Алматы қаласының білім басқармасының
Almaty Polytechnic College КМҚК

PYTHON ТІЛІНДЕ БАҒДАРЛАМАЛАУ

оқу – әдістемелік құрал

Авторлар:
Омарбеков Медет Асетович колледж директоры
Бердибаева Салиха Диновна арнайы пәндер оқытушысы

Алматы, 2025

АННОТАЦИЯ

Python - ойындар, сенімді және жылдам веб - қосымшалар және тіпті ғылымда қолданылатын ең қарапайым, әрі кең таралған әмбебап және жоғары деңгейлі бағдарламалау тілі. Әлемдегі көптеген ірі цифрлық платформалар өз жобаларында Python тілін пайдаланады.
Мысалы: YouTube, Instagram, Spotify, Netflix.

Оның қарапайым синтаксисі мен оқуға ыңғайлы құрылымы Python - ды жаңа бастағандар үшін таптырмас құрал етеді. Python тілін үйренуді бастау үшін компьютер және интернет болса жеткілікті.

Әдістемелік құралда Python тілінің негізгі ұғымдары: айнымалы - лар мен деректер типтері, шартты және тернарлы операторлар, циклдер, функциялар, тізімдер, кортеждер, сөздіктер, жолдармен, файлмен жұмыс, ерекше жағдайлар, модульдер қарастырылады. Әрбір тақырып теориялық материалмен қатар, практикалық тапсырмалармен, тест тапсырмаларымен толықтырылған. Бұл - алған білімді бекітіп, бағдарламалаудағы дағдыларды тиімді дамытуға мүмкіндік береді.

Python тілі - болашақ IT маманы үшін сенімді іргетас. Бұл оқу құралы бағдарламалау бойынша алғашқы қадамдарын жасап жатқан білім алушыларға, мұғалімдерге, өз бетімен үйренушілерге де пайдалы болмақ.

Мазмұны

PYTHON БАҒДАРЛАМАЛАУ ТІЛІНЕ КІРІСПЕ

Python - әртүрлі типтегі қосымшаларды жасауға арналған танымал жоғары деңгейлі бағдарламалау тілі. Оларға веб - қосымшалар, ойындар, жұмыс үстелі бағдарламалары және мәліметтер қорымен жұмыс жатады. Python машиналық оқыту және жасанды интеллект зерттеулері саласында кеңінен таралған.

Python тілін алғаш рет 1991 жылы голландиялық әзірлеуші Гвидо Ван Россум жариялады. Содан бері бұл тіл дамудың ұзақ жолын өтті. 2000 жылы 2.0 нұсқасы, 2008 жылы 3.0 нұсқасы шықты. Нұсқалар арасындағы үлкен алшақтықтарға қарамастан, үнемі ішкі версиялар шығарылады. Осы материалды жазу кезінде Python - ның ағымдағы соңғы нұсқасы - Python 3.12.1 қолданылды.

Python бағдарламалау тілінің негізгі ерекшеліктері:

- Коды интерпретатор арқылы тікелей орындалады;
- Динамикалық түрлендіруге ие, яғни айнымалылардың түрін алдын ала анықтау қажет емес;
- Синтаксисі ағылшын тіліне ұқсас, сондықтан оны үйрену оңай;
- Көпфункционалдылық;
- Кросс-платформалық;
- Бағдарламалау парадигмаларының кең ауқымын, соның ішінде объектіге - бағытталған және функционалды парадигмаларды қолдайды.

Python - өте қарапайым бағдарламалау тілі, оның қысқа және сонымен бірге өте қарапайым және түсінікті синтаксисі бар. Сәйкесінше, оны үйрену жеңіл және шын мәнінде бұл оның оқып үйренуге арналған ең танымал бағдарламалау тілдерінің бірі болуының себептерінің бірі.

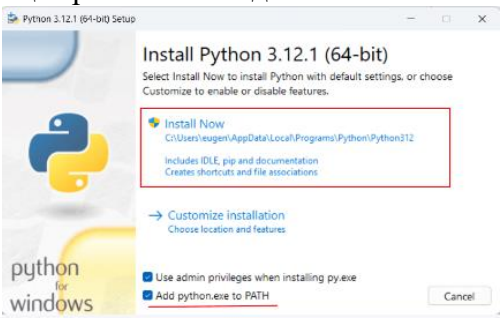
Python тек білім беру саласында ғана емес, белгілі бір бағдарламаларды, соның ішінде коммерциялық бағдарламаларды жазуда да танымал. Сондықтан біз пайдалана алатын осы тіл үшін көптеген кітапханалар жазылған.

1 – бөлім. WINDOWS ЖҮЙЕСІНДЕ ОРНАТУ ЖӘНЕ БІРІНШІ БАҒДАРЛАМА ҚҰРУ

Python тілінде бағдарламалар құру үшін бізге интерпретатор қажет. Оны орнату үшін ресми сайтында https://www.python.org/downloads/ бетіне өтіп, тілдің соңғы нұсқасын жүктеп алу сілтемесін табыңыз:

Батырманы басу арқылы ағымдағы ОЖ-ге сәйкес Python орнатушысы жүктеледі.

Windows операциялық жүйесінде орнатушыны іске қосқан кезде орнату шеберінің терезесі ашылады:



Мұнда интерпретатор орнатылатын жолды беруге болады. Оны әдепкі етіп қалдырайық, яғни C:\Users\[пайдаланушы аты]\AppData\Local\Programs\Python\Python312\.

Сонымен қатар, ең төменгі жағында интерпретаторға жолды орта айнымалыларына қосу үшін «Add Python 3.12 to PATH" құсбелгісін қойыңыз.
Осыдан кейін пәрмен жолында/терминалда python -- version пәрменін іске қосу арқылы Python және оның нұсқасын орнатуды тексере аламыз.

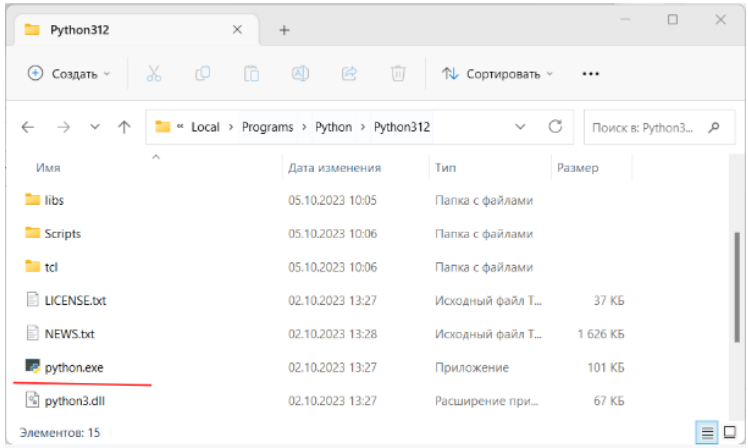
```
C:\Users\eugen>python --version
Python 3.12.1

C:\Users\eugen>
```

📖 1.1 Интерпретаторды іске қосу

Алдыңғы тақырыпта сипатталғандай интерпретаторды орнатқаннан кейін Python қосымшаларын құруды бастауға болады. Сонымен, бірінші қарапайым бағдарламаны жасайық.

Орнату кезінде мекенжай өзгертілмесе, Windows жүйесінде Python әдепкі бойынша C:\Users\[пайдаланушы аты]\AppData\Local\Programs\Python\Python[нұсқа_нөмірі]\ жолына орнатылады және python.exe деп аталатын файлды ұсынады.



🕮 1.2 PyCharm – ды орнату

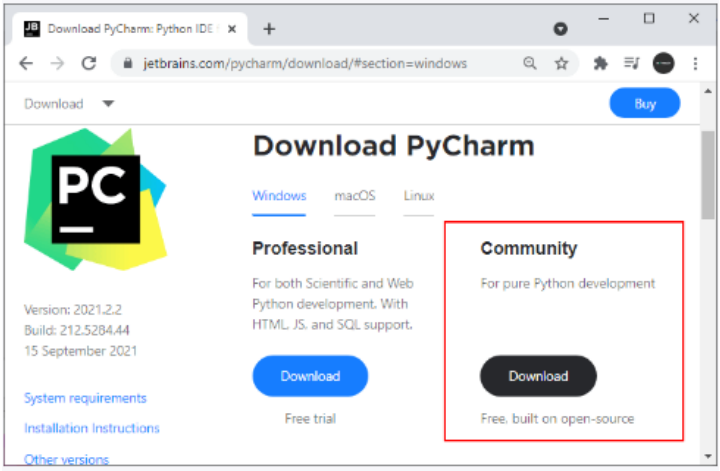
Бағдарламаларды құруда әртүрлі біріктірілген өңдеу орталары немесе IDE пайдаланылады.

IDE бізге кодты теру үшін мәтіндік редакторды ұсынады, бірақ стандартты мәтіндік редакторлардан айырмашылығы, IDE сонымен қатар толық синтаксистік бөлектеуді, автотолтыруды немесе кодтың интеллектуалды нұсқауын, жасалған скрипті бірден орындау мүмкіндігін және т.б. қамтамасыз етеді.

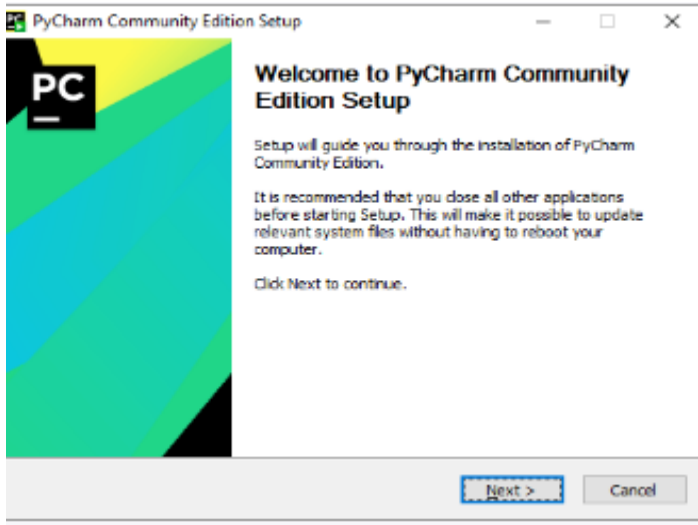
Python үшін пайдалануға болатын әртүрлі әзірлеу орталары бар, бірақ ең танымалдарының бірі - JetBrains компаниясы құрған PyCharm. Бұл орта динамикалық түрде дамып келеді, үнемі жаңартылып отырады және ең көп таралған операциялық жүйелер - Windows, MacOS, Linux үшін қол жетімді.

Рас, оның бір маңызды шектеуі бар. Атап айтқанда, ол екі негізгі нұсқада қол жетімді: ақылы кәсіби **Professional** және тегін қауымдастық **Community**. Көптеген негізгі мүмкіндіктер тегін **Community** нұсқасында да қолжетімді. Сонымен қатар, бірқатар мүмкіндіктер, мысалы, веб - әзірлеу, тек ақылы **Professional** нұсқасында қол жетімді.

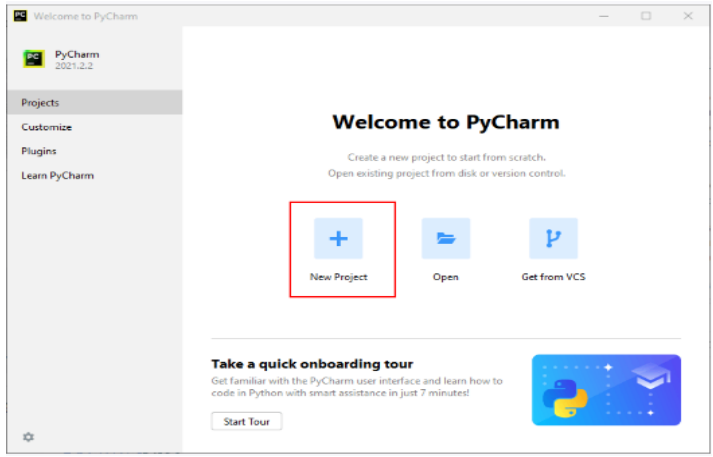
Біздің жағдайда біз тегін **Community** нұсқасын қолданамыз. Ол үшін жүктеу бетіне өтіп, PyCharm Community орнату файлын жүктеп алыңыз.



