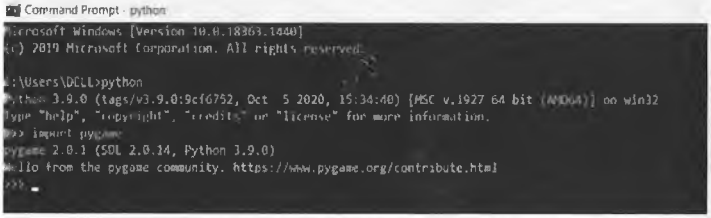
Pygame кітапханасы орнатылады, одан кейін командалық жолға

>> python

сөзін енгізіп, Enter клавишын басамыз. Нәтижесінде жүйе туралы ақпарат шығады (14.2-ә-сірет). Одан кейін

>> import pygame

командасы арқылы pygame импортталады.



Сурет 14.2-ә. Pygame кітапханасын импорттау терезесі

PyGame кітапханасын импорттағаннан кейін оны орнату керек:

pygame.init () – графикага пайдаланылатын ішкі модульді орнатады, драйверлерді жүктеп, PyGame компьютердің барлық құрылғыларын пайдалануға дайындайды.

Pygame модуліне байланысты құжаттар, оқулықтар және қосымша ақпаратты: https://www.pygame.org/docs/ веб-сайтынан көруге болады.

Python жүйесінде программа құруда Pygame кітапханасын пайдалану оны импорттаудан басталады. Кітапханамен танысуды қарапайым қосымша мысалды талдаудан бастайық.

1-мысал. Графикалық бейнелерді көрсететін экранды шығаратын программа құрайық.

```
import pygame
import sys

pygame.init()
```

```
screen = pygame.display.set_mode((520, 180))

pygame.display.set_caption("My Game")

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
```

Программаның орындалу нәтижесі (14.3-сурет).

Сурет 14.3. Графикалық бейнелерді шығару терезесі

Алдымен pygame модулін бастапқы файлға қосу қажет:

import pygame

Одан кейін **init ()** функциясын Pygame модульдерін жұмысқа дайындау үшін шақырамыз:

pygame.init()

Графикамен жұмыс істеу, ең алдымен, терезені құру, яғни графика шығарылатын алаңды дайындаудан басталады, бұл біздің жағдайда терезенің өлшемдерін көрсету болып табылады. Ол PyGame кітапханасында экранға режимді тағайындайтын

screen = pygame.display.set_mode((520, 180))

дайын модулі арқылы жүзеге асырылады.

Кез келген геометриялық фигуралар декарттық координаталар жүйесінде жататыны белгілі. Сондықтан программалау тілдерінде барлық салынатын объектілер декарттық координаталар жүйесіне орналастырылады. Программалау тілдерінде координаталар жүйесінің бас нүктесі (x = 0 және y = 0) – экранның сол жақ жоғарғы бұрышында орналасады және **x** осі экранның оң жақ бағытына қарай, **y** ось экранның төменгі бағытына қарай өседі (14.4-сурет).

Сурет 14.4. Координаталар жүйесі

1-мысалдағы

screen = pygame.display.set_mode((1200, 800)),

экранның өлшемін x=520, y=180 көрсетеді.

Ойын терезесінің атауын «My Game» атымен беру үшін

pygame.display.set_caption("My Game")

функциясын пайдаланамыз..

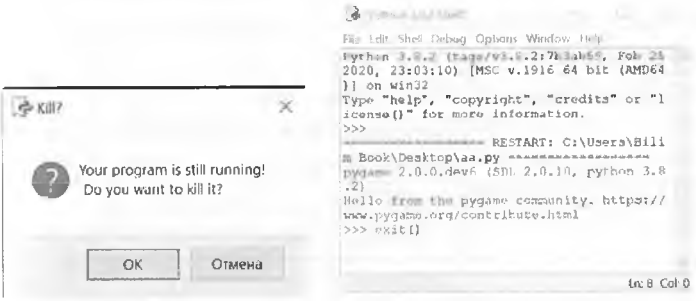
Содан кейін программада пайдаланушының негізгі графикалық терезесін жабу оқиғасын жүзеге асыратын цикл орналасады:

```
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
```

Программаның бұл бөлігінде pygame кітапханасынан шығып, жұмысты аяқтау орындалады. Ол pygame.quit () функцнясын және sys модулінен exit () функциясын шақыру арқылы жүзеге асады (14.5-сурет).

Егер төмендегі цикл пайдаланылмаса, онда ойын терезесі өшіру батырмасын басқан кезде бірден жабылмайды:

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.quit()
        sys.exit()
```



Сурет 14.5. Программаның нәтижесін көру және оны жабу терезесі

14.1-кестеде Pygame модульдері мен олардың атқаратын қызметі келтірілген.

Кесте 14.1.

Pygame модульдері мен олардың қызметі

Модуль атауы	Қызметі
import pygame	PyGame кітапханасын іске қосу
Pygame.init()	Pygame құралдарындағы init() функциясын орнату
pygame.display.set_mode (500,300))	500X300 өлшемді терезе құру.
pygame.display.set_caption (“My Game”)	«My Game» атауымен терезе тақырыбын тағайындау.
Clock=pygame.time.clock()	Time Уақыт модулі берілген функцияның алғашқы шақыруынан кейінгі өткен уақытты береді.
Pygame.display.update	Ойын терезесінің мазмұнын жаңартып отыру.

for i in pygame.event.get():
if event.type==pygame.QUIT:
quit()
	Оқиғаны бақылау циклы. Pygame модуліндегі жұмысты аяқтау үшін quit() функциясы пайдаланылады.

screen.fill([255,255,255])	экранды түске бояу

14.2. PyGame кітапханасының дайын модульдерімен танысу

14.2.1. Surface класы мен blit() әдісі

Қосымша беттерді **pygame. Surface** класы арқылы жасауға болады. Осыдан кейін оларды pygame.display. set_mode () әдісімен құрылған басты бөлікке немесе оларды бірінің үстіне бірін ауыстыра отырып орналастыруға болады. Оларды сызу **blit ()** әдісі арқылы жүзеге асырылады.

pygame беттері тек pygame.display.set_mode () функциясын шақыру немесе **surface** класының конструкторын тікелей шақыру арқылы ғана құрылмайды. Сонымен бірге, басқа да бірқатар функциялар мен әдістерді орындау нәтижесінде құрылады. Бұл беттердің маңызды роль атқаратындығымен байланысты, өйткені олар ең соңында экранда бейнеленеді. Оларды анимациядағы қабаттармен салыстыруға болады. Кластан тікелей **surface** экземплярын құрғанда биіктігі мен енін міндетті түрде көрсету қажет, бұл set_mode() әдісін шақырғанға ұқсас. Мысалы,

surf = pygame.Surface((100,100))

Ал, **blit()** әдісі беттің үстіне бетті қабаттастырып орналастыратын жағдайларда пайдаланылады, дәлірек айтқанда, бейне беттің үстіне салынған кезде пайдалаланылады. Басқаша айтқанда, бұл әдіс негізгі **Surface** класына пайдаланылады және қосалқы аргумент ретінде беріледі. Сондай-ақ, әдіске қосалқы бет орналасатын координаталар жүйесіндегі негізгі беттің сол жақ жоғарғы бұрышының координатасын беру қажет. Мысалы,

sc.blit(surf (50, 25)

Мұндағы sc – негізгі бет, blit() әдісінің көмегімен, sc негізгі беттің 50*25 координатасына surf беті салынады.

Event модулі – оқиғаны басқарады, PyGame нысаны, оқиғалар және кезектермен әрекеттесу үшін қолданылады.

2-мысал. Негізгі беттің үстіне қосымша бетті орналастыратын программа құрайық.

```
import pygame as pg
import sys

sc = pg.display.set_mode((300, 200))
surf = pg.Surface((200, 150))
surf.fill((255, 255, 255))
sc.blit(surf, (50, 25))
pg.display.update()

while 1:
    for i in pg.event.get():
        if i.type == pg.QUIT:
            sys.exit()
    pg.time.delay(100)
```

Программаның орындалу нәтижесі 14.6-суретте келтірілген.

Сурет 14.6. Негізгі бетке қосымша бетті орналастыру терезесі

14.2.2. pygame.draw модульдері

Тіктөртбұрыш салу

Rect (rectangle ағылшын сөзінен қысқартылып алынған) тіктөртбұрыш сызады.

Pygame.draw.rect(screen, color,rect, width)

Screen – бетті таңдау, Color – түсі. Егер Width=0 болса, онда фигураның іші толық боялады.

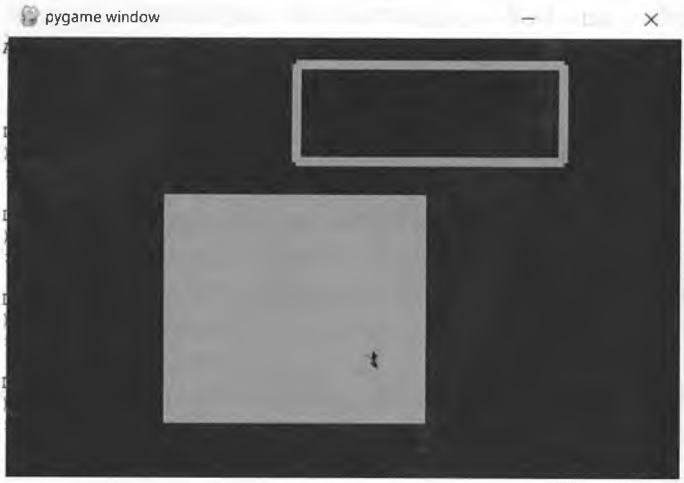
Rect[x_1, y_1, x_2, y_2] – сол жақ жоғарғы және оң жақ төменгі бұрыштарының координаталары бойынша тіктөртбұрыш салады.

Негізгі бетке орналасатын қосалқы бетке мөлдірлік қасиетін беруге болады. Ол деп аталатын әдістің көмегімен жүзеге асырылады. Мөлдірлік аргументі 0-ден 255-ке дейінгі мәндерді қабылдайды (0 – толық мөлдір).

3-мысал. Өлшемдері әртүрлі тіктөртбұрыштарды бейнелейтін программа құрайық.

```
import pygame
pygame.init()
GREEN = (0, 255, 0)
sc=pygame.display.set_mode((515,340))
pygame.draw.rect(sc, (255, 0, 255), (120, 120, 200, 175))
pygame.draw.rect(sc, (74, 128, 255), (220, 20, 205, 75),
7)
pygame.display.update()
while True:
   for i in pygame.event.get():
      if i.type==pygame.QUIT:
         exit()
pygame.time.delay(20)
```

Программаның орындалу нәтижесі 14.7-суретте келтірілген.



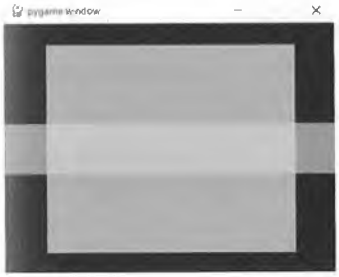
Сурет 14.7. Тік төртбұрыштарды шығару терезесі.

4-мысал. Негізгі беттің ортасына күлгін түспен тіктөртбұрыш салып, оның үстіне жартылай мөлдір бетті орналастыратын программа құрайық.

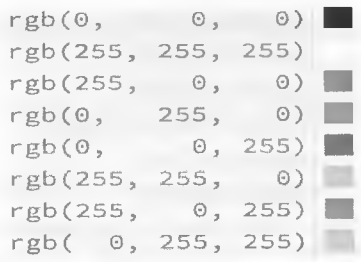
```
import pygame as pg
import sys
sc = pg.display.set_mode((400, 300))
surf = pg.Surface((300, 250))
surf.fill((255, 255, 60))
surf.set_alpha(200)
 # алдымен негізгі беттің үстіне
# күлгін тіктөртбұрыш салынады
pg.draw.rect(sc, (255, 0, 255),(0, 120, 400, 60))
# жартылай мөлдір сары түсті бет
# орналастырамыз
sc.blit(surf, (50, 25))

pg.display.update()

while 1:
   for i in pg.event.get():
```

```
    if i.type == pg.QUIT:
        sys.exit()
    pg.time.delay(100)
```

Программаның орындалу нәтижесі 14.7-суретте келтірілген.



Сурет 14.7. Жартылай мөлдір бетті шығару терезесі



Сурет 14.8. Негізгі түстер коды

14.2.3. Геометриялық фигураларды сызу

Компьютерлік экран пиксельдерден құралады, олардың әрқайсысында 3 элемент бар: қызыл (red), жасыл (green) және көк (blue). Осы түстерден құралатын түстік модель RGB деп аталады. Үш негізгі түстердің әрқайсысында 0 (өшіру) мен 255 аралығында (100% қосулы) мән болуы мүмкін, сондықтан әр элемент үшін 256 түс бар, адам көзбен осы түстерді көре алады. Python тілінде пайдаланылатын стандартты түстер коды 14.8-суретте келтірілген.

Pygame модулінде фонды түстермен толтыру үшін screen.fill() әдісі пайдаланылады. Ол үшін тек фон түсінің аргументі гана алынады. Мысалы, көк түсті фон құру үшін screen.fill(255,0,0) деп RGB түстер пикселімен көрсетеміз немесе оны бір айнымалыға меншіктеп қойып, одан кейін

сол айнымалы атауын, мысалы, screen.fill(blue) деп пайдалана беруге де болады.

5-мысал. Экранды көк түске бояйтын, ол фонның түсі болатын программа құрайық.

```
import pygame
pygame.init()
screen=pygame.display.set_mode((400,300))
pygame.display.set_caption('My Game')
blue=(0,0,255)
screen.flll(blue)
pygame.display.flip()
running=True
while running:
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            running=False
pygame.quit()
```

Программаның орындалу нәтижесі 14.9-суретте келтірілген.

Сурет 14.9. Фонның түсін өзгерту терезесі

Кесінді салу

Кесінді салу үшін оның ұштарының координаттарын көрсету жеткілікті. Бұл жағдайда line () функциясы сызықты, aaline () - тегістелген сызықты сызады (соңғысы үшін қалыңдығы көрсетілмейді):

Pygame.draw.line(screen, color, start, end, width)

pygame.draw.aaline(screen, color, start, end)

Мұндағы

Start = [x_1, y_1] – түзу сызық басталатын нүктенің координатасы.

End = [x_2, y_2] – түзу сызық аяқталатын нүктенің координатасы.

Width=1 ден бастап сызықтың ені қалыңдай береді.

6-мысал. line () функциясын пайдаланып, қабырғаларының түсі мен қалыңдығы әртүрлі болатын үшбұрыш сызатын программа құрайық (1-тәсіл).

```
import pygame
pygame.init()
LIGHT_BLUE = (64, 128, 255)
GREEN = (0, 200, 64)
YELLOW = (225, 225, 0)
sc=pygame.display.set_mode((515,340))
pygame.draw.line(sc, LIGHT_BLUE, [150, 60], [390, 120], 3)
pygame.draw.line(sc, YELLOW, [390, 120], [130, 235],4)
pygame.draw.aaline(sc, GREEN, [130, 235], [150, 60])

pygame.display.update()
while True:
   for i in pygame.event.get():
      if i.type==pygame.QUIT:
         exit()
pygame.time.delay(20)
```

Беттегі нүктелерді біріктіре отырып, кесінділерден құралған фигура сызуға болады. Ол үшін **line ()** функциясы төмендегідей форматта жазылады:

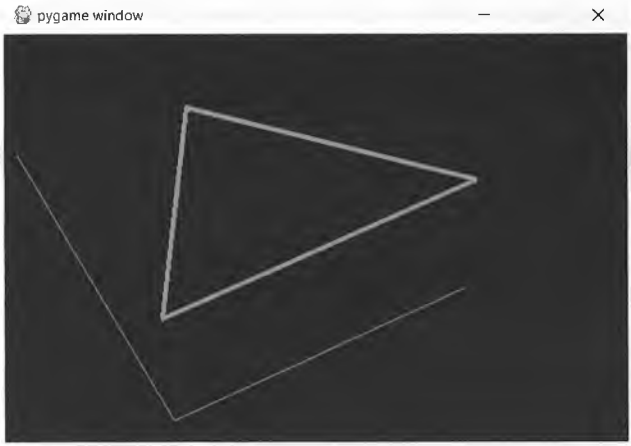
Pygame.draw.lines(screen, color, closed, pointlist, width)

Мұндағы Pointlist айнымалысы беттегі нүктелерді біріктіре отырып, кесінділер сызады. Әбір нүкте координаталар арқылы белгіленеді. Егер closed айнымалысы ақиқат мәнін қабылдаса (closed=true), онда соңғы нүкте бастапқы нүктеге қосылып, тұйық фигураға айналады.

6-мысал. line () функциясын пайдаланып, үшбұрыш сызатын программа құрайық (2-тәсіл).

```
import pygame
pygame.init()
LIGHT_BLUE = (64, 128, 255)
GREEN = (0, 200, 64)
YELLOW = (225, 225, 0)
sc=pygame.display.set_mode((515,340))

pygame.draw.lines(sc, [255,0,255], True,[[150, 60],.[390, 120],
                                        [130, 235]], 4)
pygame.draw.aalines(sc, [255,0,255], False,[[10, 100], [140, 320],
                                        [380, 210]])
pygame.display.update()
while True:
   for i in pygame.event.get():
      if i.type==pygame.QUIT:
         exit()
pygame.time.delay(20)
```

Программаның орындалу нәтижесі 14.10-суретте келтірілген.



Сурет 14.10. Сзықтарды бейнелеу

Көпбұрыш салу

Python программалау тілінде көпбұрыш салуға арналған функция Polygon () деп аталады. Көпбұрыш оның бұрыштарының төбесіндегі нүктелерінің координаталары бойынша салынады. Координаттар үзіліс нүктелерін анықтайды. Нүктелер саны ерікті болуы мүмкін.

7-мысал. polygon () функциясын пайдаланып, бес жұлдыз салатын программа құрайық.

```
import pygame
pygame.init()
sc=pygame.display.set_mode((515,340))
pygame.draw.polygon(sc, [255, 255,0], [[150, 50], [250,
200], [50,100], [260, 100], [100, 200]])
pygame.display.update()
while True:
   for i in pygame.event.get():
      if i.type==pygame.QUIT:
         exit()
pygame.time.delay(20)
```

Программаның орындалу нәтижесі 14.11-суретте келтірілген.

Сурет 14.11. polygon () функциясының орындалу нәтижесі

Шеңбер сызу

Python программалау тілінде шеңбер сызу үшін circle () функциясы пайдаланылады. Шеңбер сызу үшін оның центрі мен радиусын көрсету жеткілікті.

Pygame.draw.circle(screen, color, pos, radius, width)

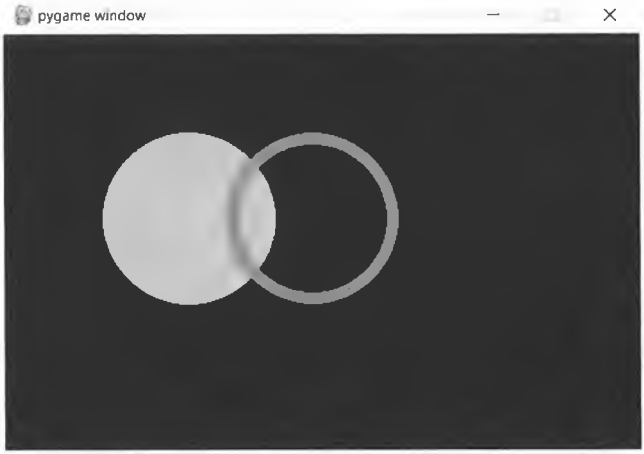
Мұндағы
Pos – шеңбердің центрінің координатасы.
Radius – шеңбердің радиусы R. Width – сызықтың қалыңдығы. Егер width=0 болса, шеңбердің іші толық боялады.

8-мысал. Бір түзудің бойында орналасқан шеңбер мен дөңгелекті бейнелейтін программа құрайық.

```
import pygame
pygame.init()
```

```
sc=pygame.display.set_mode((515,340))

pygame.draw.circle(sc, [255,255,60], (150, 150), 70)
pygame.draw.circle(sc, [0,200,64], (250, 150), 70, 10)
pygame.display.update()
while True:
    for i in pygame.event.get():
        if i.type==pygame.QUIT:
            exit()
pygame.time.delay(20)
```

Программаның орындалу нәтижесі 14.12-суретте келтірілген.



Сурет 14.12. circle () функциясының орындалу нәтижесі

Эллипс сызу

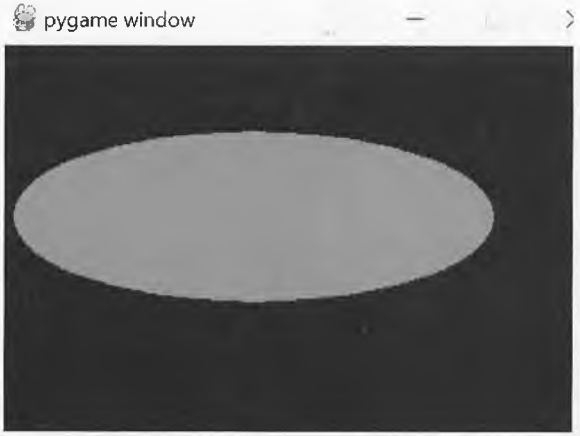
Эллипс сызу үшін **ellipse()** функциясы пайдаланылады. Эллипсті сипаттайтын төрт координатасы көрсетілуі керек:

Pygame.draw.ellipse(screen, color, rect, width)

9-мысал. Іші жасыл түспен боялған эллипсты экранға шығаратын программа құрайық.

```
import pygame
pygame.init()
GREEN = (0, 255, 0)
sc=pygame.display.set_mode((315,240))
pygame.draw.ellipse(sc, GREEN, (10, 50, 280, 100))
pygame.display.update()
while True:
   for i in pygame.event.get():
      if i.type==pygame.QUIT:
         exit()
pygame.time.delay(20)
```

Программаның орындалу нәтижесі 14.13-суретте келтірілген.



Сурет 14.13. ellipse () функциясының орындалу нәтижесі

Доға сызу

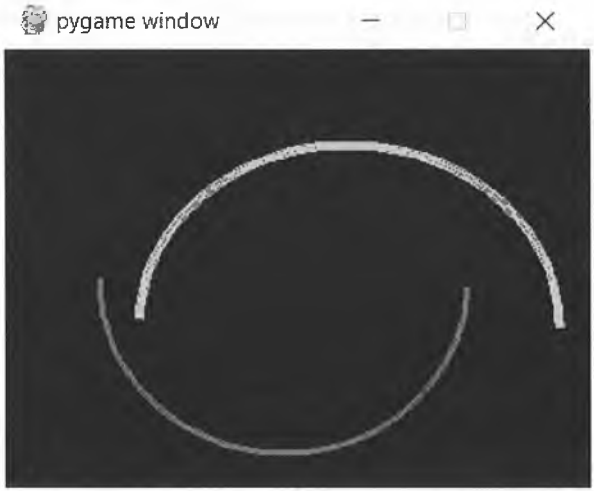
Python прогрaммалау тілінде доға сызу үшін **Pygame.draw.arc(screen, color, start, end, width)** функциясы пайдаланылады.

Мұндағы төртінші және бесінші аргументтер – радианмен көрсетілген доғаның басталуы мен аяқталуы болып табылады.

10-мысал. Доғаларды бейнелейтін программа құрайық.

```
import pygame
pygame.init()
GREEN = (0, 255, 0)
pi = 3.14
sc=pygame.display.set_mode((315,240))
pygame.draw.arc(sc, [255,255,62], (70, 50, 230, 200), 0,
pi,5)
pygame.draw.arc(sc,[0,0,255], (50, 30, 200, 190),  pi,
2*pi, 3)

pygame.display.update()
while True:
   for i in pygame.event.get():
      if i.type==pygame.QUIT:
         exit()
pygame.time.delay(20)
```

Программаның орындалу нәтижесі 14.14-суретте келтірілген.



Сурет 14.14. arc () функциясының орындалу нәтижесі

11-мысал. Жоғарыдан төмен қарай құлап келе жатқан допты бейнелейтін программа құрайық.

```
import pygame
import sys

FPS = 60
WIN_WIDTH = 400
WIN_HEIGHT = 500
Lblue = (0, 255, 255)
blue = (0, 0, 255)

clock = pygame.time.Clock()
sc = pygame.display.set_mode(
   (WIN_WIDTH, WIN_HEIGHT))

# доптың радиусы
r = 50
# допты ортасына қарай туралау
x = WIN_WIDTH // 2
# доптың координатасын жоғарғы шекарадан тыс
орналастырамыз

y = 0 - r

while 1:
   for i in pygame.event.get():
      if i.type == pygame.QUIT:
         sys.exit()

   # фонның түсін тағайындаймыз
   sc.fill(Lblue)
   # дөңгелектің суретін саламыз
   pygame.draw.circle(sc, blue,
                      (x, y), r)
   # терезені жаңартамыз
   pygame.display.update()
```

```
        # Егер доп төменгі шекарадан тысқары орналасса
        if y >= WIN_HEIGHT + r:
            # оны жоғары шекараға орналастырамыз
            y = 0 - r
        else:  # егер орналаспаса,
            # циклдың келесі қадамында дөңгелек
            # цикл қадамына сай төмен қарай орналасатын
болады
            y += 2
            clock.tick(FPS)
```

Кесте 14.2.

Pygame.draw модулінің функциялары

Қызметтер	Сиаттамалары
pygame.draw.rect	Төртбұрышты салу
pygame.draw.polygon	Көпбұрышты салу
pygame.draw.circle	Шеңбер салу
pygame.draw.ellipse	Эллипс салу
pygame.draw.arc	Эллипстің доғасын салу
pygame.draw.line	кесінді салу
pygame.draw.lines	Бірнеше іргелес сегменттерді сызу
pygame.draw.aaline	Тегіс сызықтарды сызу
pygame.draw.aalines	Тізбегі тегістеліп сызылған сызықтар

14.3. Клавиатура оқиғалары

Ойын жүйесінде адамның объектілерді басқаруы клавиатура, тышқан және джойстик арқылы жүзеге асырылады. Манипулятор арқылы қозғалысты басқару

кезінде белгілі бір типтегі оқиғалар орындалады. Оқиғаларды өңдеу pygame.event модулі арқылы жүзеге асырылатынын айттық. Сондай-ақ, осы **pygame.event** модулінде бірқатар маңызды функциялар бар, солардың бірі – болған оқиғаны кезектен шығаратын - **pygame.event.get()** болып табылады. **Pygame** модулінде қандай да бір оқиға тіркелетін болса, оған сәйкес Event класының объектісі құрылады және одан кейін программа осы объектілермен жұмыс істейді. Бұл кластың экземплярларының тек қасиеттері болады, ал әдістері болмайды. Барлық экземплярларда **type** қасиеті бар. Оқиғаның қалған қасиеттерінің жиыны осы **type** қасиетінің мәніне тәуелді.

Клавиатура оқыиғаларының екі типі бар: клавиша басылған жағдай және клавиша жіберілген жағдай, **type** осы екі мәннің бірін қабылдайды. Егер пайдаланушы клавишаны басып, қайтадан босататын болса, онда оқиғалар кезегіне екеуі де жазылады. Ал, олардың қайсысының өңделетіні программаның мазмұнына тәуелді. Егер пайдаланушы клавишаны басып, оны босатпайтын болса, онда кезекке бір ғана нұсқа – «клавишаның басылған» жағдайы ғана тіркеледі. «Клавишаның басылған» жағдайы типті оқиға бойынша **pygame.KEYDOWN** тұрақты мәніне сәйкес келетін сандық мән **type** орісіне жазылады. Ал, «клавишаның босатылған» жағдайы типті оқиға бойынша **pygame.KEYUP** тұрақты мәніне сәйкес келетін сандық мән **type** өрісіне өрісіне жазылады.

Клавиатура оқиғасының екі типіне сәйкес келетін **key** және **mod** деп аталатын атрибуттар бар. Мұндағы **key** атрибутына нақты қай клавишаның басылғаны немесе босатылғаны туралы жазылады. Ал, **mod** атрибутына клавиша басылған немесе босатылған мезеттегі **Shift** немесе **Ctrl** тәрізді түрлендіргіш клавиштар жазылады. **KEYDOWN** оқиғасында **unicode** деп аталатын өріс бар, мұнда басылған клавишаның символының типі (**str** типті мәлімет) жазылады.

13-мысал. Бағыттаушы клавиштар арқылы доптың қозғалысын басқаратын программа құрайық.