Жүктеп алғаннан кейін оны орнатамыз.



Орнату аяқталғаннан кейін бағдарламаны іске қосамыз. Іске қосқаннан кейін бастапқы терезе ашылады:



🕮 1.3 Visual Studio бағдарламасындағы Python

Python бағдарламасымен жұмыс істеуге мүмкіндік беретін әзірлеу ортасының бірі - Visual Studio. Бұл IDE - нің, айталық, PyCharm

- мен салыстырғанда артықшылығы, ең алдымен, оның Visual Studio Community тегін шығарылымында бірқатар функциялар мен мүмкіндіктер тегін қол жетімді екенін атап өткен жөн, олар PyCharm - ның Professional Edition ақылы нұсқасында ғана қол жетімді. Мысалы, бұл веб - әзірлеу, соның ішінде әртүрлі фреймворктарды пайдалану. Сонымен қатар, Visual Studio бағдарламасында Python тілінде әзірлеуге арналған құралдар қазіргі уақытта тек Windows нұсқасында қол жетімді.

2 - бөлім. PYTHON НЕГІЗДЕРІ

Python тілінде бағдарлама нұсқаулар жинағынан тұрады. Әрбір нұсқау жаңа жолға орналастырылады.

Python тілінде шегініс үлкен рөл атқарады. Қате қойылған шегініс шын мәнінде қате болып табылады. Бұл Python - ның C# немесе Java сияқты басқа бағдарламалау тілдерінен маңызды айырмашылықтарының бірі. Сонымен қатар, кодты әзірлеу нұсқаулығына сәйкес, бос орындардың санын 4 - ке көбейту арқылы (яғни, 4, 8, 16 және т.б.) жасаған жөн. 4 емес, 5 шегініс болса да, бағдарлама жұмыс істейді.

📖 2.1 Консольдік енгізу және шығару

Python бірқатар ішкі стандартты функцияларды қамтамасыз етеді. **print()** ақпаратты консольге шығарудың негізгі функциясы болып табылады. Біз шығарғымыз келетін жол осы функцияға аргумент ретінде беріледі:

```
print("Hello Python")
```

Нәтиже:

`Hello Python` мәтіні

Бұл функцияның ерекшілігі үнсіз мәндерді бөлек жолда шығарады. Мысалы:

```
print("Hello World")
print("Hello Python")
```

Консольге шығарылған кезде әрбір хабарлама бөлек жолға орналастырылады.
Нәтиже:

```
Hello World
Hello Python
```

Барлық мәндерді бір жолда көрсету үшін **end** параметрі қолданылады және ол шығарылатын жолдың соңына қосылатын таңбаларды анықтайды. Мысалы:

```
print("Hello World", end=" ")
print("Hello Python")
```

Нәтижесі:
`Hello World Hello Python`(бос орындармен бөлінген және бір жолға шығарылған)
Форматталған шығару үшін f - жолдарын (f - strings) қолдануға болады.
f - жолдар (немесе **f - strings**) - бұл Python тілінде форматталған мәтінді ыңғайлы әрі тиімді шығару үшін қолданылатын әдіс.
f - жолдар дегеніміз - жолдың алдына f әрпін қою арқылы, жолдың ішінде айнымалылар мен өрнектерді фигуралық жақшалар {} арқылы енгізуге болатын синтаксис.
Бұл тәсіл арқылы:

- Айнымалылар мәнін автоматты түрде мәтіннің ішіне енгізе аласыз.
- Форматтау(мысалы, сандарды нақты ондықтармен шығару) оңай болады.

Мысал 1:

```
name = "Мирас"
age = 20
print(f"Менің атым {name}, мен {age} жастамын.")
```

Нәтиже:

```
Менің атым Мирас, мен 20 жастамын.
```

Мысал 2:
.2f - бұл **"floating - point number"**(ондық бөлшек сан) екенін білдіреді.
.2 - санның **2 ондық таңбамен** көрсетілуін сұрайды.

```
pi = 3.14159
print(f"{pi:.2f}")
```

Нәтиже:

```
3.14          # сан тек екі ондық таңбамен дөңгелектеліп көрсетіледі
```
Мысал 3:
```
number = 123.456789
print(f"{number:.3f}")
print(f"{number:.1f}")
```
Нәтиже:
```
123.457     #  3 ондық таңба
123.5       #  1 ондық таңба
```
input() - бұл Python тіліндегі ақпаратты пернетақтадан енгізу үшін қолданылатын стандартты функция**.**

Синтаксисі:
```
айнымалы_аты = input("Хабарлама")
```
Түсіндірме:
- input() - қолданушыдан мәтіндік ақпарат енгізуді күтеді;
- Жақша ішіндегі жолдық мәтін ("Хабарлама") - бұл қолданушыға көрінетін нұсқаулық**.**
- input() әрқашан str (жол) типіндегі мән қайтарады. Егер сіз санмен жұмыс істегіңіз келсе, оны int() немесе float() түріне түрлендіру керек.

Мысал 1:
```
name = input("Атыңызды енгізіңіз: ")
print(f"Сәлем, {name}!")
```
Нәтиже:
```
Атыңызды енгізіңіз: Saliha
Сәлем, Saliha!
```
Мысал 2:
```
age = int(input("Жасыңызды енгізіңіз: "))
print(f"Сіз {age} жастасыз.")
```
Нәтиже:
```
Жасыңызды енгізіңіз: 52
Сіз 52 жастасыз.
```
Регистрге тәуелділігі

Python регистрге сезімтал**,** сондықтан print, input тағы сол сияқты барлық кілттік сөздер мен функциялар тек кішкентай әріппен жазылады.

Python тілінде:

- print
- Print
- PRINT

- бұлар үш түрлі атау болып есептеледі, және Python тек дұрыс жазылған нұсқасын таниды.

Түсініктеме(комментарий)

Түсініктемелер - бұл бағдарламаның жұмысына әсер етпейді**,** тек кодты түсіндіру немесе түсіндірме қалдыру үшін қолданылады. Олар кодпен бірге жазылады, бірақ интерпретатор оларды елемейді**.** Python тіліндегі түсініктемелердің алдына # белгісі қойылады. Оларды бөлек жолда немесе кодтан кейін орналастыруға болады:

```
# Консольге шығару
# Hello World хабарламасы
print("Hello World")
```

Python интерпретаторы егер үш тырнақша ішінде жазылған түсініктеме ешбір айнымалыға меншіктелмесе, онда оны орындамайды**,** яғни елемейді**.** Сондықтан бұл тәсілді кейбір бағдарламашылар "блоктық түсініктеме" ретінде қолданады.

```
'''
    Консольге шығару
    Hello World хабарламасы
'''
print("Hello World")
```

🕮 2.2 Айнымалылар және деректер типтері

Айнымалылар

Айнымалы (variable) - бұл ақпаратты (деректерді) уақытша сақтау үшін қолданылатын атау**.** Айнымалы атауы әріптен немесе астын сызу таңбасымен басталуы керек және тек әріптер (a - z, A - Z)**,** сандар (0 - 9) және астын сызу (_) таңбасын қамти алады.

Python тіліндегі кілттік сөздер - бұл тілдің синтаксисінде арнайы мағынасы бар сөздер, оларды айнымалылардың атауы ретінде қолдануға болмайды.

Кілт сөздер көп емес, оларды есте сақтау оңай:

False await else import pass
None break except in raise
True class finally is return
and continue for lambda try
as def from nonlocal while
assert del global not with
async elif if or yield

Мысалы, айнымалыны сипаттайық:

```
name = "Салиха"
```

Мұнда "Салиха" жолын сақтайтын name деп аталатын айнымалы анықталған.

Python - да айнымалы атауларының екі түрі қолданылады: **camel case** және **underscore notation**.

camel case айнымалы атауындағы әрбір жаңа ішкі сөз бас әріптен басталатынын білдіреді. Мысалы:

```
userName = "Салиха"
```

Underscore notation(астын сызу белгісі) айнымалы атауындағы ішкі сөздердің астын сызу арқылы бөлінгенін білдіреді. Мысалы:

```
user_name = "Салиха"
```

Сондай-ақ біз регистрге тәуелділікті ескеруіміз керек, сондықтан name және Name айнымалылары әртүрлі нысандарды көрсетеді.

```
# екі түрлі айнымалы
name = "Салиха"
Name = "Салиха"
```

Айнымалының айрықша ерекшелігі - бағдарлама жұмыс істеп тұрған кезде оның мәнін өзгертуге болады. Яғни, айнымалы бір рет қана емес, бірнеше рет әртүрлі мәнге ие бола алады.

Мысал 1:

```
name = "Томирис"  # name айнымалысы "Томирис"
мәніне тең
print(name)       # Томирис мәні шығарылады
name = "Мерей"    # мәнін "Мерей"- ге өзгертеміз
print(name)       # Мерей мәні шығарылады
```

Мысал 2:

```
age = 18        # Алғашқы мән
print(age)      # Нәтиже: 18

age = 21        # Жаңа мән берілді
print(age)      # Нәтиже: 21
```

Деректер типтері

Python бағдарламалау тілінде деректерді сақтау және өңдеу үшін көптеген әртүрлі деректер типтері қолданылады. Типтер айнымалы - ларда қандай мән сақталатынын және сол мәндермен қандай амалдар жасауға болатынын анықтайды.

Ең жиі қолданылатын негізгі(базалық) типтер:

int (бүтін сан), **float** (нақты сан), **complex** (комплекс сан), **bool** (логикалық мән), және **str** (жол/мәтін).

int - бұл бүтін сандарды білдіретін деректер типі. Ол ондық бөлшексіз сандарды сақтайды: оң, теріс немесе нөл.

Мысал 1:

```
a = 10      # оң сан
b = -25     # теріс сан
c = 0       # нөл
```

Мысал 2

```
age = 21
print("Жасы:", age)     # Жасы: 21
count = 15
print("Саны:", count)   # Саны: 15
```

Бүтін сандарды енгізу:

```
a = int (input())
```

Бүтін деректерді экранға шығару жолдары:

```
a = 1
b = 2
print(a)
print(a + b)
print(f"Қосынды = {a + b}")
```

Нәтиже:

```
1
```

```
3
Қосынды = 3
```
Мысал 3

Пайдаланушыдан екі бүтін сан енгізіп, олардың қосындысын шығаратын бағдарлама жазу.

```
# Пайдаланушыдан бүтін сандар енгізу
a = int(input("Бірінші санды енгізіңіз: "))
b = int(input("Екінші санды енгізіңіз: "))
# Қосындысын есептеу
total = a + b
# Нәтижені шығару
print(f"{a} + {b} = {total}")
```
Нәтиже:
```
Бірінші санды енгізіңіз: 8
Екінші санды енгізіңіз: 5
8 + 5 = 13
```
float - нақты (ондық бөлшек) сандар типі.

float типі 1.2 немесе 34.76 сияқты өзгермелі нүктелі сандарды білдіреді. Нүкте бүтін және бөлшек бөліктер арасындағы бөлгіш ретінде пайдаланылады.