```
import pygame
import sys

FPS = 60
W = 700  # ширина экрана
H = 300  # высота экрана
WHITE = (255, 255, 255)
BLUE = (0, 70, 225)
sc = pygame.display.set_mode((W, H))
clock = pygame.time.Clock()
# координаты и радиус круга
x = W // 2
y = H // 2
r = 50
while 1:
    for i in pygame.event.get():
        if i.type == pygame.QUIT:
            sys.exit()
    sc.fill(WHITE)
    pygame.draw.circle(sc, BLUE, (x, y), r)
    pygame.display.update()
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        x -= 3
    elif keys[pygame.K_RIGHT]:
        x += 3
    clock.tick(FPS)
```

14.4. Тышқанның оқиғалары

Pygame модулінде тышқанның үш оқиғасы өңделеді:

- тышқанның батырмасын басу (оқиғаның type қасиетінің мәні pygame.MOUSEBUTTONDOWN тұрақтысына сәйкес келеді;

- тышқанның батырмасын босату (pygame. MOUSEBUTTONUP)

- тышқанды жылжыту (pygame.MOUSEMOTION)

Қандай батырма басылса да, **button** оқиғасының басқа қасиетіне жазылады. Тышқанның сол жақ батырмасы үшін – 1, ортаңғы батырмасы үшін – 2, оң жақ батырмасы үшін – 3, ортасындағыны алға қарай айналдыру үшін - 4, ал артқа қарай айналдыру үшін – 5 сандары пайдаланылады. Ал, MOUSEMOTION оқиғасында **button** орнына **buttons** пайдаланылады және оған тышқанның үш батырмасының күйі (үш элементтен тұратын кортеж) жазылады.

Тышқан оқиғаларының басқа атрибуттарының бірі – pos, оған өтіп жатқан оқиғаның координаттары жазылады (екі саннан құралған кортеж).

Егер пайдаланушы өлшемі 200*200 болатын терезенің дәл ортасын тышқанның оң жақ батырмасымен белгілесе (басса), онда Event оқиғасы құрылады, оның өрістері

Event.type = pygame.MOUSEBUTTONDOWN, event.button = 3, event.pos = (100,100)

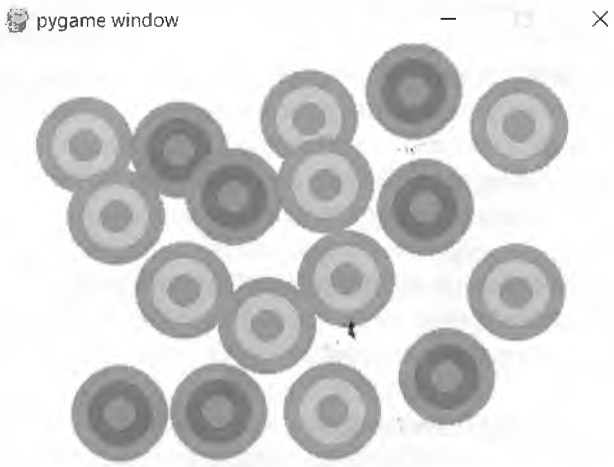
MOUSEMOTION оқиғасының **rel** деп аталатын атрибуты бар. Ол екі ось бойынша саыстырмалы ығысуды көрсетеді. Бұл атрибуттың көмегімен тышқан қозғалысының жылдамдығын бақылауға болады.

14-мысал. Тышқанның оң жақ және сол жақ батырмаларын басқанда әртүрлі концентрлі шеңберлер шығаратын программа құрайық.

```
import pygame as pg
import sys

WHITE = (255, 255, 255)
YELLOW = (225, 255, 62)
GREEN = (0, 225, 0)
BLUE = (0, 0, 225)
RED = (225, 0, 50)
PURPLE = (255,0, 255)
LIGHT_BLUE = (64, 128, 255)
```

```
sc = pg.display.set_mode((400, 300))
sc.fill(WHITE)
pg.display.update()

while 1:
    for i in pg.event.get():
        if i.type == pg.QUIT:
            sys.exit()
        if i.type == pg.MOUSEBUTTONDOWN:
            if i.button == 1:
                pg.draw.circle(
                    sc, PURPLE, i.pos, 30)
                pg.draw.circle(
                    sc, YELLOW, i.pos, 20)
                pg.draw.circle(
                    sc, RED, i.pos, 10)
                pg.display.update()
            elif i.button == 3:
                pg.draw.circle(
                    sc, LIGHT_BLUE, i.pos, 30)
                pg.draw.circle(
                    sc, BLUE, i.pos, 20)
                pg.draw.circle(
                    sc, GREEN, i.pos, 10)
                pg.display.update()
            elif i.button == 2:
                sc.fill(WHITE)
                pg.display.update()

    pg.time.delay(20)
```

Программаның орындалу нәтижесі 14.15-суретте келтірілген.



Сурет 14.15. Түрлі-түсті концентрлі шеңберлер

14.5. Шрифтпен жұмыс істеу. pygame.font модулі

Font және SysFont кластары pygame.font модулінде орналасқан, олар шрифтпен және мәтінмен жұмыс істеуге арналған. Осы аталған кластардан объект құру үшін pygame.font модулін pygame.font.init() командасы арқылы орнату қажет немесе Pygame библиотекасының барлық модулдерін pygame.init() командасы арқылы орнату қажет.

Pygame.Font және pygame.Font.SysFont кластарынан шрифт объектілері құрылады. Екінші класс жүйелік шрифт үшін пайдаланылады, сондықтан конструкторға шрифт атауын көрсету жеткілікті. Дәлірек айтқанда, Font конструкторына шрифт файлының атын көрсету жеткілікті.

Мысалы,

pygame.Font.SysFont('arial', 35)

pygame.Font.Font('/address/arial.ttf', 35)

Мұндағы екінші аргумент – пиксельмен берілген шрифтің өлшемі.

Жүйедегі шрифт түрлері туралы мағлұмат алу үшін

pygame.font.get_fonts()
функциясы пайдаланылады. Мысалы,

```
import pygame
import sys
pygame.font.get_fonts()
```

```
RESTART:
C:/Users/Asus/AppData/Local/Programs/Python/Python
37/font.py ====
pygame 2.0.1 (SDL 2.0.14, Python 3.7.9)
Hello from the pygame community.
https://www.pygame.org/contribute.html
>>>
```

Pygame модулінде шрифт түрін көрсетпей (by default), экранға мәтін шығаруға болады. Ол үшін файл атауының орнына конструкторға None объектісін көрсету қажет. Мысалы,

```
pygame.font.Font(None, 36)
```

Python тілінде Font және SysFont кластарының қайсысынан болмасын Font типті объект құрылады.

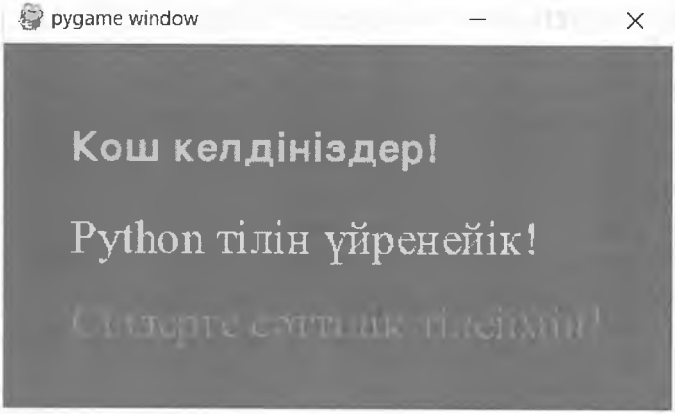
Font экземплярының **render()** әдісі арқылы көрсетілген мәтін жазылатын (Surface экземплярының) бет құрылады, оның аргументі ретінде шығарылатын мәтін, мәтіннің түсі, қажеттілігіне қарай фонның түсі көрсетіледі.

1-мысал. Экранға әртүрлі түспен, өлшемін тағайындай отырып, үш жолға мәтін шығаратын программа құрайық

```
import pygame
import sys
pygame.font.init()

sc = pygame.display.set_mode((400, 300))
sc.fill((0, 0, 225))

f1 = pygame.font.Font(None, 36)
```

```
text1 = f1.render('Қош келдіңіздер!', True,
                  (255, 255, 62))
f2 = pygame.font.SysFont('serif', 28)
text2 = f2.render("Python тілін үйренейік!", False,
                  (0, 255, 255))
f3 = pygame.font.SysFont('serif', 28)
text3 = f2.render("Сіздерге сәттілік тілеймін!", False,
                  (255, 0, 255))
sc.blit(text1, (40, 50))
sc.blit(text2, (40, 100))
sc.blit(text3, (40, 150))
pygame.display.update()

while 1:
   for i in pygame.event.get():
      if i.type == pygame.QUIT:
         sys.exit()
```

Программаның орындалу нәтижесі 14.16-суретте келтірілген.



Сурет 14.16. Графикалық режимде экранға мәтін шығару

Surface экземплярының get_rect() әдісі Rect типті объектіні береді, оның өлшемі беттің өлшеміне сәйкес

келеді. get_rect() әдісі әдісі аргументсіз пайдаланылғанда, оның сол жақ жоғарғы бұрышының координатасына (0,0) нүктесі алынады.

Біздің мысалда **get_rect()** әдісіне құрылатын тік төртбұрыш ауданына **center** қасиетін мән ретінде береміз. Бұл қасиет ретінде **Rect** экземплярының координаталар центрін анықтайды. Бұдан тағайындалған координаталар центрі мен беттің өлшеміне сай, қалған координаталар автоматты түрде есептеледі.

Ал, blit() әдісі шақырылғанда, екінші аргумент ретінде құрылған **Rect** экземпляры беріледі және бұдан сол жақ жоғарғы бұрышының координатасы есептеледі. Бағыттаушы клавишаларды (оңға және солға) басқан кезде тік төртбұрыштың X осінің координатасы өзгереді. Нәтижесінде blit() әдісі жаңа координаталар бойынша бетті сызады.

2-мысал. Бағыттаушы клавишалар арқылы басқарылатын мәтінді экранға шығаратын программа құрайық.

```
import pygame as pg
import sys
pg.init()
sc = pg.display.set_mode((650, 350))
sc.fill((200, 255, 200))
font = pg.font.SysFont('serif', 63)
text = font.render(    "Сәлем баршаңызға!!!", True, (0, 0,
225))
place = text.get_rect(
   center=(200, 150))
sc.blit(text, place)
 pg.display.update()
 while 1:
   for i in pg.event.get():
      if i.type == pg.QUIT:
         sys.exit()
    pressed = pg.key.get_pressed()
   if pressed[pg.K_LEFT]:
      place.x -= 1
   elif pressed[pg.K_RIGHT]:
```

```
        place.x += 1
    sc.fill((200, 255, 200))
    sc.blit(text, place)
    pg.display.update()
    pg.time.delay(20)
```

Программаның орындалу нәтижесі 14.17-суретте келтірілген.

Сурет 14.17. Бағыттаушы клавишалар арқылы басқарылатын мәтін

14.6. Pygame модуліне суреттерді жүктеу және сақтау

Image модулі – сурет жүктеу және сақтау функциялары бар нысан. Кескін Surface нысаны ретінде жүктеледі. Surface класы оларды басқаруға мүмкіндік береді (сызықтар салу, пиксельдерді орнату, аймақтарды түсіру және т.б.).

Pygame.image модулінің load () функциясы суретті жүктейді және сол суретті көрсететін Surface көшірмесін жасайды. load() ішіне файлдың аты енгізіледі. Негізгі форматы BMP болып табылады, бірақ егер pygame.image.get_extended () ақиқат мәнге ие болса, онда бірқатар басқа форматтарды жүктеуге болады: PNG, GIF, JPG және т.б.

12-мысал. Кез келген графикалық форматтағы суретті бейнелейтін программа құрайық.

```
import pygame
pygame.init()
screen=pygame.display.set_mode((315,240))
img_surf=pygame.image.load('butter.jpg')
img_rect=img_surf.get_rect(center=(153,125))
screen.blit(img_surf, img_rect)
pygame.display.update()
while True:
   for i in pygame.event.get():
      if i.type==pygame.QUIT:
         exit()
pygame.time.delay(20)
```

Программаның орындалу нәтижесі 14.18-суретте келтірілген

Сурет 14.18. Кез келген графикалық форматтағы суреттің графикалық экранға шығарылуы

Кесте 14.3.

PyGame модульдері

Модуль атауы	Қызметі
pygame.cdrom	CD-ROM жетектерін баскару және оған кіру
pygame.cursors	Курсор бейнелерін жүктеу
pygame.display	Экранға кіру
pygame.draw	Фигураларды, сызықтарды және нүктелерді сызу
pygame.event	Сырткы окиғаларды баскару
pygame.font	Жүйелік каріптерді пайдалану
pygame.image	Кескінді жүктеу және сактау
pygame.key	Пернетактада басылған пернені оқу
pygame.mixer	Дыбыстарды жүктеу және ойнату
pygame.mouse	Тышқанмен басқару
pygame.movie	Кинофильм файлдарын ойнату
pygame.music	Музыка және аудиолармен жұмыс
pygame.rect	Төртбұрышты аймактарды басқару
pygame.sndarray	Дыбыс деректерін басқару
pygame.surface	Суреттер мен экранды басқару
pygame.transform	Суреттерді өзгерту және жылжыту
pygame.time	Уақыт пен кадр жиілігін басқару

Бақылау сұрақтары мен жаттығулар

1. **PyGame** кітапханасының қызметі қандай?
2. **PyGame** кітапханасы қалай орнатылады?
3. **PyGame** кітапханасының жұмысы неден басталады?
4. Бейне шығарылатын графикалық экранның өлшемі қалай тағайындалады?
5. pygame.quit () және exit () функцияларының қызметі қандай?

6. Қосымша беттер құру қалай жүзеге асырылады?
7. **blit()** әдісі қандай қызмет атқарады?
8. Негізгі беттің үстіне қосымша бетті қалай орналастыруға болады?
9. pg.display.update() қандай қызмет атқарады?
10. Белгілі бір түспен ерекшеленген фон қалай құрылады?
11. **PyGame** кітапханасының графикамен жұмыс істеуге арналған функцияларын пайдалана отырып, үйдің суретін бейнелейтін программа құрыңдар (негізгі бөлігі мен терезелері - тік төртбұрыш, шатыры – үшбұрыш және т.б.).
12. **pygame.display** модуліндегі **set_caption()** функциясын пайдаланып, терезенің тақырыбы өзгеріп отыратын программа құрыңдар.
13. Горизонталь бағытта өзінің түсін өзгертіп отыратын доптың қозғалысын бейнелейтін программа құрыңдар.
14. Өшетін электромагниттік тербелісті бейнелейтін программа құрыңдар.
15. Экранда өзгермелі қогалыс жылдамдыгы бар дененің қозғалысын бейнелейтін программа құрыдар.
16. Екі дененің хаосты қозғалысын бейнелейтін программа құрыңдар.
17. Экранда жапалақтап жауған қарды бейнелейтін, тышқанның батырмалары арқылы басқарылатын программа құрыңдар.
18. Бағыттаушы клавишалар арқылы басқарылатын, жүріп бара жатқан автомобильді бейнелейтін программа құрыңдар.
19. Ұшып бара жатқан самолеттің, суда жүзіп бара жатқан қайықтың, шауып бара жатқан жылқыны бейнелейтін программа құрыңдар.
20. Ұшып бара жатқан зымыранды бейнелейтін программа құрыңдар.
21. Экранға шахмат тақтасын бейнелейтін программа құрыңдар.

ПРОГРАММАЛАР КІТАПХАНАСЫ

1. Қарапайым есептеулер. Сызықтық алгоритмдерді программалау мысалдары

1. Негізі үшбұрыш болатын пирамиданың көлемін табу. Пирамида көлемі: $V = \frac{sh}{3}$,

$$S = \sqrt{p(p-a)(p-b)(p-c)}; \quad p = \frac{a+b+c}{2}$$