Нақты сандарды енгізу:
```
a=float(input())
```
Мысалы:
```
height = 1.68
pi = 3.14
weight = 68.0
print(height)
print(pi)
print(weight)
```
Нәтиже:
```
1.68
3.14
68.0
```
Нақты сандарды форматтық түрде шығару:
```
print('{0:.2f}'.format(нақты сан))
```

Мысал 1

```
x = 10.0
y = 7.0
print("{0:.2f}".format(x/y))
```

Нәтиже:

```
1.43
```

Мысал 2

```
price = 1899.567
print("Баға: {0:.2f} тг".format(price))
```

Нәтиже:

```
Баға: 1899.57 тг
```

complex типі - бұл комплекстік сандарды көрсетуге арналған. Олар мынадай форматта жазылады:

complex типі - комплекстік сандарды нақты_бөлігі + жорамал_бөлігіj форматында көрсетеді, мұндағы жорамал бөлік соңынан j әрпі (суффикс) қойылады.

Мысал 1

```
# Комплекс сан жасау
a = complex(3, 4)

# Басып шығару
print("a =", a)

# Нақты бөлігі
print("Нақты бөлік:", a.real)

# Жорамал бөлігі
print("Жорамал бөлік:", a.imag)

# комплекс санның модулін табу
print("Модуль:", abs(a))
```

Нәтиже:

```
a = (3+4j)
Нақты бөлік: 3.0
Жорамал бөлік: 4.0
Модуль: 5.0
```

Мысал 2

```
x = complex(2, 3)    # 2 + 3j
y = complex(1, -1)   # 1 - 1j

z = x + y
print("Нәтиже:", z)
```

Нәтиже:

```
Нәтиже: (3+2j)
```

bool - бұл логикалық мәндердің типі. Ол тек екі мәнді қабылдай алады:

True – ақиқат, False – жалған.

Мысал 1:

```
a = True
b = False

print(a)
print(b)
```

Нәтиже:

```
True
False
```

Мысал 2:

```
x = 5
y = 10

print(x < y)
print(x == y)
```

Нәтиже:

```
True
False
```

str - бұл жолдық тип. Яғни, әріптерден, сандардан, таңбалардан құралған мәтінді білдіреді.

Мысалы:

```
name = "Медина"
print(name)
```

Нәтиже:

```
Медина
```

Жол ішінде кейбір арнайы белгілерді жазу үшін басқарушы тізбектер қолданылады. Бұл - арнайы символдар комбинациясы, олар жолда белгілі бір әрекетті білдіреді.

Тізбек	Мағынасы
\\	Жолдың ішіне кері слеш (\) қою үшін
\'	Жалқы тырнақша (') қою үшін
\"	Қос тырнақша (") қою үшін
\n	Жаңа жолға көшу(Enter сияқты)
\t	Табуляция (яғни 4 бос орындық шегініс) енгізу үшін

Мысалы:

```
text = "Message:\n\"Hello World\""
print(text)
```

Нәтиже:

```
Message:
"Hello World"
```

Жолдық типті төменде толығырақ қарастырамыз.

2.3 Арифметикалық амалдар

Python барлық арифметикалық амалдарды қолдайды:

Амал	Аты	Сипаттамасы	Мысал	Нәтиже
+	Қосу	Қосу	5 + 3	8
-	Алу	Азайту	7 -2	5
*	Көбейту	Көбейту	4 * 3	12
/	Бөлу	Нақты санды нақты санға бөлу	7 / 2	3.5
//	Бүтін бөлу	Бүтін санды бүтін санға бөлгендегі бүтін бөлік	7 // 2	3
%	Қалдық	Бүтін санды бүтін санға бөлгендегі қалдық бөлік	7 % 2	1
**	Дәреже	Бір мәнді екіншісінің дәрежесіне шығару	2 ** 3	8

Мысалы: Нәтиже

```
print(6 / 2)     3.0
print(7 / 2)     3.5
print(7 // 2)    3
print(7 % 2)     1
print(6 ** 2)    36
```

Құрама меншіктеу операторлары

Python тілінде айнымалыға мән меншіктеу кезінде арифметикалық амалдарды бірден орындауға болады. Мұны құрама меншіктеу операторлары деп атайды.

Оператор	Математикада	Мысал
+=	x = x + 5	x += 5
-=	x = x - 3	x -= 3
*=	x = x * 2	x *= 2
/=	x = x / 4	x /= 4
//=	x = x // 2	x //= 2
%=	x = x % 3	x %= 3
**=	x = x ** 2	x **= 2

Мысалы:

```
x = 10
x += 5
x *= 2
x -= 10
x /= 4
x //= 5
x %= 5
x **= 2
print(x)
```

Нәтиже:

```
x = x + 5 → 15
```

```
x = x * 2 → 30
x = x - 10 → 20
x = x / 4 → 5.0
x = x // 5 → 2
x = x % 5 → 0
x = x ** 2 → 100
```

Python - да кірістірілген math модулі математикалық, тригонометриялық және логарифмдік операцияларды орындауға арналған функциялар жиынын ұсынады.

Негізгі функциялары:

- pow(num, power) - num санын power дәрежесіне шығарады;
- sqrt(num) - num санының квадрат түбірі;
- ceil(num) - жоғары жаққа дөңгелектеу (ең кіші бүтін саннан үлкен жаққа);
- floor(num) - төменгі жаққа дөңгелектеу (ең үлкен бүтін саннан кіші жаққа);
- factorial(num) - санның факториалы (мыс: 5! = 5×4×3×2×1);
- degrees(rad) - радианды градусқа айналдырады;
- radians(grad) - градус мәнін радианға айналдырады.

Тригонометриялық функциялар (радианмен жұмыс істейді):

- cos(rad) - косинус бұрышы (радианмен);
- sin(rad) - синус бұрышы (радианмен);
- tan(rad) - тангенс бұрышы (радианмен);
- acos(rad) - арккосинус бұрышы (радианмен);
- asin(rad) - арксинус бұрышы (радианмен);
- atan(rad) - арктангенс бұрышы (радианмен);

Логарифмдік функциялар:

- log(n, base) - n санының base негізіндегі логарифмі;
- log10(n) - ондық логарифм, яғни log(n, 10)

Математикалық функцияларды қолдану үшін алдымен, math модулін бағдарламаға импорттау керек:

```
import math
```

Бұл модуль көптеген математикалық функцияларды қамтиды.

math модулінің кейбір маңызды функциялары:

1. math.sqrt(x) - квадрат түбір табады

```
a = 16
z = math.sqrt(a)
print(f"{a} санының квадрат түбірі: {z}")
```

Нәтижесі:

```
16 санының квадрат түбірі: 4.0
```

2. math.pow(x, y) - x-тің y дәрежесін есептейді

```
a=2
b=3
z = math.pow(a, b)
print(f"{a} санының {b} дәрежесі: {z}")
```

Нәтижесі:

```
2 санының 3 дәрежесі: 8.0
```

3. math.factorial(x) – факториал есептейді

```
a = 5
fact = math.factorial(a)
print(f"{a} санының факториалы: { fact }")
```

Нәтижесі:

```
5 санының факториалы: 120
```

4. math.sin(x), math.cos(x), math.tan(x) - тригонометриялық функциялар

```
x = math.pi / 4
result_c = math.sin(x)
result_k = math.cos(x)
print(f"Синус: {result_c}")
print(f"Косинус: {result_k}")
```

Нәтижесі:

```
Синус: 0.7071067811865475
Косинус: 0.7071067811865476
```

5. math.log10(x) – 10 негізіндегі логарифм

```
a = 100
logarithm = math.log10(a)
print(f"{a} санының 10 негізіндегі логарифмі:
{logarithm}")
```

Нәтижесі:

```
100 санының 10 негізіндегі логарифмі: 2.0
```

6. math.log(x) - x санының натурал логарифмін есептейді

```
x = 10
natural_log = math.log(x)
print(f"ln({x}) = {natural_log }")
```

Нәтижесі:

```
ln(10) = 2.302585092994046
```

Негізі 2 болатын логарифмді есептеу үшін мына формуланы қолдануға болады:

```
x = 8
base = 2
logarithm = math.log(x) / math.log(base)
print(f"log{base}({x}) = {logarithm}")
```

Нәтижесі:

```
log2(8) = 3.0
```

7. math.pi - π саны (3.141592653589793).
8. math.e - e саны (2.718281828459045).
9. math.ceil(x) - x санын жоғары дөңгелектейді (яғни, ең кіші бүтін санға дейін көтереді)

```
x = 3.14
result = math.ceil(x)
print(result)
```

Нәтиже:

```
4
```

10. math.floor(x) - x санын төмен дөңгелектейді (ең үлкен бүтін санға дейін)

```
x = 3.7
print("ceil:", math.ceil(x))
print("floor:", math.floor(x))
```

Нәтиже:

```
ceil: 4
floor: 3
```

11. math.fabs(x) - x нақты санының абсолюттік мәнін (модулін) қайтарады

```
x = -5.7
absolute = math.fabs(x)
print(f"|{x}| = {absolute}")
```

Нәтиже:
|-5.7| = 5.7

Тапсырмалар

1. Студент туралы тиісті мәліметтерді сұрайтын интерактивті бағдарлама жазыңыз (аты-жөні, туған күні, мекен-жайы (тек қаланы ғана алуға болады), сүйікті іс-әрекеті, сүйікті тағамы, сүйікті фильмі және т.б.), содан кейін оларды кесте түрінде экранға шығару.
2. Үштаңбалы сан берілген. Сол санның цифрларының қосындысын табу.
3. Z=5xy+x функциясының t=2 болғандағы мәнін есептеу, мұндағы, y=5x-2; x=0.2t+2t;
4. F=z(x-y) функциясының t=0.02 болғандағы мәнін табу керек, мұндағы z=5xy+sin22y, y=x2+5x+lg4x, x=0.2t3+2t;
5. Қосымша айнымалыны қолданып, x және y айнымалыларының мәндерінің орындарын ауыстыру;
6. Бір зат алдымен 10%-ға қымбаттап, одан соң 10%-ға арзандатылса, заттың бағасы қанша %-ға өзгергенін анықтау;
7. Топта N білім алушы. Бақылау жұмысынан кейін мынадай бағалар алынды: A – бес, B – төрт, C – үш. Үштердің, төрттердің, бестердің пайызын табу;
8. Қолданушыдан екі бүтін санды енгізуді сұрайтын және олардың қосындысының квадраты $(a+b)^2$ мен квадраттарының қосындысын a^2+b^2 шығаратын бағдарлама құру. Бағдарламаның экран көрінісі: Екі сан енгізіңіз: a=3, b=2
   3 және **2** қосындыларының квадраты **25** тең
   3 және **2** квадраттарының қосындысы **13** тең
9. Бүтін оң төрторынды сан берілген. Осы санның цифрларының қосындысын және көбейтіндісін табу;
10. Берілген радиус бойынша шеңбердің ұзындығын, дөңгелектің ауданын және шардың көлемін есептеу;
11. Тәулік басынан N секунд өтті (N бүтін сан). Тәулік басынан өткен толық минуттар санын табУ;
12. Автомобиль үш түрлі аумақтағы жолды әртүрлі жылдамдықпен жүріп өтті. Автомобильдің орташа жылдамдығын табу.

13. Студенттің сүйікті афоризмін немесе оның ұранын экранға шығаратын бағдарламаны және автор туралы ақпаратты, мысалы, студенттің инициалдарын бөлек жолда жазыңыз.