```
from math import *
a = int(input())
b = int(input())
c = int(input())
h = int(input())
p = (a+b+c)/2
s = sqrt(p*(p-a)*(p-b)*(p-c))
v = s*h/3
print('Пирамида көлемі V= ',v)
```

2. Конустың бүйір беті мен көлемін табу.
$S = \pi RL; \quad V = 1/3 * \pi r^2 h$

```
print('Косинустың бүйір беті мен көлемін табу')
r = int(input('радиюсын енгіз: '))
l = int(input('ұзындығын енгіз: '))
h = int(input('биіктігін енгіз: '))
v = (3.142*h*pow(r,2))/3
s = 3.142*l*r
print('косинус параметірлері: ')
print('R=',r/3,'L=',l,'H=',h)
print('V=',v/8/3,'  S=',s/8/3)
```

2. Тармақталған алгоритмдерді программалау мысалдары

3. ax > b теңсіздігін шешу.

```
print('ax > b теңсіздігін шешу')
a = int(input())
b = int(input())

if a!=0:
   print('x>',b/a)
elif a==0:
   print('x<',b/a)
elif b<0:
   print('x-кез келген сан')
else:
   print('шешім жоқ')
```

4. $\frac{a \ln |x|}{\sqrt{a-x^2}}$ егер |x|<1
егер |x|>1

```
from math import*
print('Program ES')
a = int(input('a: '))
x = int(input('b: '))

if abs(x)<1:
    y = a*ln*(abs(x))
elif abs(x)>1:
    y = sqrt(a-pow(x,2))
print('y=',y)
```

5. $ax^2+x+c=0$ теңдеуін шешу

```
from math import *
print('ax*x+bx+c=0')
a = int(input('a: '))
b = int(input('b: '))
c = int(input('c: '))
```

```
if a==0 and b==0:
    print('шешімі жоқ')
elif a==0:
    print('түбірі: ',-c/b)
elif c==0:
    print('x1=',0,'x2=',-b/a)
else:
    d = pow(b,2)-4*a*c
    if d>0:
            x1 = (-b+pow(d,2))/(2*a)
            x2 = (-b-pow(d,2))/(2*a)
            print('түбірлері: ',x1,' ',x2)
    else:
            print('нақты түбірлері жоқ')
```

3. Қайталану алгоритмдерін программалау мысалдары

6. Бүтін сандарды экранға шығару:

а) 1-ден N-ге дейінгі;

ә) N-нен 1-ге дейінгі;

б) N0-ден N1 дейінгі сандарды экранға шығару.

```
# a)program NA
n = 10
for k in range (1,n):
    print(k)
# Progrm NA1
n = int(input())
for k in range (1,n):
    print(k)

# Program A2
n = int(input('n='))
k = n
while k>=1:
    print(k)
    k = k-1
# ә)Program NA2
n = 10
```

```
for k in range (n,1,-1):
    print(k)

# б)Program NA3
n0 = int(input('n0='))
n1 = int(input('n1='))

for n in range (n0,n1):
    print(n)
```

7. y=f(x) функциясының [a,b] аралығындағы мәндер таблицасын құру керек. Қадамды h деп алу қажет.

$x_0=a$, $x_1=x_1+h$, $x_2=x_1+h=x_0+h+h=x_0+2h\ldots$

$x_1=x_0+ih$, $y_1=f(x_1)$.

Нүкте саны: k:=[(b-a)/h]+1.

Айталық: $f(x)=(x^2+2*\sin x+1)/(\cos x+x-1)$, h=0.2, a=-4, b=4 болсын.

```
from math import *
a = int(input())
b = int(input())
h = int(input())
n = floor((b-a)/h)+1
x = a
for k in range (1, n, 1):
    y = (x*x+2*sin(x)+1)/(cos(x)+x-1)
    print('x=',x,'y=',y)
    x = x+h
```

8. Екі санның ең үлкен ортақ бөлгішін табу программасы

```
from math import *
m = int(input('m: '))
n = int(input('n: '))

x = m
y = n
```

```
while x != y:
    if x>y:
            x = x-y
    else:
            y = y-x
print('EYOB= ',x)
```

9. x^k/k - мәні берілген а санынан кіші болатын К санын табу.

А)	Ә)
x = int(input('x: ')) a = int(input('a: ')) k = 1 p = x while p/k < a: k = k+1 p = x*p print('k=',k)	a = int(input('a: ')) x = itn(input('x: ')) k=0 p=1 do: k = k+1 p = x*p while p/k <= a: print('k= ',k)

10. Берілген М санына кіші ең үлкен Фибоначчи санын анықтау. Фибоначчи саны:

```
m = int(input('m: '))

u = 0
u1 = 1
u2 = 1

while u2 < m:
    u2 = u1+u
    u = u1
    u1 = u2
print('u=',u)
```

11. n! есептеу. n!=(n-1)!n - пайдаланамыз.

```
n = int(input('n: '))
f = 2
for k in range (3, n):
    f = f*k
print('f=',f)
```

12. Қатардың қосындысын E>0 дәлдігімен есептеу.

```
x = int(input('x= '))
e = int(input('e= '))
n = 0
y = 1
s = 1
while y > e:
    n = n+1
    y = y*x/n
    s = s+y
print('s=',s)
```

13) Алғашқы 100 жай сандарды табыңыз.

```
from math import *
print('Jai san: ')
print('1,')
k = 0
s = 0
i = 1

while k <= 100:
    i = i+1
    for j in range (1, i-1):
            if i%j == 0:
                    s = s+1
            if s == 1:
                    k = k+1
                    print(i,',')
    s=0
print('...')
```

14. N жолдан тұратын мәтіндегі латынның дауысты үлкен әріптерін шығару.

```
n = int(input('n: '))
for i in range (1, n):
    while not Elon:
            l = input('l: ')
            if l in ['A','E','I','O','U']:
                    print(l)
    println()
```

15. yi+1=1/2(yi+x/y) жуықтау формуласын пайдаланып $y = \sqrt{x}$ мәнін E>0 дәлдігімен анықтау.

Нұсқау: а) y0 - үшін x-ке кез келген сан алуға болады, немесе 1-ді алуға да болады.

```
x = int(input('x: '))
e = int(input('e: '))

if x > 0:
    y0 = 1
    y = (y0+x/y0)/2
    while abs(y-y0) > e:
            y0 = y
            y = (y0+x/y0)/2
    print(y,' ',abs(y-y0))
else:
    print('теріс саннан түбір табылмайды')
```

16) X5 + 9X4 + 1.7X2-9.6 көпмүшелігінің мәнін есептеңіз. Мұндағы X = 0,1, … , 5.

```
from math import *
x = real
y = [1,2,3,4,5]
for i in range (1,5):
    y[i]=exp(5*ln(i))-9*exp(e*Ln(i))+1.7*pow(i,2)-9.6
    print('y[i]= ', y[i]/3/1)
```

17) n натурал саны, a, b нақты сандары берілген (a<>b).

r0, r1, … , rn тізбегін алыңыз. Мұндағы ri = a+ih, h=(b-a)/n.

```
from crt import *
from math import *

r = array[0.20]
a = int(input('a: '))
b = int(input('b: '))
n = int(input('n: '))

h = (b-a)/n
for i in range (0,n):
    r[i] = a+(i*h)
for i in range (0,n):
    print(r[i]:5:2,' ')
```

18) a, b бүтін сандары берілген (a<>0, b<>0). Олардың барлық ортақ бөлгіштерін табыңдар.

```
from math import *

k = 300

a = int(input('a: '))
b = int(input('b: '))
print('iiiiii')

for i in range (1, k):
    if a % i == 0:
            print('i=', i,'')
print('jjjjjj')

for j in range (1, k):
    if b % j == 0:
            print('j=', j,'')
```

19. Қарпайым калькулятордың моделін бейнелейтін программа құрыңдар.

```
print("Ноль орындалатын операция белгісі"
    "\n программаның орындалуын тоқтатады")
while True:
  s = input("операция белгісі (+,-,*,/): ")
  if s == '0':
    break
  if s in ('+', '-', '*', '/'):
    x = float(input("x="))
    y = float(input("y="))
    if s == '+':
      print("%.2f" % (x+y))
    elif s == '-':
      print("%.2f" % (x-y))
    elif s == '*':
      print("%.2f" % (x*y))
    elif s == '/':
      if y != 0:
        print("%.2f" % (x/y))
      else:
        print("нольге бөлуге болмайды!")
  else:
    print("операция белгісі қате!")
```

4. Массивтер

20. Әрқайсысы 50 элементтен тұратын A,B,C массивтері берілген. Олардың элементтерінің орта мәндерінен D массивін құру.

```
#Massiv 19 esep
n = 5
A = []
B = []
C = []
D = list(range(n))
print("A[] = ")
A = [ int(input()) for i in range(n)]
print("B[] = ")
B = [ int(input()) for i in range(n)]
```

```
print("C[] = ")
C = [ int(input()) for i in range(n)]

for i in range(len(A)):
   D[i] = (A[i] + B[i] + C[i]) / 3
print(D)
```

21. Сызықты А массиві берілген. К-элементтен N-элементке (K<N) дейінгі элементтердің кіші мәндісін және оның индексін табу.

```
#Massiv 20 esep
n = 10
k = 5
a = []

print("a[] = ")
a = [ int(input()) for i in range(n)]

min = a[k]
I = k
for i in range(k, n):
   if a[i] < min:
      min = a[i]
      I = i
print(min)
print(I)
```

22. Сызықты Х массиві берілген. Х-тің элементтерінің оң және теріс мәндерінен жеке екі массив құру.

```
#Massiv 21 esep
n = 10
a = []
b = []
c = []
j = 0
k = 0
```

```
a = [ int(input('Введите число A[%d] = '%(i+1))) for i in range(n)]

for i in a:
    if i <= 0:
        b.append(i)
    else:
        c.append(i)
print(b)
print(c)
```

23. n натурал саны, a_1, … , a_n нақты сандар тізбегі берілген. a_1, … , a_n тізбегінің кемуі бойынша реттелгенін анықтаңыз.

```
#Massiv 22 esep
n = int(input("n = "))
a = [ int(input('Введите число A[%d] = '%(i+1))) for i in range(n)]
a = sorted(a, reverse = True)

print(a)
```

24. a_1, … , a_n бүтін сандар тізбегі берілген (тізбектің қайталанатын мүшелері болуы мүмкін). Тізбекте ең болмағанда қатар тұрған сәйкес екі мүшесінің болатынын анықтаңыз.

```
#Massiv 23 esep
n = int(input("n = "))
a = [ int(input('Введите число A[%d] = '%(i))) for i in range(n)]
for i in range(n):
    if a[i] == a[i + 1]:
        print(i, a[i])
        print(i + 1, a[i + 1])
```

25. P(x)=a0xn+a1xn-1+…+ an-1x+an көпмүшелігінің мәнін есептеу.

Есептеуге қолайлы болу үшін Горнер схемасын пайдаланамыз:

P(x)=(…(a0x+a1+)x+a2)x+…+an-I)x+an

Бұл схеманы пайдаланғанда алгоритм: y:=y*x+A[i] түріне келеді.

```
#Massiv 24 esep
n = int(input("n = "))
x = int(input("x = "))
a = [ int(input('Введите число A[%d] = '%(i))) for i in
range(n + 1)]
y = a[n]
for i in range(1, n + 1):
    y *= x
    y += a[n-i]
print(y)
```

26. Сызықты $b_1,b_2,\ldots,b_n$ массиві берілген. Оның ең үлкен элементін, орналасқан орнын табу.

Алгоритм:

$$y_{max}=\begin{cases} y_i, \text{ егер } y_i > y_{max}, \\ y_{max}, \text{ егер } y_i < y_{max} \end{cases}$$

```
#Massiv 25 esep
n = int(input("n = "))
a = [ int(input('Введите число A[%d] = '%(i))) for i in
range(n)]

max = a[0]
I = 0

for i in range(n):
    if a[i] > max:
        max = a[i]
        I = i
print(max, I)
```

27. B(nxm) (n<=10, m<=20) массивінің оң элементтерінің қосындысын табу.

```
#Massiv 27 esep

n = int(input("n = "))
m = int(input("m = "))
s = 0

a = []
for i in range(n):
    b = []
    for i in range(m):
        b.append(int(input()))
    a.append(b)

for i in range(n):
    for j in range(m):
        if a[i][j] > 0:
            s += a[i][j]

print(s)
```

28. Әр тік жолдың (21-есеп) оң элементтерінің қосындысынан құралған сызықтық массивті табу.

```
#Massiv 28 esep

n = int(input("n = "))
m = int(input("m = "))

a = []
for i in range(n):
    b = []
    for i in range(m):
        b.append(int(input()))
    a.append(b)
```

```
for i in range(n):
    for j in range(m):
        print(a[i][j], end = ' ')
    print()

c = []
for i in range(m):
    s = 0
    for j in range(n):
        s += a[j][i]
    c.append(s)

print(c)
```

29. $a_1, a_2, \ldots, a_n$ массив элементтерін өсу реті бойынша реттеп жазу.

```
#Massiv 29 esep
n = int(input("n = "))
a = [ int(input('Введите число A[%d] = '%(i+1))) for i in range(n)]
a = sorted(a)

print(a)
```

30. $y_i = \sum_{k=1}^{n} x_i / k!$ - анықталатын массив құру,

мұндағы $x_i = x_0 + ih$, $x \in [x_0, x_m]$.

```
#massiv 30 esep
m = int(input("m = "))
x0 = int(input("x0 = "))
xm = int(input("xm = "))
h = int(input("h = "))
n = int(input("n = "))

x = x0
```

```
m1 = int((xm - x0) / h) + 1

a = []
for i in range(1, m1):
    p = 1
    s = 0

    for k in range(1, n):
        p *= k
        s += x / p

    a.append(s)
    x = x + h
   # print(a)

print(a)
```

31. A(mxn), B(nxl) матрицаларының көбейтіндісін табу.

Алгоритм: - $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{ki} (i = 1, m; j = 1, i)$, яғни c(m,l) - табу.

```
#Massiv 31 esep

m = int(input("m = "))
n = int(input("n = "))
l = int(input("l = "))

a = []
for i in range(m):
    z = []
    for i in range(n):
        z.append(int(input()))
    a.append(z)

b = []
```

```
for i in range(n):
    x = []
    for i in range(l):
        x.append(int(input()))
    b.append(x)

c = []
for i in range(m):
    v = []
    for j in range(l):
        s = 1
        for k in range(n):
            s *= a[i][k] * b[k][j]
        v.append(s)
    c.append(v)

for i in range(m):
    for j in range(l):
        print(c[i][j], end = ' ')
    print()
```

32. Екі қабырғасы бірлермен қоршалған сандық үшбұрыш Паскаль үшбұрышы деп аталады, егер үшбұрыштың ішінде орналасқан екі санның қосындысы сол сан тұрған жолдың астындағы жақын тұрған санға тең болса:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
.....................
```

n натурал саны берілген. Паскаль үшбұрышының алғашқы n жолын табатын программа құрыңдар.

```
#Massiv 32 esep
```

```
n = int(input("n = "))

a = []
for i in range(n):
    a.append([])
    a[i].append(1)
    for j in range(1, i):
        a[i].append(a[i - 1][j - 1] + a[i - 1][j])
    if(n != 0):
        a[i].append(1)


for i in range(n):
    for j in range(i + 1):
        print(a[i][j], end = ' ')
    print()
```

33. N цифрдан тұратын натурал сан Армстронг саны деп аталады, егер N дәрежеге алынған оның цифрларының қосындысы сол санның өзіне тең болса. (Мысалы, 153 = 13 + 53 + 33). Екі, үш. төрт цифрлардан тұратын барлық Армстронг санын табатын программа құрыңдар.

```
# massiv 33 esep

max = int(input("max = "))
s = 0
k = 0

for i in range(max):
    temp = i
    while temp != 0:
        temp = int(temp / 10)
        k += 1
    temp = i
    while temp:
        s += pow(int(temp % 10), k)
        temp = int(temp / 10)
    if s == i:
```

```
        print(s)
    s = 0
    k = 0
```

34. ε нақты саны берілген (ε>0). A_1, A_2, … A_n төмендегі заңдылықпен құрылған:

$A_n = n/(sqrt(n^2 +1) – Sqrt(n^2 - 1)$.

$|A_n – A_{n-1}| < ε$ шартын қанағаттандыратын алғашқы A_n (n>=0) мүшені табыңыз.

```
# massiv 34 esep
from cmath import *

n = int(input("n = "))
e = int(input("e = "))

a = []
for i in range(1, n + 1):
    a.append(i / (sqrt(i * i + 1) - sqrt(i * i - 1)))

for i in range(len(a) - 1):
    if abs(a[i] + a[i + 1]) < e:
        print (a[i], a[i + 1])
```

35. B(nxm) (n<=10, m<=20) массивінің оң элементтерінің қосындысын табу.

```
#Massiv 35 esep
n = int(input("n = "))
m = int(input("m = "))
a = []
for i in range(n):
    b = []
    for i in range(m):
        b.append(int(input()))
    a.append(b)

max1 = a[0][0]
```

```
for i in range(n):
    for j in range(m):
        if i < j:
            a[i][j] = 0
        elif i > j:
            a[i][j] = 2
        else:
            a[i][j] = 1

for i in range(n):
    for j in range(n):
        print(a[i][j], end=' ')
    print()
```

36. N - фамилияны алфавит бойынша жазу

```
#string 35 esep

n = int(input("n = "))
a = []

a = [i for i in input(" Familyalardi engiz = ").split()]
b = sorted(a)
#b = sorted(a, reverse=True)
print(b)
```

37. Т жолының Т1 жарты жолын Т2 жолымен ауыстыру.

```
t=input("Алғашқы тексты енгиз: ")
t1=input("Ауысатын тексты енгиз: ")
t2=input("Ауыстыратын тексты енгиз: ")
print(t.replace(t1, t2))
```

5. Функциялар

38. $z = \frac{tha + th^2(b-a)}{\sqrt{th(a^2-b^2)}}$ есептеуге $thx = \frac{e^{2x}-1}{e^{2x}+1}$ формуласын пайдаланып анықтау.

```
#function 38 esep
from math import *
a = int(input(" a = "))
b = int(input(" b = "))

def TH(x):
    c = exp(x * 2)
    r = (c - 1) / (c + 1)
    return r

t1 = TH(a)
c = a - b
t2 = TH(c)
d = pow(a, 2) - pow(b, 2)
t3 = TH(d)
z = (t1 - pow(t2, 2)) / pow(t3, 2)

print(z)
```

39. $R = \sqrt{x^2 + y^2}$ және A=arctg(y/x) полярлық координаталарын есептеу (x>0, y - декарттық координаталар)

```
#function 39 esep
from math import *
n = int(input("n = "))

def pol():
    r = (x * x + y * y)
```

```
    a = atan(y / x)
    print("r = ", r, "a = ", a)

for i in range(n):
    x = int(input("(x>0) x = "))
    y = int(input("y = "))
    pol()
```

40. $d = \prod_{i=1}^{n} R_i, \qquad s = \sum_{i=1}^{n} A_i$ есептеу, мұндағы Ri, Ai - поляр координаталары, xi=i/(1+i2); yi=1/i арқылы есептеледі.

```
# 40 esep
from math import *

n = int(input("n = "))

r = 0
a = 0
d = 1
s = 0

def pol():
    global r, a
    r = sqrt(x * x + y * y)
    a = atan(y / x)

for i in range(1, n + 1):
    x = i / (i * i +1)
    y = 1 / i
    pol()
    d = d * r
    s = s + a

print("d = ", d, "s = ", s)
```

41. A, B, C нақты сандары берілген. Табыңыз:
(max(a,a+b)+max(a6 b+c))/(1+max(a+bc,1,15)

```
#procedure 41 esep

a = int(input("a = "))
b = int(input("b = "))
c = int(input("c = "))

def maximum(x, y):
   if(x > y):
      max = x
   else:
      max = y
   return max

s = maximum(a, a + b) + maximum(a, b + c) +
maximum(a + b * c, 1.15)
print("s = ", s)
```

42. A, B нақты сандары берілген. Табыңыз:
U=min(a,b), V=min(ab,a+b), min(u+v^2, 3014).

```
def min(a, b):
   if a>b:
      return b
   return a
a=float(input())
b=float(input())
u=min(a, b)
v=min(a*b, a+b)
print('U =',u)
print('V =',v)
print('MIN(U+V^2, 3014) =', min(u+v*v, 3014))
```

43. $s = \sum_{i=1}^{n} f^2(i)$ - есептеу,

$$f(i) = \begin{cases} \frac{1}{i^2}, & \text{егер i тақ болса} \\ \frac{1}{i^2+1}, & \text{егер i жұп болса} \end{cases}$$

```
#function 43 esep

n = int(input("n = "))

def f(i):
    if i % 2 == 0:
        f = 1 / (i*i)
    else:
        f = 1 / (i * i + 1)
    return f

def S(n):
    s = 0
    for i in range(1, n + 1):
        s += f(i)
    return s

s = S(n)
print("s = ", s)
```

44. $C^m{}_n$=n!/m!(n!-m!) - есептеу.

```
#function 44 esep

n = int(input("n = "))
m = int(input("m = "))

def fact(n):
    f = 1
    for i in range(1, n + 1):
```

```
        f *= i
    return f

c = fact(n) / (fact(m) * (fact(n) - fact(m)))
print("c = ", c)
```

45. Бүтін сандардан құрылған массив элементтері берілген. Көрсетілген қадам санына сәйкес цтклдық ығысу жасайтын программаны функцияны пайдалана отырып құрыңдар.

```
def shift(lst, steps):
    if steps < 0:
        steps = abs(steps)
        for i in range(steps):
            lst.append(lst.pop(0))
    else:
        for i in range(steps):
            lst.insert(0, lst.pop())

nums = [8, 5, 6, 7, 3, 9, 0]
print(nums)

shift(nums, -2)
print(nums)

shift(nums, 3)
print(nums)
```

6. Графикалық операторлар

46. Түзу кесіндісінен және тең қабырғалы үшбұрыштан құралған бағыттаушы кесіндіні (12-сурет.) былай бейнелеу қажет: а) координатасы (100,100) болатын нүктеден координатасы (150,100) болатын нүктеге горизонталь бағытталған кесінді; ә) координатасы (100,50) болатын нүктеден координатасы (100,150) болатын нүктеге вертикаль бағытталған кесінді; б) координатасы (100,100) болатын нүктеден координатасы (100,50)

болатын нүктеге вертикаль бағытталған кесіндіні бейнелейтін программа құру.

```
#grafic 46 esep
import turtle
k = turtle.Turtle()
k.forward(100)
```

47. Экранда биллиард шарының қозғалысын бейнелеу.

```
#graphic 47 esep

import turtle, random

window = turtle.Screen()

border = turtle.Turtle()
border.speed(0)
border.up()
border.hideturtle()
border.pensize(5)
border.goto(200, 200)
border.down()
border.color('green')
border.goto(200, -200)
border.goto(-200, -200)
border.goto(-200, 200)
border.goto(200, 200)


ball = turtle.Turtle()
ball.hideturtle()
ball.shape('circle')
ball.color('red')
ball.up()

randx = random.randint(-190, 190)
```

```
randy = random.randint(-190, 190)
ball.showturtle()
ball.goto(randx, randy)

dx = 1
dy = 1

while True:
    x,y = ball.position()

    if x + dx >= 200 or x + dx <= -200:
        dx = -dx
    if y + dy >= 200 or y + dy <= -200:
        dy = -dy

    ball.goto(x + dx, y + dy)

window.mainloop()
```

48. Экранда жанып тұратын екі геометриялық фигураны (квадрат және шар) салу қажет. Фигуралардың түсі белгілі бір уақыт интервалында өзгеріп отыруы қажет.

```
#graphic 48 esep

import turtle, random, time

def choiseColor():
    red = random.random()
    green = random.random()
    blue = random.random()
    return red, green, blue

while True:
    ball = turtle.Pen()
    ball.hideturtle()
    ball.color(choiseColor())
```

```
ball.begin_fill()
ball.circle(100)
ball.end_fill()

square = turtle.Pen()
square.hideturtle()
square.up()
square.left(180)
square.forward(150)
square.color(choiseColor())
square.begin_fill()
square.forward(150)
square.right(90)
square.forward(150)
square.right(90)
square.forward(150)
square.right(90)
square.forward(150)
square.right(90)
square.end_fill()
time.sleep(1)
```

49. Сағаттың секундтық және минуттық стрелкасының қозғалысын бейнелейтін программа құру қажет. Басқа сөзбен айтсақ, екі стрелканың бір мезгілде айналуын бейнелеу, стрелканы кесіндімен алмастыруға болады.

```
from turtle import Screen, Turtle, Vec2D
from time import localtime

CENTER = Vec2D(0, 0)

def draw_line(position, heading, length, color):
    hands.up()
    hands.goto(position)
    hands.down()
    hands.setheading(heading)
```

```
        hands.color(color)
        hands.forward(length)

    def tick():
        time = localtime()

        second_heading = time.tm_sec * 6
        minute_heading = time.tm_min * 6 + second_heading /
60
        hour_heading = time.tm_hour % 12 * 30 +
minute_heading / 12

        hands.clear()

        draw_line(CENTER, second_heading, 300, 'red')
        draw_line(CENTER, minute_heading, 200, 'blue')
        draw_line(CENTER, hour_heading, 100, 'green')

        screen.update()
        screen.ontimer(tick, 1000)

    screen = Screen()
    screen.mode("logo")    # 0 degrees at top, clockwise
angles!
    screen.tracer(False)  # force manual screen updates

    # What this turtle draws is "permanent"
    dial = Turtle(visible=False)
    dial.penup()
    dial.dot()
    dial.setx(330)  # remember mode is "logo"
    dial.pendown()
    dial.circle(330)

    # What this turtle draws has to be redrawn on every tick
    hands = Turtle(visible=False)

    tick()
```

```
screen.mainloop()
```

50. Бірінің ішіне бірі орналасқан 10 квадратты бейнелейтін программа құрыңдар. Квадараттар қызыл және көк түспен кезектесіп боялатын болсын.

```
from turtle import Screen, Turtle, Vec2D
from time import localtime

CENTER = Vec2D(0, 0)

def draw_line(position, heading, length, color):
    hands.up()
    hands.goto(position)
    hands.down()
    hands.setheading(heading)
    hands.color(color)
    hands.forward(length)

def tick():
    time = localtime()

    second_heading = time.tm_sec * 6
    minute_heading = time.tm_min * 6 + second_heading /
60
    hour_heading = time.tm_hour % 12 * 30 +
minute_heading / 12

    hands.clear()
    draw_line(CENTER, second_heading, 300, 'red')
    draw_line(CENTER, minute_heading, 200, 'blue')
    draw_line(CENTER, hour_heading, 100, 'green')
    screen.update()
    screen.ontimer(tick, 1000)
screen = Screen()
screen.mode("logo")  # 0 degrees at top, clockwise
angles!
screen.tracer(False)  # force manual screen updates
```

```
    # What this turtle draws is "permanent"
    dial = Turtle(visible=False)
    dial.penup()
    dial.dot()
    dial.setx(330)  # remember mode is "logo"
    dial.pendown()
    dial.circle(330)

    # What this turtle draws has to be redrawn on every tick
    hands = Turtle(visible=False)

    tick()

    screen.mainloop()
```