Тест тапсырмалары

1. Сандық мәнді сақтайтын айнымалы атауын таңдаңыз
book = "The Hobbit"
pages = 310
A. book
B. pages
C. The Hobbit
D. Hobbit
ANSWER: B
2. print() операторы қызметі
A. Экранға мәнді шығарады
B. Айнымалыны өзгертеді
C. Мәнді экранда сақтайды
D. Мәнді енгізеді
ANSWER: A
3. Төмендегі код нәтижесі
budget = 200
print(budget)
A. budget
B. 200
C. print(budget)
D. 2
ANSWER: B
4. Төмендегі код орындалғанда экранда көрінетін 2 мән
username = “magician”
points = 50
lives = 3
print(username)
print(points)
A. print
B. 50
C. magician

D. lives
E. points
F. 3
G. username
ANSWER: B, C
5. Код орындалғанда экранда көрінетін 2 мән
name = “Tom”
level = 14
print(name)
level = level + 1
print(level)
A. level
B. 15
C. 14
D. Tom
E. name
F. 1
ANSWER: B, D
6. Мәні 100 болатын score айнымалысының дұрыс сипаттамасы
A. 100 = score
B. score 100
C. score = 100
D. 100 score
ANSWER: C
7. Мәтін бөлігін көрсететін деректер типі
A. Мәтін
B. Сөз
C. Жол
D. Символ
ANSWER: C
8. Төмендегі код нәтижесі
print(“Game Over”)
A. Game Over
B. қате
C. ештеңе
D. Game

ANSWER: C
9. Төмендегі кодтағы айнымалылар саны
credit = 300
Credit = 280
CREDIT = 320
A. 1
B. 2
C. 3
D. 6
ANSWER: C
10. Underscore notation түріндегі айнымалы атауы
A. dog_name
B. dog-name
C. DogName
D. dog*name
ANSWER: A
11. Пайдаланушы id - ін сақтайтын айнымалыны сипаттаңыз
A. user*id
B. user/id
C. user#id
D. user_id
ANSWER: D
12. Төмендегі код нәтижесі
a = 3
a = 5
a = 7
print(a)
A. 5
B. 7
C. a
D. 3
ANSWER: B
13. Төмендегі код нәтижесі
user_entry = input()
A. Қате код
B. Экранда "user_entry" хабарламасын көрсетеді

C. Пайдаланушыдан user_entry деп аталатын айнымалыда сақталатын мәнді сұрайды
D. Файлға ақпарат жазады
ANSWER: C
14. Жолды таңдаңыз
A. 2
B. “nintendo”
C. 2.5
D. username
ANSWER: B
15. Бүтін мәнді таңдаңыз
A. 3.14
B. 2
C. “Apple”
D. -5.7
ANSWER: B
16. Нақты санды таңдаңыз
A. 6.8
B. 0
C. 4
D. “Number”
ANSWER: A
17. Код нәтижесі
variable = 5/2
A. “2.5”
B. 2
C. 2.5
D. 1
ANSWER: C
18. Төмендегі код нәтижесі
a = “basket”
b = “ball”
print(a + b)
A. basketball
B. ball
C. қате

D. basket
ANSWER: A
19. Айнымалыда сақталған мән
var = “360”
A. Жол
B. Бүтін сан
C. Нақты сан
D. Символ
ANSWER: A
20. Код нәтижесі
print(“360” + “360”)
A. “720”
B. 360360
C. 720
D. 360
ANSWER: B
21. Төмендегі код нәтижесі
x = 9
y = 3
print(x / y)
A. 3.0
B. 3
C. 9
D. 27.0
ANSWER: A
22. Деректерді бүтін санға түрлендіру
A. number()
B. num()
C. float()
D. int()
ANSWER: D
23. Төмендегі код нәтижесі
print(14 + “km”)
A. 14km
B. 14
C. қате

D. 14+km
ANSWER: C
24. Төмендегі код нәтижесі
print(“14” + “km”)
A. 14
B. Қате
C. 14km
D. 14+km
ANSWER: C
25. Төмендегі код нәтижесі
x = 9
y = 3.0
print(x + y)
A. 9.0
B. 12
C. 12.0
D. 9
ANSWER: C
26. Төмендегі код нәтижесі
print(3 * ”7”)
A. 777
B. 3”7”
C. “777”
D. 37
ANSWER: A
27. Төмендегі код нәтижесі
print(20**3/100)
A. 80
B. 0.6
C. 80.0
D. 2
ANSWER: C
28. Төмендегі код нәтижесі
print((2**2)**(12%10))
A. 16
B. 64

C. 88
D. 32
ANSWER: A
29. Төмендегі код нәтижесі
print(1+2+3+4%5)
A. 0
B. 2
C. 2.0
D. 10
ANSWER: D
30. Төмендегі код нәтижесі
print((5+5)*20/(5+5))
A. 20.0
B. 20
C. 10
D. 10.0
ANSWER: A
31. Код нәтижесі
a=3
b=6
print(a%b)
A. 6
B. 2
C. 3
D. 63
E. 1
ANSWER: C
32. Код нәтижесі
print(1+2*3-4**(4-2))
A. -9
B. -7
C. 7
D. 9
E. 4
ANSWER: A
33. Код нәтижесі

print(“5” * 2)
A. 0
B.2
C. 5
D. 10
E. 55
ANSWER: E
34. Код нәтижесі
print(5%4+2**2)
A. 0
B.2
C. 5
D. 10
E. 55
ANSWER: C
35. Код нәтижесі
print (int (9%((5//2)+ (4/2))))
A. 1.0
B. 2
C. 2.5
D. 1
E. 9
ANSWER: D
36. Код нәтижесі
print (1+2+3+4%5)
A. 1
B. 11
C. 10
D. 5
E. 2
ANSWER: C
37. Код нәтижесі
print(3+float (2)*4)
A. 11
B. 20.0
C. 11.0

D. error
E. 10
ANSWER:C
38. Бүтін санды таңда
A. 9.57
B. "22"
C. 530
D. "Moonlight"
E. True
ANSWER: C

🕮 2.4 Шартты және тернарлы операторлар

2.4.1 Салыстыру амалдары

- > (үлкен);
- >= (артық немесе тең);
- < (кіші);
- <= (кіші немесе тең);
- = = (тең);
- != (тең емес)

Мысалы:

```
a = 5
b = 6
result = 5 = = 6 # амал нәтижесін айнымалыға сақтау
print(result)     # нәтиже False - 5 саны 6 ға тең емес
print(a != b)     # нәтиже True
print(a > b)      # нәтиже False - 5 саны 6 дан кіші
print(a < b)      # нәтиже True - 5 саны 6 дан кіші
bool1 = True
bool2 = False
print(bool1 = = bool2)  # нәтиже False - bool1 bool2 ге тең емес
```

Салыстыру амалдары әртүрлі объектілерді - жолдарды, сандарды, логикалық мәндерді салыстыра алады, бірақ амалдың екі операнды да бір типте болуы тиіс.

2.4.2 Логикалық амалдар

Құрама шартты өрнектерді құру үшін логикалық амалдар қолданылады. Python - да келесі логикалық амалдар бар:

- and – логикалық көбейту «ЖӘНЕ»

Мысалы:

```
age = 22
weight = 58
result = age > 21 and weight == 58
print(result)
```

Нәтиже: `True`

- or – логикалық қосу «НЕМЕСЕ»

Мысалы:

```
age = 22
isMarried = False
result = age > 21 or isMarried
print(result)
```

Нәтиже:

`True`, age > 21 өрнегі мәні True болғандықтан

- not – логикалық терістеу «ЕМЕС»

Мысалы:

```
age = 22
isMarried = False
print(not age > 21)    # Нәтиже False
print(not isMarried)   # Нәтиже True
print(not 4)           # Нәтиже False
print(not 0)           # Нәтиже True
```

x	y	not x	x or y	x and y
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

Ақиқаттық кестесі

2.4.3 Шартты оператор

Шартты оператор - белгілі бір шарттың орындалуына байланысты бағдарламаның орындалу бағытын өзгерту үшін қолданылатын оператор. Python тілінде шартты операторлар if, elif және else кілт сөздері арқылы жүзеге асады.
Синтаксисі:

```
if логикалық_өрнек:
    нұсқаулар
[elif логикалық өрнек:
    нұсқаулар]
[else:
    нұсқаулар]
```

if кілттік сөзінен кейінгі логикалық өрнек мәні True ақиқат болса, онда келесі нұсқаулар блогы орындалады, олардың әрқайсысы жаңа жолдан басталуы керек және if өрнегі басынан шегінуі керек (оны 4 бос орынмен немесе 4-ке еселік санмен шегініс жасаған жөн.)
Мысалы: a және b сандары 0 - мен аяқтала ма тексеріңіз:

```
a = int(input())
b = int(input())
if a % 10 = = 0 and b % 10 == 0:
    print('YES')
else:
    print('NO')
```

Нәтиже:

```
10
16
YES
```

else блогы

Егер if логикалық өрнегінің мәні False жалған болса, онда else блогы орындалады.
Мысалы:
a санының жұптығын тексеру

```
a = int(input())
if a % 2 == 0:
    print('жұп сан')
else:
    print('тақ сан')
```
Нәтиже:
```
15
тақ сан
```
Сонымен қатар, else блогының нұсқаулары да жолдың басынан шегінуі керек.

elif блогы

Егер бірнеше балама шарттарды енгізу қажет болса, онда қосымша elif блоктарын, содан кейін нұсқаулар блогын пайдалануға болады.

Python алдымен if логикалық өрнегін тексереді. Егер ол True болса, онда if блогындағы нұсқаулар орындалады. Егер бұл шарт False мәнін қайтарса, Python elif ішіндегі өрнекті тексереді.

Егер elif-тен кейінгі өрнек True болса, онда elif блогының нұсқаулары орындалады. Ал егер ол False болса, else блогының нұсқаулары орындалады.

Қажет болса, әртүрлі шарттар үшін бірнеше elif блоктарын анықтауға болады.

Мысалы:
```
language = "german"
if language == "english":
    print("Hello")
elif language == "german":
    print("Hallo")
elif language == "french":
    print("Salut")
else:
    print("Сәлем")
```
Нәтиже:

```
Hallo
```

2.4.4 Кіріктірілген шартты оператор

if конструкциясының өзінде кіріктірілген if конструкциялары болуы мүмкін. Олар арқылы күрделірек логикаларды жүзеге асыруға болады.

Мысалы

(x, y) координатасымен нүкте берілген. Осы нүкте жататын жазықтық ширегінің нөмерін экранға шығаратын бағдарлама жазу.

```
x = float(input("x координатасын енгізіңіз: "))
y = float(input("y координатасын енгізіңіз: "))
if x > 0:
    if y > 0:                       # x > 0, y > 0
        print("1 - ширек")
    else:                           # x > 0, y < 0
        print("4 - ширек")
else:
    if y > 0:                       # x < 0, y > 0
        print("2 - ширек")
    else:                           # x < 0, y < 0
        print("3 - ширек")
```

Нәтиже:

```
x координатасын енгізіңіз: 10
y координатасын енгізіңіз: 5
1 - ширек
```

2.4.5 Тернарлы оператор

Тернарлы оператор Python-да if - else операторын қысқа түрде жазуға мүмкіндік береді. Тернарлы оператор қысқаша шарттарды тексеру үшін өте ыңғайлы.

Синтаксисі:

```
value_if_true if condition else value_if_false
```

– condition - тексерілетін шарт (True немесе False).

– value_if_true - шарт орындалса (True болса) қайтарылатын мән.
– value_if_false - шарт орындалмаса (False болса) қайтарылатын мән.

Мысалы:

```
x = 10
result = "Оң сан" if x > 0 else "Терiс сан"
print(result)
```

Нәтиже:

```
Оң сан
```

Тапсырмалар

1. a, b, c нақты сандары берілген. Егер a≥b≥c болса, онда осы сандарды екі еселеңдер, әйтпесе оларды абсалют мәндерімен ауыстыру;
2. a, b, c, d нақты сандары берілген. Егер a<b<c<d болса, онда олардың әрқайсысын ең үлкенімен ауыстырыңыз. Егер a>b>c>d болса, онда сандарды өзгеріссіз қалдырыңыз. Кері жағдайда барлық сандарды олардың квадраттарымен ауыстыру;
3. Координатасы x және y болатын нүкте жататын жазықтық ширегінің нөмерін k айнымалысына меншіктеу;
4. Би мектебіне бойы 168 см-ден қысқа және 178 см-ден ұзын емес қыздар мен ер балалар қабылданады. Олардың салмақтары бойларымен мына формула бойынша сәйкес келу керек: салмақ мәні ≤ (бой мәні - 115). Түсуші би мектебіне қабылдана ма, қабылданбай ма анықтау;
5. Екі сан берілген. Егер бірінші санның абсалюттік шамасы екінші санның абсалюттік шамасынан үлкен болса, онда бірінші санды бес есе кішірейтіп, ал кері жағдайда, оларды өзгертпей жазатын бағдарлама құру;
6. Квадрат теңдеудің түбірін табуға бағдарлама құру;
7. n (n≤9999) натурал саны берілген. Төрттаңбалы санның полиндром болып табылатынын анықтау, мысалы, 2222, 6116, 0440 және т.с.с.