51. Түстердің ауысуын дыбыспен ұштастыратын программа құру қажет.

```
    import turtle
    import winsound

    def choiseColor(k):
        if k % 2 == 0:
            return 'blue'
        else:
            return 'red'

    def sq(a, k):
        sqa.color(choiseColor(k))
        for i in range(4):
            sqa.forward(a)
            sqa.left(90)
        winsound.PlaySound('sound2.wav',
winsound.SND_ASYNC)

    sqa = turtle.Turtle()
    sqa.speed(10)
```

```
d = 40
k = 0
for i in range(10):
    sq(d, k)
    d += 20
    k += 1
```

52. Күн планетасын эллипс бойымен айналып тұрған планеталарды бейнелейтін программа құрыңдар.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from astropy.time import Time
from astroquery.jplhorizons import Horizons

sim_start_date = "2018-01-01"     # simulating a solar system starting from this date
sim_duration = 2 * 365      # (int) simulation duration in days
m_earth = 5.9722e24 / 1.98847e30  # Mass of Earth relative to mass of the sun
m_moon = 7.3477e22 / 1.98847e30

class Object:                 # define the objects: the Sun, Earth, Mercury, etc
    def __init__(self, name, rad, color, r, v):
        self.name = name
        self.r    = np.array(r, dtype=np.float)
        self.v    = np.array(v, dtype=np.float)
        self.xs = []
        self.ys = []
        self.plot = ax.scatter(r[0], r[1], color=color, s=rad**2, edgecolors=None, zorder=10)
        self.line, = ax.plot([], [], color=color, linewidth=1.4)

class SolarSystem:
    def __init__(self, thesun):
```

```
            self.thesun = thesun
            self.planets = []
            self.time = None
            self.timestamp = ax.text(.03, .94, 'Date: ', color='w',
transform=ax.transAxes, fontsize='x-large')
        def add_planet(self, planet):
            self.planets.append(planet)
        def evolve(self):          # evolve the trajectories
            dt = 1.0
            self.time += dt
            plots = []
            lines = []
            for p in self.planets:
                p.r += p.v * dt
                acc = -2.959e-4 * p.r / np.sum(p.r**2)**(3./2)  #
in units of AU/day^2
                p.v += acc * dt
                p.xs.append(p.r[0])
                p.ys.append(p.r[1])
                p.plot.set_offsets(p.r[:2])
                p.line.set_xdata(p.xs)
                p.line.set_ydata(p.ys)
                plots.append(p.plot)
                lines.append(p.line)
            self.timestamp.set_text('Date: ' + Time(self.time,
format='jd', out_subfmt='date').iso)
            return plots + lines + [self.timestamp]

    plt.style.use('dark_background')
    fig = plt.figure(figsize=[6, 6])
    ax = plt.axes([0., 0., 1., 1.], xlim=(-1.8, 1.8), ylim=(-1.8,
1.8))
    ax.set_aspect('equal')
    ax.axis('off')
    ss = SolarSystem(Object("Sun", 28, 'red', [0, 0, 0], [0, 0,
0]))
    ss.time = Time(sim_start_date).jd
    colors = ['gray', 'orange', 'blue', 'chocolate']
```

```
    sizes = [0.38, 0.95, 1., 0.53]
    names = ['Mercury', 'Venus', 'Earth', 'Mars']
    texty = [.47, .73, 1, 1.5]
    for i, nasaid in enumerate([1, 2, 3, 4]):  # The 1st, 2nd,
3rd, 4th planet in solar system
        obj = Horizons(id=nasaid, location="@sun",
epochs=ss.time, id_type='id').vectors()
        ss.add_planet(Object(nasaid, 20 * sizes[i], colors[i],
                    [np.double(obj[xi]) for xi in ['x', 'y', 'z']],
                    [np.double(obj[vxi]) for vxi in ['vx', 'vy',
'vz']]))
        ax.text(0, - (texty[i] + 0.1), names[i], color=colors[i],
zorder=1000, ha='center', fontsize='large')
    def animate(i):
        return ss.evolve()
    ani = animation.FuncAnimation(fig, animate,
repeat=False, frames=sim_duration, blit=True, interval=20,)
    plt.show()
```

53. Таңдау арқылы сұрыптау алгоритмін пайдалана отырып, массив элементтерін сұрыптайтын программа құрыңдар.

```
    def sel_sort(array):
        for i in range(len(array) - 1):
            m = i
            j = i + 1
            while j < len(array):
                if array[j] < array[m]:
                    m = j
                j = j + 1
            array[i], array[m] = array[m], array[i]
    n = int(input("n = "))
    a = [ int(input('енгізіңіз A[%d] = '%(i+1))) for i in
range(n)]

    print(a)
    sel_sort(a)
    print(a)
```

54. Москва сақиналы автомобиль жолының ұзындығы 109 км. Байкер осы жолмен жүруді 0-ші километрден сағатына **v** жылдамдықпен бастайды. Ол **t** сағаттан кейін жолдың қандай бөлігінде тоқтайды.

Енгізілетін мәліметтер

Программаға жылдамдық пен уақыттың мәндері енгізіледі. Егер v>0 болса, байкер сақиналы жолда оң бағытта, ал v<0 болса, теріс бағытта болады.

Шығарылатын мәліметтер

Программа 0-ден 108-ге дейінгі бүтін санды шығаруы тиіс, бұл байкердің тоқтайтын белгіленген номері

Мысалдар

Енгізілетін мәліметтер	**Шығарылатын мәліметтер**
60	11
20	

Енгізілетін мәліметтер	**Шығарылатын мәліметтер**
-1	108
1	

Программа коды:

```
v=int(input())
t=int(input())
print((109+(v*t%109))%109)
```

55. Екі натурал сан (n және m) берілсін. Егер олардың біреуі екіншісіне бүтіндей бөлінетін болса, экранға 1, кері жағдайда кез келген бүтін саны шығаратын программа құрыңдар.

Есепті шығаруда және қайталану құрылымдарын пайдалануға болмайды.

Енгізілетін мәліметтер
n және m натурал сандары енгізіледі.

Шығарылатын мәліметтер
Есептің шартына сәйкес жауабын шығарыңдар
Мысалдар

Енгізілетін мәліметтер	Шығарылатын мәліметтер
8	1
2	

Енгізілетін мәліметтер	Шығарылатын мәліметтер
2	1
8	

Программа коды:
```
n=int(input())
m=int(input())
a1=n%m
a2=m%n
print(a1*a2+1)
```

56. Жеке жолдарға жазылған үш натурал сан (a,b,c) берілген. Қабырғалары осындай мәнге тең болатын үшбұрыш болатынын анықтаңдар.

Шығарылатын мәліметтер

Егер үшбұрыш болатын болса, YES жолын, кері жағдайда NO жолы шығарылуы қажет.

Мысалдар

Енгізілетін мәліметтер	Шығарылатын мәліметтер
3	YES
4	
5	

Енгізілетін мәліметтер	Шығарылатын мәліметтер
3	NO
8	
17	

Программа коды:

```
a=int(input())
b=int(input())
c=int(input())
if a+b>c and a+c>b and b+c>a:
   print("YES")
else:
   print("NO")
```

57. $ax + b = 0$ теңдеуін шешіңдер. Мұндағы *a және b* бүтін сандар.

Енгізілетін мәліметтер:
a және b бүтін сандары енгізіледі.

Шығарылатын мәліметтер:
Егер теңдеудің шешімі сандық мәндер болса, барлық шешімін шығару керек; егер шешімі болмаса, "NO" сөзін шығару керек (тырнақшасыз) және егер шексіз көп шешімі болса, "INF" сөзі (тырнақшасыз) шығарылуы тиіс.

Мысалдар

Енгізілетін мәліметтер	Шығарылатын мәліметтер
6	NO
-2	

Енгізілетін мәліметтер	Шығарылатын мәліметтер
1	1
1	

Программа коды:

```
a=int(input())
b=int(input())
if a==0 and b==0:
   print("INF")
elif a==0 or b%a!=0:
   print("NO")
else:
   print(-b//a)
```

58. Берілген сандардың ішіндегі нольдерді, оң сандар мен теріс сандарды есептеңдер.

Енгізілетін мәліметтер

Алдымен N саны енгізіледі, одан кейін соған тең сандар тізбегі енгізіледі.

Шығарылатын мәліметтер

Алдымен нольдер саны, оң сандардың және теріс сандардың саны (мөлшерін) шығарылуы тиіс.

Енгізілетін мәліметтер	**Шығарылатын мәліметтер**
5	4 1 0
28	
0	
0	
0	
0	

Программа коды

```
n=int(input())
k=0
m=0
p=0
```

```
for i in range(n):
    a=int(input())
    if a==0:
        k=k+1
    elif a>0:
        m=m+1
    else:
        p=p+1
print(k, m, p)
```

59. Бүтін сандардан құрылған массив берілген. Массивте таңбалары бірдей көрші тұрған элементтер жұбының бар екенін анықтайтын программа құрыңдар.

Енгізілетін мәліметтер

Алдымен массивтегі элементтер саны N енгізіледі (1<=N<=100000). Одан кейін N массив элементтері енгізіледі. Массив элементтері бүтін сандардан құрылған.

Шығарылатын мәліметтер

Таңбалары бірдей көрші элементтер жұбы болатын болса, YES сөзін кері жағдайда NO сөзін шығару қажет.

Мысалдар:

Енгізілетін мәліметтер	**Шығарылатын мәліметтер**
5	NO
1 -1 2 -3 4	

Енгізілетін мәліметтер	**Шығарылатын мәліметтер**
5	YES
1 -1 -5 2 0	

Программа коды

```
n=int(input())
```

```
s=input()
a=s.split()
k=0
for i in range(len(a)):
   a[i] = int(a[i])
for i in range(n-1):
   if (a[i]>0 and a[i+1]>0) or (a[i]<0 and a[i+1]<0):
      k=k+1
if k!=0:
   print("YES")
else:
   print("NO")
```

60. Бүтін сандардан құрылған массив берілген. Массив элементтері өсу реті бойынша сұрыпталмаған (әрбір келесі элемент алдыңғысынан кіші емес). Массивтегі әртүрлі элементтер санын анықтайтын программа құрыңдар.

Енгізілетін мәліметтер

Алдымен массивтегі элементтер саны N енгізіледі (1<=N<=100000). Одан кейін бос орындар арқылы N массив элементтері енгізіледі. Массив элементтері -2^{31}-нен $2^{31} - 1$ аралығындағы бүтін сандардан құрылған.

Шығарылатын мәліметтер

Тек бір ғана сан шығады, ол массивтегі әртүрлі элементтер саны.

Мысалдар:

Енгізілетін мәліметтер	**Шығарылатын мәліметтер**
5	2
2 1 1 1 1	

Енгізілетін мәліметтер	Шығарылатын мәліметтер
5	1
1 1 1 1 1	

Программа коды:

```
n=int(input())
k=1
a=input().split()
for i in range(len(a)):
   a[i] = int(a[i])
for i in range(n-1):
   if a[i]!=a[i+1]:
      k=k+1
print(k)
```

ЕСЕПТЕР

1. Жазықтықтағы екі нүктенің координатасы берілген. Сол нүктелердің центрі координаталар жүйесінің бас нүктесінде орналасқан шеңбердің бойында жататынын анықтаңдар.

2. n және m бүтін санды айнымалыларының мәндері 3-ке еселік болатынын анықтаңдап. Егер екі санның мәні де 3-ке еселік болса, қосындысын, кері жағдайда айырмасын есептеңдер.

3. n (n<100) натурал саны берілген. N санында қанша цифр бар?

4. n (n<9999) натурал саны берілген. Санның төрт цифрының да әр түрлі болатынын анықтаңдар.

5. n (n<9999) натурал саны берілген. Санның төрт цифрын қосып есептегендегі полиндром болатынын дәлелдеңдер. Мысалы, 2222, 6116, 0440 және т.с.с.

6. n (n<9999) натурал саны берілген. Берілген санның үш цифры бірдей сан бола ма?

7. n натурал саны берілсін. N саны цифрларының қосындысының кубына тең бола ма?

8. n натурал саны берілген. Есептеңдер:

1*2+2*3*4+…+n(n+1)…2n.

9. n натурал, x нақты саны берілген. Есептеңдер: x ***/2^n

10. n натурал саны берілген. Есептеңдер:

$$\frac{\cos 1}{\sin 1}\cdot\frac{\cos 1+\cos 2}{\sin 1+\sin 2}\cdot\ldots\cdot\frac{\cos 1+\ldots+\cos n}{\sin 1+\ldots+\sin n}$$

11. Бүтін k және n (n>=k>=0) сандары берілген. Есептеңдер: (n(n-1)…(n-k-1))/k!

12. n натурал саны берілген. Осы санның барлық қарапайым бөлгіштерін табыңдар.

13. $\sum_{i=1}^{\infty}(a_i-b)^2$ қосындысын есептеңдер. Мұндағы

a_i $\begin{cases} 1, \text{егер і тақ болса} \\ 1/2, \text{кері жағдайда} \end{cases}$

a_i $\begin{cases} 1, \text{егер і тақ болса} \\ 1/3, \text{кері жағдайда} \end{cases}$

14. Өлшемі NxM нақты матрицаның ең үлкен және ең кіші элементін тауып, оларды жолдың 1-ші және соңғы элементтерінің орнына орналастырыңдар.

15. Өлшемдері NxM және MxL нақты A және B матрицалары берілген реті NxL болатын C=A+B матрицаны табыңдар. C матрицасының элементін мына формауламен есептеңдер:

$$c_{ij} = \sum_{j=1}^{m} a_{ij} b_{jk}$$

16. Өлшемі 25x4 болатын бүтін санды A матрицасы группадағы 25 студенттің 4 емтиханнан тұратын сессиясының нәтижесі жайлы мәліметті құрайды. Барлық бағасы «5» болатын студенттердің саны мен фамилиясын табыңдар.

17. Өлшемі NxM болатын A матрицасы студенттердің оқу орындары арасындағы спартакиадасындағы N факультеттің M спорт түрі бойынша алған орындары жайлы мәліметті құрайды. Факультеттерді жалпы командалық сынақта олардың алған орындарының реті бойынша орналастырыңдар.

18. Өлшемі NxN болатын символдық квадрат матриңа берілген. Бас диагональдан жоғары орналасқан барлық элементтері «a» әрібімен, бас диагональдан төмен жатқан элементтері «0» әрібімен алмастырыңдар.

19. N натурал саны, символдары берілген және олардың арасындағы қос нүкте бар. Бірінші қос нүктемен екінші қос нүктенің арасында орналасқан барлық

символдарды алыңдар. Егер екінші қос нүкте болмаса бірінші қос нүктеден кейінгі символдарды алыңдар.

20. N натурал саны және S1,…,Sn символдар берілген. Қатар келген бос орындардың ең үлкен санын есептеңдер.

21. N натурал саны және S1,…,Sn символдары берілген S1,…,Sn тізбегіне кіретін A,B,C - әріптер санын анықтаңдар.

22. N натурал саны және S1,…,Sn символдары тізбегі берілген тізбектегі child сөзін children сөзімен алмастырыңдар.

23. S1,…,Sn символдар тобы берілген. S1=S20, S2=S19, S10=S11 теңсіздіктердің қасындағы жалған теңдіктер санын табыңдар.

24. N натурал саны берілген. 0-ден N-ге дейінгі бүтін сандарды үштік санау жүйесінде шығарыңдар.

25. Берілген текст тек цифрлар мен әріптерден тұрады. Текст 6-ға еселі болатын ондық сандардың жазылуы болатынын анықтаңдар.

26. A және B массивтерінің оң элементтерінің қосындысы мен санын есептеңдер.

27. Есептеңдер $Z=(x1+y1)/(x2-y2)$, мұндағы x1, x2 – $2x^2+x-4=0$ теңдеуінің түбірі, ал y1, y2 – $ay^2+2y-1=0$ теңдеуінің түбірі.

28. A(3,5) және B(4,7) бүтін санды матрицаларының 3-ке еселі элементтерін баспаға шығарыңдар.

29. n, m және x(n), y(m) массивтері берілген. Есептеңдер:

$$z = \left(\sum_{i=1}^{n} \sin x_1 + \sum_{i=1}^{m} \cos y_1 \right) / \sum_{i=1}^{n} / x_1 / \ 30.$$

$$z(x) = \int_0^x P_n(t)dt - 3{,}4 \cdot \int_0^x Q_m(t)dt$$ функциясының таблицалық мәндерін [0,1] аралығында 0,2 қадаммен есептеңдер. Мұндағы Pn(t) және Qm(t) сәйкес n, m - дәрежелі көпмүшеліктер.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР

1. Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения: Издательство Питер, 2020 г. //3-издание. - 512 с.

2. Мак Грат М. Программирование на Python для начинающих: [перевод с англ. М.А.Райтмана]. – Москва: Эксмо, 2015 г. – 192 с.

3. Стивенсон Б. Python. Сборник упражнений. - Издательство: ДМК Пресс, 2021 г. – 238 с.

4. 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.

5. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для вузов / Д. Ю. Федоров. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2021. — 210 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/478098

6. Буйначев С. К. Основы программирования на языке Python : учебное пособие / С. К. Буйначев, Н. Ю. Боклаг ; [науч. ред. Ю. В. Песин]. – Екатеринбург : Изд-во Урал. ун-та, 2014. – 91 с.

7. Лутц М. Изучаем Python : пер. с англ. / М. Лутц. – СПб. : Символ-Плюс, 2009.

8. Мусин Д. Самоучитель Python, Выпуск – 0,2. – май, 2017. – 154 с.

9. Халықова К.З. Паскаль тілінде программалау: Оқу құралы. – Алматы: Абай атындағы Алматы мемлекеттік университеті, 2002. – 244 б.

10. Густокашин М.С. Основы программирования на Python. https://www.coursera.org/learn/python-osnovy-programmirovaniya/home/week/1...9 (Дата обращения 11.12.2020)

11. Шапошникова . Python. Введение в программированию. Июнь, 2021 https://younglinux.info/pygame, https://younglinux.info/python (Дата обращения 01.03.2021)

12. Информатика. Язык Python. https://foxford.ru/wiki/informatika/rekursiya-v-python (Дата обращения 01.07.2021)

13. Pythonist. https://pythonist.ru/opredelenie-chetnosti-chisla-s-ispolzovaniem-rekursii/ (Дата обращения 01.08.2021)

14. Сборник задач. Работа со строками. https://smartiqa.ru/python-workbook/str#6 (Дата обращения 09.08.2021)

15. Лабораторная работа. Функции и процедуры в Python.

https://sites.google.com/site/moisitkovskij/home/5-kurs/l-r-5/l-r-7-spiski-kak-odnomernye-massivy (Дата обращения 11.07.2021)

16. Питонтютор. Работа с массивами. https://pythontutor.ru/lessons/2d_arrays/ (Дата обращения 01.06.2021)

17. Python 3. Множества в Python. https://python-scripts.com/sets#create-sets (Дата обращения 12.08.2021)

18. Bestprog. Python. Теория. Файлы. https://www.bestprog.net/ru/2020/04/22/python-files-general-concepts-opening-closing-a-file-functions-open-close-ru/ (Дата обращения 19.07.2021)

19. PythonRU. Файлы в Python. https://pythonru.com/osnovy/fajly-v-python-vvod-vyvod (Дата обращения 21.07.2021)

Қосымша 1

Python программалау тілі бойынша тест сұрақтары

1. Айнымалылардың қандай типтері бар (бірнеше нұсқаны таңдаңыз):

a. float
b. str
c. num
d. int
e. bool
f. for

2. int типі қандай шаманы сипаттау үшін пайдаланылады:

a. логикалық
b. ондық бөлшек сандар
c. бүтін сан
d. нақты сандар

3. float типі қандай шаманы сипаттау үшін пайдаланылады:

a. логикалық
b. ондық бөлшек сандар
c. бүтін сан
d. нақты сандар

4. str типі қандай шаманы сипаттау үшін пайдаланылады:

a. нақты шаманы
b. символдар жолы
c. логикалық
d. бүтін сан

5. Python тілінде айнымалы атауы ретінде ... алуға болмайды :

a. орыс әріптерін

b. латын әріптерін
c. +,=, !, ?
d. сандар
e. астыңғы сызықша
f. скобкалар

6. print (‘45+6’) өрнегінің орындалу нәтижесі қандай болады?
a. 51
b. 45+6
c. 6
d. 45

7. Төмендегілердің қайсысын операндалар ретінде орындауға болады?
a. Өрнектер
b. Айнымалылар
c. Операциялар
d. Әдебиеттер

8. Дұрыс алынған айнымалы атауын көрсетіңіз:
a. AXby
b. 2RB
c. Абылай
d. StuDent
e. A+B
f. _ABBA
g. [QuQu]

9. name = "John"
print('Hi, %s' % name)
операторлар орындалғанда экранда қандай нәтиже көрінеді?

a. “Hi”
b. “Hi, name”
c. қате
d.”Hi, John”

10. Төмендегі математикалық өрнектің Python тілінде жазылуын көрсетіңіз:

$$\frac{x^5}{y}$$

a. Sqr(x)/y
b. x*5/y
c. Pow(x,5)/y
d. (x)^5/y
e. X**5:y
f. X**5/y

11. name = "snow storm"
print("%s" % name[6:8])
a. st
b. sto
c. to
d. syntax error

12. $a^{e^x} + sinx - \sqrt{a}$ орнегі Python тілінде қалай жазылады?

a. a,ex+sin(x)-pow(a)
b. pow(a,exp(x))+sinx+sqrt(a)
c. pow(a,exp(x))+sin(x)-sqrt(a)
d. a**exp(x)+sin(x)-sqrt(a)
e. a**exp(x)+sin(x)-pow(a)
f. a**(x)+sin(x)-pow(a)

13. Нәтиженің бір жолға жазылуын көрсететін оператор:

a. print('30','20','10',sep='+', end=' ')
b. print('30','20','10',map='+', sep=' ')
c. print('30','20','10',sep='+', list=' ')
d. print('30','20','10',sep='+', format=' ')

14. Программаға модульді қалай қосуға болады?

a. import math
b. import math()

c. import (math)
d. import.math

15. Программа орындалғаннан кейін қандай нәтиже шығады?

```
a=20
b=a+4
a=b*100
print(a)
```

a. 1600
b. 20
c. 16
d. 100

16. 17//4 өрнегінің орындалу нәтижесі қандай ?
a. 1
b. 2
c. 3
d. 4

17.
```
A=2
B=2.4*A
x=(A+B)/A*B-A
print('x=', x)
```
Берілген программа бойынша x-тің мәнін анықтаңыз.
a. 4,8
b. 3,75
c. 15,75
d. 9,52

18.
```
A=4
B=2
x=A/B MOD B
print('x=',x)
```
берілген программа бойынша x-тің мәнін анықтаңыз.
a. 1
b. 0

c. 2
d. 4

19. Программа орындалғаннан кейін консольде қандай жауап көрсетеді?

```
a=5
b='3'
print(a+b)
```

a. 53
b. 8
c. TypeError(ошибка)
d. 35

20. Программаның орындалу нәтижесі қандай?

```
x = 4.5
y = 2
print(x // y)
```

a. 2
b. 2.25
c. 9
d. 20.25

21. print(type(1 / 2))
a. class 'int'
b. class 'number'
c. class 'float'
d. class 'double'

22. from math import* - нені білдіреді?
a. Математикалық кітапхана
b. Математикалық айнымалы
c. Айнымалыны енгізу
d. Экранға шығару операторы

23. Жаңа жолға көшіретін символ
a. \v
b. \t
c. \r

d. \n

24. Берілгендерді енгізуге арналған функция:
a. Inpu t()
b. print()
c. while
d. elif

25. 10 div 3 = өрнегінің нәтижесі қандай?
a. 5
b. 2
c. 3
d. 1

26. Программа коды орындалғаннан кейін экранға не шығады?

```
a=5
b=10//2
print(a==b)
```

a. 2
b. Ештеңе көрсетпейді
c. True
d. False

27. Тармақталуы бар процестерді ұйымдастыру үшін _____ оператор пайдаланылады.
a. Сызықтық
b. Шартты
c. Циклдық
d. Шартсыз көшу

28. С айнымалысының мәні неге тең?

```
a=2
b=5
if a>b:
c=a
```

```
        else:
            c=b
```

a. 5
b. 2
c. 7
d. 3

29. Программа орындалғаннан кейін a айнымалысының неге тең болады?

```
a=2
if a<3:
   print(a+1)
else:
   print(a-1)
```

a. 4
b. 3
c. 2
d. 1

30. Программа орындалғанда алынатын нәтиже қандай?

```
x = True
y = False
z = False
if not x or y:
   print(1)
elif not x or not y and z:
   print(2)
elif not x or y or not y and x:
   print(3)
else:
print(4)
```

a. 4
b. 3
c. 2
d. 1

31. x=int(input('x='))
if x>0:
y=x+2
else:
y=2*pow(x,3)
print(y)

x-ке -4 мәнін берсе, программаның жауабы қандай болады?

a. 16
b. 8
c. -16
d. 10

32. $y = \begin{cases} x + 5, & \text{егер } 16 \le x \le 26 \\ 2x + 20, & \text{басқа жағдайлар үшін} \end{cases}$

Өрнектің python тіліндегі программасының дұрыс нұсқасын көрсетіңіз.

a. x=5
if x<=16 and x>=26:
y=x+5
else:
y=2x+20
print("y=", y)

b. x=5
if x>=0:
y=x+5
else:
y=2*x+20
print("y=")

c. x=5
if x>=16, x<=26:
y=x+5
else
y=2*x+20
print("y=", y)

d. x=5
if x>=16 and x<=26:

```
    y=x+5
else:
    y=2*x+20
print("y=", y)
```

33. Программа орындалғандағы нәтиже қандай болады?

```
x=71
y=115
if x>y:
    print("max =" ,x)
else:
    print("max =",y)
```

a. max=115
b. max=71
c. max=186
d. max=71, max=115

34. Берілген программада name='Aiman' мәні берілген жағдайда, қандай жауап көрсетеді?

```
name=str(input(''))
    if name=='Askar':
            print('Hello', name,'!')
    else:
            print('I don't know you!')
```

a. Hello
b. Hello, Askar!
c. I don't know you!
d. I don't know you

35. for i in range(1,5):
цикл қанша рет қайталанады?

a. 3
b. 4
c. 5
d. 6

36. for i in range (10,21,3):
оператордың қадам саны қанша?

a. 3
b. 21
c. 2
d. 10

37. Бұл программа үзіндісі орындалғаннан кейін «а» бүтін айнымалысының мәні неге тең болады?

```
a = 10
for i in range(2):
  a -= 1
  print(a)
```

a. 9 8
b. 10 9 8 7 6 5 4 3 2
c. 8
d. 9

38. Программадағы тармақталуды көрсету үшін ... пайдаланылады.

a. If
b. For
c. While
d. Def

39. Программа орындалғаннан кейін с айнымалысының мәні неге тең болады?

```
x=3
c=0
while x>1:
  c=c+1
  x=x-1
```

a. 1
b. 2
c. 3
d. 4

40. Цикл дегеніміз –

a. Бірнеше рет орындалатын реттілік.
b. Бір рет орындалатын тізбек
c. Бастапқы кодтың соңына баратын айнымалы
d. Бірнеше рет орындалатын функция

41. Программада циклды ұйымдастыру үшін ... пайдаланылады.

a. If
b. For
c. While
d. Def
e. Continue

42. 1-ден 100-ге дейінгі жұп сандарды кері ретпен шығаратын программаны көрсетіңіз:

a.
```
i = 100
while i >= 1:
    print(i)
      i -= 2
```
b.
```
i = 1
while i <= 100:
    print(i)
      i -= 2
```
c.
```
for i in range(100,1,-2):
  print(i)
```
d.
```
for i in range(100,1,2):
  print(i)
```
e.
```
for i in range(1,100,2):
  print(i)
```

43. «Алдымен шарт тексеріліп, егер шарт орындалса, онда циклдың денесі орындалады» - бұл қандай оператор?

a. If
b. For
c. While
d. Def

44. Программаның орындалу нәтижесі қандай?