8. Егер цифрларының қосындысы 3-ке бөлінсе, онда сан 3-ке бөлінеді. Осы шартты берілген үштаңбалы санның мысалында тексеру;
9. Үш бүтін сан берілген. Олардың ішінен -10-нан +10-ға дейінгі аралықта жататындарын табу;
10. Нүктенің координаталарын (x, y) және шеңбердің радиусын r енгізіңіз. Егер шеңбердің центрі координат басында жататын болса, онда берілген нүктенің шеңберге жататынын анықтайтын бағдарлама жазу;
11. Р бағаны енгізіп, шарттарға сәйкес нәтиже алатын бағдарлама жазу. Егер баға 90≤Р≤100 аралығында болса, «Жарайсың!», , егер 70≤Р≤89 болса, «Жақсы!» және егер 50≤Р≤69 болса «Нашар!»
12. x саны [2, 5] немесе [-1, 1] аралығында жататындығын анықтау;
13.

$$p=\begin{cases}\sqrt{|a\cdot b|}+2\cdot c, a\cdot b<-2\\ a^3+b^2-c^2, -2\le a\cdot b\le 2\\ a^c-b, a\cdot b>2\end{cases}$$

Тест тапсырмалары

1. 551 < 485 салыстыру амалының нәтижесі
A. False
B. True
C. 551
D. 485
ANSWER: A
2. True мәнін шығару үшін салыстыру амалын таңдаңыз print(70 30)
A. <
B. >
C. =
D. >=
ANSWER: B
3. Код нәтижесі

print(3 > 15)
print(3 < 15)
A. False,True
B. True, False
C. False, False
D. True, True
ANSWER: A
4. Кодта True мәнін шығаратын салыстыру амалын таңдаңыз
soil_moisture = 80
print(soil_moisture…100)
A. <
B. >
C. 80
D. 100
ANSWER: A
5. Код нәтижесі
soil_moisture = 80
soil_moisture = 120
print (soil_moisture < 100)
A. 120
B. True
C. 80
D. False
ANSWER: D
6. Логикалық амалдың нәтижесі
A. Бүтін сан
B. Ондық бөлшек
C. Логикалық мән
D. Жол
ANSWER: C
7. Код нәтижесі
light_on = True
door_locked = False
print(light_on or door_locked)
A. or
B. and

C. False
D. True
ANSWER: D
8. Код нәтижесі
a = (3 > 2) or False
A. or
B. 2
C. False
D. True
ANSWER: D
9. Код нәтижесі
active = True
registered = False
print (active or registered)
A. not
B. True
C. False
D. or
ANSWER: B
10. Код нәтижесі
password = “SecretWord”
guess = “1234”
print(guess != password)
A. True
B. False
C. Қате
D. guess
ANSWER: A
11. Код нәтижесі
age = 16
if age >= 18:
    Print(“Regular price”)
else:
    Print(“Discount”)
A. Regular price
B. Discount

C. 16
D. 18
ANSWER: B
12. True мәнін алу үшін салыстыру амалын таңдаңыз
is_student = True
age = 20
is_student __ (age < 18)
A. or
B. and
C. not
D. +
ANSWER: A

🕮 2.5 Цикл операторлары

Циклдер қандай да бір шартты қанағаттандыруға байланысты кейбір әрекеттерді қайталап орындауға мүмкіндік береді. Python-да цикл операторларының келесі түрлері бар: **while, for.**

2.5.1 while циклы

Қайталану саны алдын ала белгісіз болғанда және қайталау шарты циклдің басында тексерілетін болса - онда while циклы қолданылады.

while циклі қандай да бір шарттың ақиқаттығын тексереді, егер шарт ақиқат болса, ол цикл нұсқауларын орындайды.

while циклы операторының синтаксисі:

```
while шартты өрнек:
    нұсқаулар
```

Орындалу тәртібі:

– алдымен шартты өрнек тексеріледі;
– егер шартты өрнектің мәні ақиқат болса, онда нұсқаулар - қайталанатын код блогы орындалады;
– жалған болған жағдайда цикл жұмысын тоқтатады.

while цикліне қатысты барлық операторлар келесі жолдарда орналасады және while кілт сөзінің басынан шегініс болуы керек.

```
number = 1
while number < 5:
    print(f"number = {number}")
    number += 1
print("Бағдарлама жұмысы аяқталды")
```
Нәтиже:
```
number =1
number =2
number =3
number =4
Бағдарлама жұмысы аяқталды
```

Бұл жағдайда while циклі әзірге айнымалы number 5-тен кіші болғанша орындалады.
Цикл блогының өзі екі нұсқаудан тұрады:
```
print(f"number = {number}")
number += 1
```
Назар аударыңыз, олар while операторының басынан - бұл жағдайда жолдың басынан шегінеді. Бұл Python-ға олардың циклге жататынын анықтауға мүмкіндік береді. Циклдің басында алдымен number айнымалысының бастапқы мәні көрсетіледі, содан кейін циклдің ішінде оған жаңа мән тағайындалады.

Сондай-ақ, print("Бағдарлама жұмысы аяқталды ") соңғы операторы жолдың басынан шегініс жасалмағанын, сондықтан ол while циклінің бөлігі емес екенін ескеріңіз.
Бүкіл цикл процесін келесі түрде көрсетуге болады:

1. Алдымен, number айнымалысының мәні 5-тен кіші ме екені тексеріледі. Ал айнымалы мәні бастапқыда 1-ге тең болғандықтан, бұл шарт True мәнін қайтарады, сондықтан цикл нұсқаулары орындалады. Цикл нұсқаулары консольге number = 1 жолын шығарады, содан кейін number айнымалысының мәні бірге артады - енді ол 2-ге тең. Цикл нұсқауларының блогын бір рет орындау итерация деп аталады. Яғни, осылайша циклде бірінші итерация орындалады.
2. number < 5 шарты қайтадан тексеріледі, оның мәні True, өйткені number = 2, сондықтан цикл нұсқаулары тағы орындалады

Цикл нұсқаулары консольге number = 2 жолын шығарады Ал содан кейін number айнымалысының мәні қайтадан артады - енді ол 3-ке тең. Осылайша, екінші итерация орындалады.

3. number < 5 шарты қайтадан тексеріледі, оның мәні бұрынғыша True, өйткені number = 3, сондықтан цикл нұсқаулары тағы орындалады

Цикл нұсқаулары консольге number = 3 жолын шығарады Ал содан кейін number айнымалысының мәні қайтадан артады - енді ол 4-ке тең. Осылайша, үшінші итерация орындалады.

4. number < 5 шарты қайтадан тексеріледі, оның мәні бұрынғыша True, өйткені number = 4, сондықтан цикл нұсқаулары тағы орындалады

Цикл нұсқаулары консольге number = 4 жолын шығарады Ал содан кейін number айнымалысының мәні қайтадан артады - енді ол 5-ке тең. Осылайша, үшінші итерация орындалады.

5. number < 5 шарты қайтадан тексеріледі, енді оның мәні False, өйткені number = 5, сондықтан циклден шығу орындалады.

Әрі қарай циклден кейінгі әрекеттер орындалады. Осылайша, бұл цикл төрт итерация жасайды.

while циклі үшін шарттың мәні False болғанда, қосымша else блогын қолдануға болады:

```
number = 1
while number < 5:
    print(f"number = {number}")
    number += 1
else:
    print(f"number = {number}. Цикл жұмысы
аяқталды")
print("Бағдарлама жұмысы аяқталды")
```

Яғни, бұл жағдайда алдымен шарт тексеріліп, while операторлары орындалады. Содан кейін шарт False мәніне тең болған кезде else блогындағы операторлар орындалады. else блогындағы нұсқаулар да цикл құрылымының басынан шегініске ие екенін ескеріңіз. Бұл жағдайда консольге төмендегідей нәтиже шығарылады:

```
number = 1
number = 2
```

```
number = 3
number = 4
number = 5. Цикл жұмысы аяқталды
Бағдарлама жұмысы аяқталды
```

Егер шарт бастапқыда False болса, else блогы пайдалы болуы мүмкін және бұл жағдайда кейбір әрекеттерді орындай аламыз:

```
number = 10
while number < 5:
    print(f"number = {number}")
    number += 1
else:
    print(f"number = {number}. Цикл жұмысы
аяқталды ")
print("Бағдарлама жұмысы аяқталды")
```

Бұл жағдайда number < 5 шарты басынан False болады, сондықтан цикл бірде бір итерацияны орындамайды және бірден else блогына өтеді.

Мысалдар:

Экранға і мәндерін шығару:

```
i = 0
while i < 4: # цикл әзірше і 4-тен кіші болғанша
орындалады
     print(f"{i}")
     i+=1
```

Нәтиже:

```
0
1
2
3
```

«Blastoff!» сөзін экранға шығару

```
n = 5
while n > 0:
    print(n)
    n -= 1
print('Blastoff')
```

Нәтиже:

```
5
4
3
2
1
Blastoff
```

else блогының қалай жұмыс істейтінін мысалда көрсетейік:

```
i = 0
while i < 4: # цикл әзірше i 4-тен кіші болғанша
орындалады
     print(f"{i}")
     i+=1
else:
     print(f"{i} >= 4 ")
```

Нәтиже:

```
0
1
2
3
4>=4
```

2.5.2 break, continue операторлары

Python - да циклдардың әдепкі әрекетін өзгертуге мүмкіндік беретін бірнеше операторлар бар.

break операторы

break операторы циклды ертерек тоқтатуға мүмкіндік береді:

- break ағымдағы циклды үзеді және келесі өрнектерді орындауды жалғастырады;
- егер бірнеше кірістірілген циклдар қолданылса, break ішкі циклды бұзады және блоктан кейінгі өрнектерді орындауды жалғастырады;
- break for және while циклдарында қолданылуы мүмкін.

continue операторы

continue операторы циклдің келесі итерациясына өтуді орындайды. Яғни, continue циклдегі қалған өрнектерден «секіруге» және келесі итерацияға өтуге мүмкіндік береді.

Мысалы:

break операторын қолдану

```
i = 0
while i < 10:
     if i = = 5:
        break
else:
    print(i)
    i += 1
```

Код нәтижесі:

```
0
1
2
3
4
```

Мысалы:

break және continue операторларын қолдану

```
while True:
     i = int(input())
     if i == 10:
          break                    #циклді аяқтау
               if i > 10:
                    print(f"Сіз сан енгіздіңіз")
                    continue
     print(f"{i} – бұл цифр")
```

Бағдарлама сандарды енгізуді ұсынады. Егер цифр енгізілсе (0-ден 9-ға дейін), бағдарлама цифрдің енгізілгенін хабарлайды. Егер сан енгізілсе (10-нан үлкен), бағдарлама сан енгізілгенін хабарлайды. 10 саны енгізілгенде циклден шығады.

while циклін қолданған кезде абай болыңыз, өйткені ол шексіз цикл жасай алады.

```
while True:
```

```
    print(f"Мені тоқтату үшін Ctrl - C пернелер тіркесін басыңыз")
```

Бұл мысалдағы циклды тек Ctrl - C пернелер тіркесін басу арқылы тоқтатуға болады.

Тапсырмалар

1. 0 енгізілгенге дейін пайдаланушыдан сандар тізбегін енгізуді сұрайтын, содан кейін бағдарлама енгізілген барлық сандардың қосындысын консольге шығаратын бағдарлама жазу;
2. Санның түбірін табу: Қолданушыдан санды сұрайтын және оның квадрат түбірін берілген дәлдікпен табатын бағдарлама жазу. Түбірді жуықтап есептеу үшін while циклды пайдалану;
3. «Санды тап» ойыны(мүмкіндіктердің шектеулі санымен): компьютер кездейсоқ санды жасыратын және пайдаланушы оны табатын бағдарлама жазу. Пайдаланушыға санды табу үшін белгілі бір мүмкіндіктер санын (мысалы, 5) беру;
4. Санның факториалы: Пайдаланушыдан N санын сұрайтын және оның факториалы (N!) есептейтін бағдарлама жазу. Факториалды есептеу үшін while циклды қолдану;
5. Пайдаланушыдан санды сұрайтын және while циклын қолданып оның жай сан екенін анықтайтын бағдарлама құру;
6. Python тілінде банк шотын симуляциялайтын бағдарлама. Бағдарлама пайдаланушыға шотына ақша салуға және алуға мүмкіндік береді. Пайдаланушы шыққанша цикл арқылы транзакциялар орындалады;
7. Кездейсоқ құпия сөз генераторы: берілген ұзындықтағы кездейсоқ құпия сөздерді генерациялайтын бағдарлама жасау. Белгіленген ұзындықтағы құпия сөз алынғанша құпия сөздерді генерациялау үшін while циклды пайдалану;
8. Бағдарлама экранда келесі ретті көрсетуі қажет:

   7 14 21 28 35 42 49 56 63 70 77 84 91 98
9. Фибоначчи сандар тізбегін бағдарламалау (пайдаланушы Фибоначчи тізбегі элементінің реттік нөмірін енгізеді, ал бағдарлама мәнді шығару қажет).

10. Консольге келесі сандар тізбегін шығару:

1 2 4 8 16 32 64 128 256 512

11. [1; 20] аралығында b=2 қадаммен өзгеретін y= 5x -2 функциясы мәндерінің кестесін шығару;
12. Соңы 0 болатын сандар тізбегі берілген. Осы сандардың ең кішісінің реттік номерін анықтау;
13. Соңы 0 болатын, 0-ден өзгеше сандар тізбегі берілген. Осы тізбектегі теріс сандардың санын және оң сандардың қосындысын табу.

Тест тапсырмалары

1. Код нәтижесі

```
i = 0
while i < 5:
        i += 1
if i ==3
break else:
print (0)
```

A. 0
B.11
C.22
D. 0
ANSWER: D

2. Код нәтижесі

```
x = 5
while (x>=0):
        x=x-1
        print(x)
```

A. 2
B. -1
C. 0
D. -2
ANSWER: B

3. «Sell Coffee» хабарламасы … рет көрсетіледі

```
cups = 30
```

```
while cups > 0:
        print(“Sell Coffee”)
        cups = cups – 1
```
A. 10
B. 1
C. 30
D. 12
ANSWER: C
4. Код нәтижесі
```
count = 1
while count < 10:
        print(count)
```
A. Қате
B. Шексіз цикл
C. 10 итерация
D. 9 рет
ANSWER: B
5. Шарт жалған болғанда, while циклынан шығу
```
seats = 300
while seats > 0:
          print ("Sell ticket")
          seats = seats - 1
```
A. seats = 1
B. seats = 0
C. seats = 2
D. seats = 3
ANSWER: B

2.5.3 for циклы

Циклдің тағы бір түрі for циклы болып табылады. Бұл цикл мәндер жиыны арқылы қайталанады, әрбір мәнді айнымалыға меншіктейді, содан кейін циклде сол айнымалымен әртүрлі әрекеттерді орындай аламыз.

Python тіліндегі for циклінің кез келген күрделі деректер типтерінің элементтерін (мысалы, жол немесе тізім) қайталау мүмкіндігі бар.

For циклінің синтаксисі:

```
for айнымалы in мәндер_жиынтығы:
    нұсқаулар
```

Мәндер жиынтығын, мысалы, таңбалар жиынын білдіретін жол деп санауға болады. Мысал қарастырайық:

```
message = "Hello"
for c in message:
    print(c)
```

Нәтиже:

```
H
e
l
l
o
```

Python тіліндегі range() функциясы белгілі бір сандар диапазонын (аралығын) жасау үшін қолданылады. range(бастау, тоқтау, қадам) Python тіліндегі range() функциясын шақыру осылай көрінеді.

Синтаксисі:

```
range(тоқтау)
range(бастау, тоқтау)
range(бастау, тоқтау, қадам)
```

Параметрлер:

- бастау(*start*) - диапазонның басы (үнсіз 0);
- тоқтау(*stop*) - диапазонның соңы(енбейді);
- қадам(*step*) - әр қадамда қаншаға өсетінін немесе кемитінін көрсетеді (әдепкісі - 1).

Мысалдар:

1. Тек тоқтау параметрі:

```
for i in range(5):
    print(i)
```

Нәтиже:

```
0
1
```

```
2
3
4
```
2. Бастау мен тоқтау:
```
for i in range(2, 6):
    print(i)
```
Нәтиже:
```
2
3
4
5
```
3. Қадаммен:
```
for i in range(1, 10, 2):
    print(i)
```
Нәтиже:
```
1
3
5
7
9
```
4. Теріс қадам (кему бағыты):
```
for i in range(10, 0, -2):
    print(i)
```
Нәтиже:
```
10
8
6
4
2
```

Ол көбінесе for циклдарында қолданылады.
```
for i in range():
     нұсқаулар
```
Орындалу тәртібі:

і айнымалысына range() функциясының бірінші элементінің мәні меншіктеледі, содан кейін нұсқау орындалады. Содан кейін і

айнымалысына келесі мән ретпен меншіктеледі және range() функциясының барлық элементтері қайталанбайынша солай жалғаса береді.

Үнсіз бастау нөлге және қадам бірге тең.

```
for n in range(10):
    print(n, end = " ")
```

Егер range функциясында бір параметр берілсе, онда ол сандар аралығының ең үлкен мәнін білдіреді. Бұл жағдайда 0-ден 10-ға дейінгі (қоса есептелмеген) реттілік жасалады. Нәтижесінде біз келесі консоль нәтижесін аламыз:

```
0 1 2 3 4 5 6 7 8 9
```

Сондай-ақ, range() функциясына аралықтың ең кіші мәнін беруге болады:

```
for n in range(4, 10):
    print(n, end=" ")
```

Мұнда 4 - тен 10 - ға дейін(қосылмаған) реттілік жасалады.

Нәтиже:

```
4 5 6 7 8 9
```

Сондай-ақ, range() функциясына үшінші параметрді беруге болады, ол қадамды көрсетеді:

```
for n in range(0, 10, 2):
     print(n, end=" ")
```

Мұнда 0 - ден 10 - ға дейінгі реттілік(қосылмаған) 2 қадаммен жасалады.

Нәтиже:

```
0 2 4 6 8
```

Мысал_1

```
for i in range(4)
     print(i)
```

Экранға 0, 1, 2, 3 сандары экранға шығады

Мысал_2

N бүтін санның қосындысын табу

```
n = int(input("N санын енгізіңіз: "))
# N — қанша сан енгізілетінін сұраймыз
total = 0  # Қосындыны сақтау үшін айнымалы
```

```
for i in range(n):
    num = int(input(f"{i+1}-санды енгізіңіз: "))
# Әр санды енгізу
    total += num  # Әр санды қосамыз

print("Сандардың қосындысы:", total)
```

Нәтиже:

```
N санын енгізіңіз: 3
1-санды енгізіңіз: 5
2-санды енгізіңіз: 7
3-санды енгізіңіз: 10
Сандардың қосындысы: 22
```

Мысал_3

Келесі сандар тізбегінің n элементтерінің қосындысын табыңыз:

1 -0.5 0.25 -0.125 ... n

Элементтер саны(n) пернетақтадан енгізіледі. Тізбектің әр мүшесін және оның қосындысын экранға шығарыңыз. for циклдік құрылымын қолданыңыз.

Шешімі:

Бұл жағдайда сандар тізбегі, мұндағы әрбір келесі элемент абсолют мәні бойынша алдыңғысынан екі есе аз және қарама-қарсы таңбаға ие элементтерден тұрады. Бұл келесі элементті алу үшін алдыңғы элементті -2-ге бөлу керек дегенді білдіреді.

Қандай да бір айнымалыға тізбектің бірінші элементінің мәні меншіктелуі керек (бұл жағдайда ол 1). Әрі қарай, циклде оның мәнін қосынды жинақталатын айнымалыға қосыңыз, содан кейін ағымдағы мәнді -2-ге бөліп, тізбектің келесі элементінің мәнін меншіктеңіз. Цикл n рет орындалуы керек.

```
n=int(input('Тізбектің элементтер санын
енгізіңіз:'))
term = 1  # Бірінші мүшесі
sum = 0
print(term)
for i in range(n):
  sum += term
  term/=-2
```

```
  print(term)
print("Қосынды:", sum)
```

Нәтиже:

```
Тізбектің элементтер санын енгізіңіз: 5
1
-0.5
0.25
-0.125
0.0625
-0.03125
Тізбек қосындысы: 0.6875
```

Тапсырмалар

1. X және Y арасындағы барлық бүтін сандарды өсу ретімен шығарыңыз және осы сандардың санын анықтау;
2. Нақты сан - 1 кг кәмпиттің бағасы берілген. 1, 2, ... 10 кг кәмпиттің құнын экранға шығару;
3. Енгізілген жол палиндром болып табыла ма анықтау. Мысалы: ABBA, kazak және т. б.
4. Көбейту кестесін Пифагор кестесі түрінде экранға шығару;
5. N натурал саны берілген. N санындағы ең үлкен цифрды кездестірген сайын көшірмесін жасау. Мысалы, 349291 → 34992991
6. Латин алфавитінің символдарын консольге шығару;
7. Барлық симметриялы төрттаңбалы сандарды экранға шығару. Мысалы: 1221, 7007
8. Цифрларының қосындысы A - ға тең, ал өзі B - ға бөлінетін барлық үштаңбалы сандарды экранға шығару (A және B пернетақтадан енгізіледі);
9. [1; 20] интервалында b=2 қадамымен y= 5x – 2 функция мәндерін экранға шығару;
10. 10 нақты сандар енгізіңдер. Осы сандар тізбегіндегі оң сандардың қосындысын, теріс сандардың және нөлдер санын анықтаңыз.
11. n натурал саны берілген. Есептеу: $1/1^1 + 1/2^2 + ... + 1/n^n$.

12. Цифрларының квадраттарының қосындысы 13 - ке бөлінетін барлық екітаңбалы сандарды экранға шығару;
13. $y=x^3$ функциясын x=6, 5, 4, ..., 1 болғандағы мәндерін анықтау.