```
i=1
while i<=10:
  print(i**2)
    i+=1
```

a. 1 4 9 16 25 36 49 64 81 100
b. 1 3 5 7 9 11 …100
c. 10-ға дейінгі сандардың квадраттарын көрсетеді
d. 2-нің 1-ден 10-ға дейінгі дәрежелерін көрсетеді
e. 2 4 8 16 32 64 128 ...

45. def a(b, c, d): pass – операторы нені сипаттайды?

a. тізімді анықтайды және инициализация жасайды.
b. Ештеңе жасамайтын функцияны анықтайды.
c. Параметрлерді беретін функцияны анықтайды.
d. Бос классты анықтайды.

46. Төмендегі программа орындалғанда қандай жауап алынады?

```
A={1,2,3}
B={3,2,3,1}
print(A==B)
```

a. 1,2,3
b. 3,2,1
c. True
d. False

47. Төмендегі программа орындалғанда экранға ... басылып шығады.

```
s = {7, 3, 11, 2, 5}
for num in s:
      print(num)
```

a. 7 3 11 2 5
b. 28

c. 5 2 11 3 7
d. 2 3 5 7 11

48. A.extend(L) – операторы қандай қызмет атқарады?
a. A тізімінің соңына барлық L тізімін қосу
b. Тізімнің соңына A элементін қосу
c. L элементінің тізімге кіру саны
d. Тізімді сұрыптау (тізімді өзгертеді, ештеңе қайтармайды)

49. Төмендегі операторлар орындалганда алынатын нәтиже қандай?
A='Python'
print(s[::2])

a. Py
b. th
c. Pto
d. On

50. Төмендегі операторлар орындалғанда алынатын нәтиже қандай?
A='Python'
print(s[::-2])

a. Py
b. thon
c. on
d. nhy

51. A және B жиынындары берілген. A.union(B) операторының қызметі қандай?
a. B жиынтығынан барлық элементтерді a жиынтығына қосады
b. A және B жиындарының жиынтығы болып табылатын жиынды қайтарады
c. A және B жиындарының қиылысы болып табылатын жиынды қайтарады

d. A және B жиындарының айырмасын қайтарады (A-ға кіретін, бірақ в-ға кірмейтін элементтер))

52. Тізімдегі элементтердің оң индексі нешеден басталады?

a. 1
b. -1
c. 0
d. 2

53. Тізімдегі элементтер қандай таңбамен бөлінеді?

a. Үтірмен
b. Қос нүктемен
c. Тырнақшамен
d. Нүктелі үтірмен

54. Тізімнің соңына бір сәтте тек қана бір элемент қосатын функцияны көрсетіңіз?

a. Insert()
b. Extend()
c. Append()
d. Дұрыс жауап жоқ

55. List = [10, 20, 30, 40]
print(List.pop())
программасының орындалу нәтижесі қандай болады?

a. 10
b. 20
c. 30
d. 40

56. a = [1,2,3,None,(),[],]
print(len(a))
программасының орындалу нәтижесі қандай болады?

a. 4
b. 5

c. 6
d. 7

57. Төмендегі программа орындалғанда қандай жауап алынады?

```
A = ['red', 'green', 'blue']
print('***'.join(A))
```

a. red***green***blue
b. redgreenblue
c. redgreenblue***
d. ***redgreenblue

58. ASCII код бойынша 65-тен 70-ке дейінгі сандардың сәйкес символдық таңбасын шығаратын программаның дұрыс нұсқаларын көрсетіңіз

a.
```
S1=65
Sn=70
print('Берілген сандардың сәйкес символдары:')
for i in range(S1,Sn):
    S=ord(i)
    print(S,end=' ')
```

b.
```
print('Берілген сандардың сәйкес символдары:')
for i in range(65,70):
S=chr(i)
  print(S,end=' ')
```

c.
```
S1=65
Sn=70
print('Берілген сандардың сәйкес символдары:')
for i in range(S1,Sn):
    S=chr(i)
    print(S,end=' ')
```

d.
```
print('Берілген сандардың сәйкес символдары:')
for i in range(65,70):
S=str(i)
  print(S,end=' ')
```

e.
```
S1=65
Sn=70
```

```
print('Берілген сандардың сәйкес символдары:')
for i in range(S1,Sn):
    print(S,end=' ')
```

59. A-дан Z-ке дейінгі символдардың сандық кодын көрсететін программаның дұрыс нұсқасын анықтаңыз:

a.
```
print('Келесі A-дан Z-ке дейінгі символдардың коды: ')
for i in range(ord('1'),ord('27')):
    print(i, end=' ')
```

b.
```
print('Келесі A-дан Z-ке дейінгі символдардың коды: ')
for i in range(ord('A'),ord('Z')):
    print(i, end=' ')
```

c.
```
print('Келесі A-дан Z-ке дейінгі символдардың коды: ')
for i in range(A, Z):
    print(i, end=' ')
```

d.
```
print('Берілген сандардың сәйкес символдары:')
for i in range(str(A),str(Z)):
        S=ord(i)
        print(S,end=' ')
```

60. Массив дегеніміз не?

A) Программада реттелген мәлімет
B) операцияларды қолдайды
C) Бір типті көп элементтен тұрады
D) Бірнеше типті элементтен тұрады

61. Массив элементтерінің типтері ...

a. әртүрлі типті болуы мүмкін
b. тек қана бүтін типті болу керек
c. тек қана нақты типті болу керек
d. теріс мән болмау керек

62. Практикада мәліметтерді тікбұрышты кестелер түрінде сақтауға тура келеді. Мұндай кестелер _____ деп аталады.

a. Матрицалар
b. функциялар
c. 2, 3 немесе n өлшемді массивтер
d. Тізім
e. жиым

63. import os
os.remove("demofile.txt")
операторларының қызметі қандай?

a. файлды ашады
b. файлды жояды
c. файлды оқиды
d. файлға жазады

64. Берілген string айнымалысындағы барлық кіші әріптерді үлкен әріптерге ауыстыратын функция?
a. Title()
b. Upper()
c. Strip()
d. Find()

65. Сөйлемде санның бар немесе жоқ екенін тексеретін функция?
a. Lower()
b. Replace()
c. Isdigit()
d. Upper()

66. Массивті төмендегідей толтыру үшін
X = [1 3 5 7 9 11]
цикл денесіне қандай оператор қойылады?
X = [0]*6
for k in range(6):

...
a. X[k] = 2*k
b. X[k] = 2*k - 1

c. X[k] = 2*k + 1
d. X[k] = 2*(k + 1)

67. Массивті төмендегідей толтыру үшін
X = [12 9 6 3 0 -3]
цикл денесіне қандай оператор қойылады?

```
    X = [0]*6
for k in range(6):
...
```

a. X[k] = 12 - 2*k
b. X[k] = 3*k - 12
c. X[k] = 3*(k + 1) + 9
d. X[k] = 12 - 3*k

68. N элементтен тұратын X массиві берілген. S айнымалысы бойынша массивтің жұп элементтерінің санын табу үшін массив орнына қандай шарт қойылады?

a. S // 2 == 1
b. S % 2 == 0
c. X[j] % 2 == 1
d. X[j] % 2 == 0

69. Кездейсоқ жағдайларды дер кезінде тоқтатып, программа жұмысын қалыпты жағдайда аяқтайтын оператор

a. If-else
b. try-except
c. For
d. While
e. Def

70. Екі санды қосу үшін функция дұрыс пайдаланылған программаны табыңыз:

a.
```
def add_two_nums(a, b):
    c = a + b
    return c
        print(add_two_nums(3, 5))
```

b. c = a + b
return c
print(add_two_nums(3, 5))
c. def add_two_nums(a, b):
c = a + b
return c
add_two_nums(3, 5)
d. add_two_nums(3, 5)

71. Циклдің ішіне кіретін операторлардың орындалуын ғана тоқтатып, қайталану шартын тексеруге жіберетін оператор
a. break
b. continue
c. If-else
d. try-except

72. Рекурсия дегеніміз -
a. Функцияның өзін-өзі шақыруы
b. Параметрлі функция
c. Тек формальды параметрді функцияда пайдалану
d. Функцияда сандық параметрдің пайдаланылуы

73. Функция параметр мәнін өзгерте алады:
a. параметр - айнымалы,
b. параметр мән бойынша беріледі,
c. параметр сілтеме бойынша функцияға беріледі,
d. өзгермелі түрдегі параметр.

74. Тұрақты шамаларды сақтауға арналған объект ... деп аталады.
a. кортеж
b. массив
c. жиын
d. құрылым

75. Тізімдер тәрізді өзгеретін, реттелмеген элементтердің жиыны, оның элементтері «**түйінді сөз: мән**» түрінде сипатталады:

a. сөздік
b. жолдар
c. тізімдер
d. кортеж

76. Python тілінде параметрлер функцияға қалай беріледі:

a. мәні бойынша,
b. сілтеме,
c. арнайы хаттама бойынша
d. мекенжай кеңістігі бойынша.

77. Сөздіктегі соңғы элементті өшіруге арналған функция болады. Ол үшін Төмендегілердің қайсысы функцияны орындауға шақыру процесінің құрамына кірмейді:

a. del
b. popitem()
c. get()
d. clear.

78. Программалардағы функцияларды бөлу қай тәсілге сәйкес келеді:

a. объектіге бағытталған программалау,
b. құрылымдалған программалау,
c. прототиптеу әдісі,
d. экстремалды программалау.

79. Python модулі – бұл …

a. мәтіндік файл,
b. екілік файл,
c. Python abs () функциясы,
d. программаның модульділігін жоғарылату үшін қолданылатын арнайы алдын ала анықталған құрылым.

80. Python программалау тілінде функциялар _____ операторының көмегімен анықталады.

a. Array
b. Def
c. For
d. Print

үшін clear() функциясы пайдаланылады.

81. Сөздікті барлық элементтерден тазалап, босату үшін пайдаланылатын , функция ...

a. del
b. popitem()
c. get()
d. clear.

82. Break операторының қызметі қандай?
a. Программаны аяқтау
b. Циклдан шығу
c. Компьютерді бұзу
d. Программаны жою

83. Pygame кітапханасын іске қосу командасын белгілеңіз:

a. pygame.init()
b. import pygame
c. pygame.display.update()
d. pygame.display.set_caption(" ")

84. Pygame құралдарының инициализациясы:

a. pygame.init()
b. import pygame
c. pygame.display.update()
d. pygame.display.set_caption(" ")

85. Экранда циклге дейін объектілерді көрсету (жаңарту):

a. pygame.init()
b. import pygame
c. pygame.display.update()
d. pygame.display.set_caption(" ")

86. Ойын терезесіндегі тақырыптың тағайындалуы:
a. pygame.init()
b. import pygame
c. pygame.display.update()
d. pygame.display.set_caption(" ")

87. Ойын терезесін құру:
a. pygame.event.get
b. screen.fill()
c. pygame.display.flip()
d. pygame.display.set_mode((600,400))

88. Ойын терезесіндегі Артқы фонды түрлі түспен толтыру:
a. pygame.event.get
b. screen.fill()
c. pygame.display.flip()
d. pygame.display.set_mode((600,400))

89. Font экземплярының әдісі арқылы көрсетілген мәтін жазылатын (Surface экземплярының) бет құрылады?
a. pygame.event.get()
b. **render()**
c. pygame.display.flip()
d. pygame.display.set_mode((600,400))

90. Pygame.event.get модульдың тағайындалуы

a. жалғыз оқиғаны кезектен күтеді (ждет одиночного события из очереди)
b. оқиғаларды кезектен жояды (удаляет события из очереди)

c. оқиғаларды кезектен алады (получает события из очереди)
d. жаңа оқиғаны кезекке орналастырады (размещает)

91. Pygame.event.post модулінің тағайындалуы:
a. жалғыз оқиғаны кезектен күтеді
b. оқиғаларды кезектен жояды
c. оқиғаларды кезектен алады
d. жаңа оқиғаны кезекке орналастырады

92. Pygame.event.clear модульдың тағайындауы:
a. жалғыз оқиғаны кезектен күтеді
b. оқиғаларды кезектен жояды)
c. оқиғаларды кезектен алады
d. жаңа оқиғаны кезекке орналастырады

93. Pygame.event.wait модульдың тағайындауы:
a. жалғыз оқиғаны кезектен күтеді
b. оқиғаларды кезектен жояды
c. оқиғаларды кезектен алады
d. жаңа оқиғаны кезекке орналастырады

94. Оқиғаны басқару батырмасы - оңға, солға:
a. if keys [pygame.K_LEFT] x-=speed
b. if keys [pygame.K_UP] y-=speed
c. if keys [pygame.K_RIGHT] x+=speed
d. if keys [pygame.K_DOWN] y+=speed

95. Оқиғаны басқару батырмасы - жоғары, төмен:
a. if keys [pygame.K_LEFT] x-=speed
b. if keys [pygame.K_UP] y-=speed
c. if keys [pygame.K_RIGHT] x+=speed
d. if keys [pygame.K_DOWN] y+=speed
a. Суреттерді ойынға орналастыру командалары
b. img_surf=pygame.image.load ('111.bmp')
c. pygame.draw.rect(sc, GREEN, (200, 200, 250, 200))
d. pygame.draw.circle(sc, YELLOW, (100, 100), 50)
e. a_surf=pygame.image.load('222.bmp').convert()

96. Графикалық примитивтерді шығаратын командалар:

a. img_surf=pygame.image.load ('111.bmp')
b. pygame.draw.rect(sc, GREEN, (200, 200, 250, 200))
c. pygame.draw.circle(sc, YELLOW, (100, 100), 50)
d. a_surf=pygame.image.load('222.bmp').convert()

97. Суреттерді ұсынуға арналған PyGame нысаны

a. Surface
b. Event
c. Image
d. Picture

98. Төртбұрыш салу командасы

a. pygame.draw.circle
b. pygame.draw.rect
c. pygame.draw.lines
d. pygame.draw.rectangle

99. Пішіндерді салуға арналған PyGame нысаны

a. mouse
b. image
c. draw
d. caption

100. Жадыдағы орынды үнемдеу үшін тек бейненің болігін ғана жаңарту функциясы:

a. fill( )
b. update( )
c. delay( )
d. caption ( )

МАЗМҰНЫ