Тест тапсырмалары

1. Бағдарлама For Loop сөзін ____ рет шығарады
```
for i in range(5):
    print("For Loop")
```
A. 1
B. 2
C. 4
D. 5
ANSWER: D
2. "Hello" сөзі ____ рет шығарылады
```
for i in range(100):
    print("Hello")
```
A. 10
B. 3
C. 100
D. 99
ANSWER: C
3. range(5) 5 санның тізбегін құрады. Ол құратын тізбекті таңдаңыз
A. 0, 1, 2, 3, 4
B. 1, 2, 3, 4, 5
C. 0, 1, 2, 3, 4, 5
D. 1, 2, 3, 4,78
ANSWER: A
4. Шегіністің дұрыс қолданылуын көрсететін кодты таңда
```
1.  for i in range(3):
        print("Hello")
2.  for i in range(3):
            print("Hello")
3.  for i in range(3):
                print("Hello")
```
A. 1

B. 2
C. 3
D. 2,3
ANSWER: A
5. Бұл код экранға шығарады
```
for i in range(5):
print("Congrats")
```
A. 5
B. Congrats
C. Қате
D. 4
ANSWER: C
6. Код _____ жол шығарады
```
for i in range(5):
    print("Hey!")
```
A. 5
B. 1
C. 4
D. 6
ANSWER: A
7. Бұл код экранға ____ шығарады
```
for i in range(3):
    print(i < 1)
```
A. 3 логикалық өрнек
B. 3 сан
C. 3 әріп
D. 3 нақты сан
ANSWER: A
8. Бұл код ____ сөз шығарады
```
for i in range(3):
    print ("First")
    print ("Second" )
```
A. 3
B. 2
C. 6
D. 4

ANSWER: C
9. range(3) сандардың тізбегі
A. 0, 1, 2
B. 0, 1, 2, 3
C. 1, 2, 3
D. 1, 2, 3, 4
ANSWER: A
10. Кодтың нәтижесі
x=5
print([y//2 for y in range (6)][x])
A. 1
B. 2
C. 3
D. 4
ANSWER: B
11. Кодтың нәтижесі

```
for i in range(10):
    a = i
    if a > 3:
            break
            b = a + 2
            print(b)
```

A. 3
B. 4
C. 5
D. 6
ANSWER: D
12. Кодтың нәтижесі

```
a = 0
for i in range(10):
    if i = = 0:
        a = 13
   else:
       a-=1
print (a)
```

A. 0

B. 10
C. 13
D. 2
ANSWER: C

🕮 3 - бөлім . ТІЗІМДЕР, КОРТЕЖДЕР ЖӘНЕ СӨЗДІКТЕР

Python тілінде тізімдер (list), кортеждер (tuple) және сөздіктер (dictionary) - деректерді сақтау үшін жиі қолданылатын құрылымдық типтер бар.

Тізім(list) - реттелген элементтер жиынын немесе тізбегін сақтайтын деректер типі. Көптеген бағдарламалау тілдерінде массив деп аталатын ұқсас деректер құрылымы бар.

3.1 Тізімдер

Тізім элементтеріне қатынау

Тізім элементтеріне қол жеткізу үшін тізімдегі элементтің нөмірін көрсететін индекстерді пайдалану керек. Индекстер нөлден басталады. Яғни, бірінші элементте 0 индексі болады, екінші элементте 1 индексі болады және т.б. Элементтерге соңынан қол жеткізу үшін -1-ден басталатын теріс индекстерді пайдалануға болады. Яғни, соңғы элементтің индексі -1 болады.

Python тілінде тізімдер құру

Тізімді құрудың бірнеше жолы бар.

Тізімді құру үшін тік жақшалар [] қолданылады, олардың ішінде тізімнің элементтері үтірмен бөлінген.

1. Python кодында бос тізім осылай көрінеді:

```
s = []    # бос тізім
```

2. Мәндері бар тізімдерді құру мысалдары:

```
l = [5, 75, -4, 7, -52]                  # бүтін сандар тізімі
l = [1.13,5.34,12.63,4.6,34.0,12.8] # нақты сандар тізімі
l = [“Назерке”, “Арланбек”, “Жанар”, “Ислам”]
```

жолдар тізімі
3. Тізімдерді «+» белгісі арқылы біріктіруге (конкатенация) болады:
```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
result = list1 + list2
print(result)
```
Нәтиже:
```
[1, 2, 3, 4, 5, 6]
```
4. Тізім элементтерін енгізу үшін for циклін және range() функциясын қолдану:
```
for i in range(N):
    x[i] = int(input())
```
5. Тізімді енгізудің қарапайым нұсқасы:
```
x = [int(input())for i in range(N)]
```
Тізім элементтерін шығару
```
l = [5, 56, 6, 3, 6, 7, 3, -12, 37, -42]
```
1. Бүтін типтегі тізімді шығару:
```
print(l)
```
Нәтиже:
```
[5, 56, 6, 3, 6, 7, 3, -12, 37, -42]
```
2. `l` тізімінің алғашқы 5 элементін бір қатарға шығару:
```
for i in range(5):
    print(l[i], end = " " )
5 56 6 3 6
```
Python - да генераторларды қолданып та тізім құруға болады.
Тізімдер генераторлары
Бірінші тәсіл:
Тізімге бірдей мәндерді көшіру көбейтумен ауыстырылады. Бірлермен толтырылған 10 элементтің тізімі
```
l = [1]*10
print(l)
```
Нәтиже:
```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```
Екінші тәсіл:
```
l = [i for i in range(10)]
print(l)
```

Нәтиже:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Тізім әдістері:

Әдіс	**Сипаттамасы**	**Мысал**
append(x)	Соңына элемент қосады	l.append(5)
insert(i, x)	Белгілі бір (i) орынға x элементін кірістіреді	l.insert(2, 10)
extend(l2)	Басқа тізімнің элементтерін соңына қосады	l.extend([4, 5])
remove(x)	Алғаш кездескен x мәнін өшіреді	l.remove(3)
pop([i])	i индексіндегі элементті алып тастайды (немесе соңғысын)	l.pop()
index(x)	x мәнінің бірінші индексін қайтарады	l.index(7)
count(x)	x элементі неше рет кездесетінін қайтарады	l.count(3)
sort()	Элементтерді өсу ретімен сұрыптайды	l.sort()
reverse()	Тізімді кері аударады	l.reverse()
copy()	Тізімнің көшірмесін қайтарады	l2 = l.copy()
clear()	Барлық элементтерді өшіреді	l.clear()

Мысалы:

```
l = [3, 1, 4, 1, 5]
```

Нәтиже:

```
l.append(9)      # [3, 1, 4, 1, 5, 9]
l.insert(2, 10)  # [3, 1, 10, 4, 1, 5, 9]
l.remove(1)      # [3, 10, 4, 1, 5, 9]
l.sort()         # [1, 3, 4, 5, 9, 10]
l.reverse()      # [10, 9, 5, 4, 3, 1]
```

Python тізімдермен жұмыс істеуді жеңілдету үшін көптеген кірістірілген функцияларды ұсынады. Олар тізімдердің мәндерін өңдеуге, іздеуге, түрлендіруге көмектеседі.

Функция	**Сипаттамасы**	**Мысал**
len(list)	Тізімдегі элементтер санын қайтарады	len([1, 2, 3]) → 3

Функция	Сипаттамасы	Мысал
sum(list)	Барлық сандардың қосындысын қайтарады	sum([1, 2, 3]) → 6
min(list)	Ең кіші элементті қайтарады	min([3, 7, -1]) → -1
max(list)	Ең үлкен элементті қайтарады	max([3, 7, -1]) → 7
sorted(list)	Тізімді сұрыптайды (жаңа тізім қайтарады)	sorted([3, 1, 2]) → [1, 2, 3]
list()	Тізім құру үшін қолданылады	list("abc") → ['a', 'b', 'c']
reversed(list)	Тізімді кері бағытта қайтарады	list(reversed([1, 2, 3])) → [3, 2, 1]

Мысалы:
nums = [3, 5, 7, 2, 9]

	Нәтиже:
print(len(nums))	5
print(sum(nums))	26
print(min(nums))	2
print(max(nums))	9
print(sorted(nums))	[2, 3, 5, 7, 9]

Мысал_1

```
print('Тізімді енгізу. Мысал_1: ')
x=[]
for i in range(4):
    x.append(int(input()))
print()
```

Нәтиже:

```
Тізімді енгізу. Мысал_1:
45
4
85
2
```

```
[45, 4, 85, 2]
```

Мысал_2:

```
x = []
print('Тізімді енгізу. Мысал_2:')
x = [int(input()) for i in range(4)]
print(x)
```

Нәтиже:

```
Тізімді енгізу. Мысал_2:
4
5
7
8
[4, 5, 7, 8]
```

Мысал_3

Тізім генераторын қолданып, тізімді 0 - ден 9 - ға дейінгі сандардың квадраттарымен толтырыңыз.

```
l = [i**2 for i in range(10)]
print(l)
```

Нәтижесі:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Python - да кездейсоқ сандарды генерациялау үшін random модулін пайдалануға болады.

Мысал_4

(10, 80) аралығындағы кездейсоқ 10 саннан құрылған тізім құру:

```
from random import *
# random модулінен барлық функцияларды импорттайды
l = [randint(10, 80) for i in range(10)]
print('(10, 80) аралығындағы кездейсоқ 10 саннан
құрылған тізім:')
print('l = [randint(10, 80) for i in range(10)]:')
print(l)
print()
```

Нәтиже: (кездейсоқ сандар әрқашан өзгеріп тұрады):

```
10,80) аралығындағы кездейсоқ 10 саннан құрылған
тізім
```

```
l = [randint(10,80) for i in range(10)]:
[70, 33, 79, 61, 34, 27, 11, 55, 52, 31]
```

Мысал_5

Тізім әдістерін қолдану

```
a = [0, 2, 2, 2, 4]  # тізім a
b = [5, 6, 7, 2, 9]  # тізім b

print('Бастапқы тізім a: ', a)
print('Бастапқы тізім b: ', b)

x = 99
y = 5

a.append(x)
print('a.append(x): ', a)

a.extend(b)
print('a.extend(b): ', a)

a.insert(3, x)
print('a.insert(3, x): ', a)

a.remove(x)
print('a.remove(x): ', a)

print('a.pop(5): ', a.pop(5))
print(a)

print('a.index(y, 0, len(a)): ', a.index(y, 0,
len(a)))

print('a.count(2): ', a.count(2))

a.reverse()
print('a.reverse(): ', a)

z = a.copy()
```

```
print('z = a.copy(): ', z)

z.clear()
print('z.clear(): ')
print('z = ', z)
```

Бағдарлама нәтижесі:

```
Бастапқы тізім a:  [0, 2, 2, 2, 4]
Бастапқы тізім b:  [5, 6, 7, 2, 9]
a.append(x):  [0, 2, 2, 2, 4, 99]
a.extend(b):  [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.insert(3, x):  [0, 2, 2, 99, 2, 4, 99, 5, 6, 7,
2, 9]
a.remove(x):  [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.pop(5):  99
[0, 2, 2, 2, 4, 5, 6, 7, 2, 9]
a.index(y, 0, len(a)):  5
a.count(2):  3
a.reverse():  [9, 2, 7, 6, 5, 4, 2, 2, 0]
z = a.copy():  [9, 2, 7, 6, 5, 4, 2, 2, 0]
z.clear():
z =  []
```

Тапсырмалар

1. Бүтін сандардан тұратын тізім берілген. a[i] < i шартын қанағаттандыратын элементтерді экранға шығару;
2. A тізімі берілген. Ең үлкен элементті және оның индексін табу;
3. B тізіміндегі 5-ке тең және 5-тен үлкен элементтердің сандарын есептеу;
4. B тізіміндегі барлық оң элементтердің қосындысын және барлық теріс элементтердің көбейтіндісін тауып, экранға шығару;
5. Бүтін сандардан тұратын A тізімі берілген. Енгізілген m бүтін саны осы тізімде қанша рет кездесетінін анықтау;
6. Нақты сандардан тұратын тізім берілген. Теріс сандардың көбейтіндісін және оң сандардың қосындысын тауып, оларды салыстыру;

7. 6 бүтін саннан тұратын тізім берілген. Теріс сандардың қосындысын және санын, ең үлкен элементті табу;
8. 10 нақты саннан тұратын тізім берілген. Тізімдегі оң сандар, теріс сандар және нөлдер санын анықтау;
9. Тізімнің барлық элементтерін кез келген X санына теңестіру;
10. A тізімі берілген. Оң сандардың қосындысын және санын табу;
11. Бүтін сандардан тұратын тізім берілген. [c, d] кесіндісі аралығындағы тізім элементтерін экранға шығару;
12. A тізімі берілген. Осы тізімдегі реттік номерлері жұп болатын элементтердің ішінен ең кішісін анықтау;
13. A тізімінің орта мәнінен үлкен A тізімінің элементтерінен құралған B тізімін құрыңыз.

3.2 Тізімдер тізімі

Тізімдер жолдар мен сандар сияқты стандартты деректерден басқа тізімдерді де қамтуы мүмкін, яғни бір тізімнің ішінде басқа тізімдер сақталады. Бұл күрделі деректер құрылымдарын құруға мүмкіндік береді, мысалы, кестелер, матрицалар немесе әртүрлі топтағы элементтер.

Көп есептерде деректерді кестелерге сақтауға тура келеді. Бұл кестелер екі өлшемді тізімдер деп аталады.

Екі өлшемді тізім жолдар мен бағаналардан тұрады. Кірістірілген тізімнің элементіне қол жеткізу үшін екі индекс қолдану қажет: Екі өлшемді тізімнің жеке элементіне қатынау A[i][j]

i - жолдар нөмері, 0<=i<n , ал j - бағаналар нөмері, 0<=j<m.

Мысал 1

n жолдар мен n бағандардан тұратын квадрат тізім берілген. Сол жақ жоғарғы бұрыштан оң жақ төменгі бұрышқа өтетін негізгі диагональдағы элементтерге (яғни i==j болатын A[i][j] элементтері) 1 мәнін, негізгі диагональдан жоғары элементтерге 0 мәнін, негізгі диагональдан төмен элементтерге 2 мәнін меншіктеу бағдарламасын құрайық.

```
# n x n өлшемді матрицаны құру
n = 4
```

```
A = []  # Бос тізім (матрица)

# Матрица элементтерін толтыру
for i in range(n):
    row = []  # Жаңа жол (жол тізімі)
    for j in range(n):
        if i == j:
            row.append(1)  # Негізгі диагональ
        elif i < j:
            row.append(0)  # Диагональдан жоғары
        else:
            row.append(2)  # Диагональдан төмен
    A.append(row)  # Жолды матрицаға қосу

# Матрицаны экранға шығару
for row in A:
    print(row)
```

Нәтиже:

[1, 0, 0, 0]
[2, 1, 0, 0]
[2, 2, 1, 0]
[2, 2, 2, 1]

 **Тапсырмалар**

1. А матрицасы берілген. Бағаналарды сәйкес жолдармен ауыстыру;
2. Матрицаның оң элементтерінің санын табу;
3. A(n, m) матрицасы берілген. Барлық мәндерінің арифметикалық ортасын табу;
4. Пифагор кестесін экранға шығару. Пифагор кестесі – элементтері A[i,j]=i*j формуласымен анықталатын 10x10 квадраттық кесте.
5. Матрицадағы ең үлкен және ең кіші элементті тауып, олардың орындарын ауыстыру;
6. Матрицаның екінші қатарындағы теріс элементтердің қосындысын табу;
7. Матрицаның ең кіші элементі мен оның реттік номерін шығару;

8. 10 х 10 матрицасы берілген. А матрицасын мына ретпен толтыру:

0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0
0	0	0	3	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0
0	0	0	0	0	5	0	0	0	0
0	0	0	0	0	0	6	0	0	0
0	0	0	0	0	0	0	7	0	0
0	0	0	0	0	0	0	0	8	0
0	0	0	0	0	0	0	0	0	9

9. A(n, m) бас диагоналінен жоғары орналасқан, жұп және оң элементтердің көбейтіндісін және қосындысын табу;
10. A(n, m) матрицасы берілген. Тақ нөмірлі бағаналардың элементтерінің арифметикалық ортасын табу;
11. A(n, m) матрицасы берілген. Осы матрицаның индекстерінің қосындысын тақ болатын элементтерді экранға шығару;
12. A(n, m) матрицасы берілген. Негізгі диоганаль элементтерінің қосындысын табу;
13. A(n*n) матрицасы берілген. индекстерінің айырмасы i-j=k болатын элементтердің қосындысын табу. k берілген бүтін сан.

Тест тапсырмалары

1. Код нәтижесі
A = [[ ]] * 3
A[0].append(3)
print(A)
A. [[3], [], []]
B. [[3], [3], [3]]]
C. [3, [], []]
D. [[3], [3], [3], 3]
ANSWER: B
2. Тізімдегі мәндер саны
cart = [“milk”, “eggs”, “apples”]

A. 1
B. 2
C. 3
D. 4
ANSWER: C
3. Тізімнің атауы
games = [“Snake”, “Puzzle”, “Chess”]
A. Snake
B. Puzzle
C. Chess
D. games
ANSWER: D
4. Тізімдегі мәндер типі
prices = [0.59, 2.99, 14.5]
A. Нақты
B. Логикалық
C. Бүтін
D. Символдық
ANSWER: A
5. Тізім - бұл мәндер тізбегі. Тізімдегі әрбір мәннің ... болады
A. Циклі
B. Түсі
C. Индексі
D. Биіктігі
ANSWER: C
6. Тізім индексі _ - ден басталады
A. -1
B. 1
C. 0
D. 4
ANSWER: C
7. Тізімде мәндер _ бөлінуі тиіс
A. Жақшалармен
B. Үтірлермен
C. Нүктелі үтірлермен
D. Нүктелермен

ANSWER: B
8. Код нәтижесі
animals = ["cat", "dog", "bird"]
print(animals[2])
A. cat
B. bird
C. dogcat
D. dog
ANSWER: C
9. Тізімдегі үшінші элементтің индексі
songs = ["Trip", "Perfect", "Yesterday"]1
A. 2
B. 3
C. 0
D. 4
ANSWER: B
10. Код нәтижесі
products = ["apples", "oranges", "bananas"]
products[2] = "lime"
print(products[2])
A. lime
B. oranges
C. products
D. apples
ANSWER: A
11. Код нәтижесі
words = ["rise", "sun", "glasses"]
print(words[1] + words[0]
A. sunglasses
B. sun
C. sunrise
ANSWER: C
12. Тізімді сақталатын мәндер
A. Тек жолдар
B. Тек бүтін сандар
C. Деректердің кез келген типі

D. Тек нақты сандар
ANSWER: C
13. Кодтың нәтижесі
num = [0, 1, - 1, 99, 7, 5]
num.sort)
print(num[-1])
A. 0
B. -1
C. 99
D. 7
ANSWER: C
14. numbers = [1, 3, 5, 6]
for i in numbers:
    for c in range(i):
        print(c)
___рет шығарылады
A. 5
B. 4
C. 3
D. 0
ANSWER:B
15. Кодтың нәтижесі
test=[]
for i in range(2):
  test.append (100)
  test.append (3)
print(test[1])
A. 5
B. 4
C. 3
D. 0
ANSWER: C
16. Кодтың нәтижесі
myList = [2, 3, 6]
myList.append (5, 4)
print(myList[3])

A. 6
B. 4
C. 3
D. Error
ANSWER: D
17. Тізімнің ұзындығын қайтаратын функция
A. len()
B. items()
C. count()
D. nums()
ANSWER: A
18. "boat" мәнінің индексі
items = ["bike", "car", "boat", "scooter"]
A. 1
B. 2
C. 3
D. 4
ANSWER: B
19. Кодтың нәтижесі
a = [[]] * 2
a.append (2)
print (a)
A. [2]
B. [[] 2]
C. [2, []]
D. [[], [], 2]
ANSWER: D
20. Код нәтижесі:
movies = ["Avatar", "Titanic", "Avengers"]
movies.append ("Alien")
print (movies [3])
A. Avatar
B. Titanic
C. Avengers
D. Alien
ANSWER: D

21. Код нәтижесі
topic = ["Maths", "English", "Physics"]
topic.pop(2)
print (topic)
A. ["Maths", "Physics"]
B. ["Maths", "English", "Physics"]
C. ["Maths", "English"].
D. ["Physics ", "English"].
ANSWER:C
22. Movies деп аталатын тізім үшін len функциясын шақырудың дұрыс әдісін таңдаңыз
A. movies.len()
B. len(movies)
C. len.movies()
D. len()
ANSWER: B

3.3 Кортеждер

Кортеж(Tuple) - Python-дағы өзгермейтін деректер құрылымы. Кортеждер бірнеше элементтен тұратын жинақтарды сақтауға арналған және оларды жай жақшалар( ) арқылы анықтайды.
Кортеждің негізгі ерекшеліктері:
1. Өзгермейтін: Бір рет жасалған кортеждегі элементтерді өзгертуге болмайды;
2. Индекс арқылы қолжетімді: Кортеждің элементтерін индекстермен шақыруға болады;
3. Қайталанатын элементтерді сақтауға болады;
4. Түрі мен ұзындығы бойынша өзгермейді;

Мысал 1:
```
my_tuple = (1, 2, 3, 4, 5)

print(my_tuple)
```
Нәтиже:
```
(1, 2, 3, 4, 5)
```
Мысал 2:

Кортеждің элементтеріне индексі арқылы қол жеткізу:

```
my_tuple = (10, 20, 30, 40, 50)

print(f"Бірінші элемент: {my_tuple[0]}")

# Соңғы элемент
print(f"Соңғы элемент: my_tuple[-1])
```

Нәтиже:

```
Бірінші элемент: 10
Соңғы элемент: 50
```

Кортеждің ерекшеліктері:

1. Өзгермейді:

```
my_tuple = (1, 2, 3)
my_tuple[0] = 10
```

Нәтиже:

Қате болады, кортеж өзгермейді

```
TypeError: 'tuple' object does not support item
assignment
```

2. Әртүрлі типтерді сақтай алады:

```
mixed_tuple = (1, "Hello", 3.14, [1, 2, 3])
print(mixed_tuple)
```

Нәтиже:

```
(1, 'Hello', 3.14, [1, 2, 3])
```

Кортеждің ұзындығын табу:

```
my_tuple = (1, 2, 3, 4)
print(f"Кортеж ұзындығы: {len(my_tuple)}")
```

Нәтиже:

```
Кортеж ұзындығы: 4
```

Кортеждерді біріктіру:

```
tuple1 = (1, 2)
tuple2 = (3, 4)
combined_tuple = tuple1 + tuple2
print(f"Қосылған кортеж: {combined_tuple}")
```

Нәтиже:

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

Кортежді жою:

```
my_tuple = (1, 2, 3)
del my_tuple
# Кортежді жою
# print(my_tuple)  # Қате болады, себебі ол жойылған
```

Тапсырмалар

1. Көп өлшемді кортежден мәліметтерді шығару:
Келесі құрылым берілген:

```
data = (
    ("Alice", 30, ("Python", "C++")),
    ("Bob", 25, ("Java", "Go")),
    ("Carol", 28, ("JavaScript", "Python"))
)
```

Тапсырма:

- "Python" тілін білетін адамдардың аттарын шығару.
- Орташа жасты есептеу.

2. Кортежді топтап өңдеу
Берілген тізім:

```
data = [("A", 5), ("B", 7), ("A", 3), ("C", 8), ("B", 2)]
```

Тапсырма:
Әрбір әріпке қатысты сандардың қосындысын есептеп, мынандай форматта шығару:

```
[("A", 8), ("B", 9), ("C", 8)]
```

3. Кортеж ішіндегі кортеждерді сұрыптау
Берілген:

```
data = [("user1", 3), ("user2", 1), ("user3", 5)]
```

Тапсырма:
Екінші мән (санау) бойынша кему ретімен сұрыптау.
4. Кортеждерден уникалды мәндер жинау:
Берілген:

```
users = (
    ("Ali", "Python"),
    ("Sara", "Java"),
    ("Ali", "C++"),
```

```
    ("Sara", "Python"),
    ("Dina", "Python")
)
```

Тапсырма: Әрбір адамның қайталанбайтын(уникалды) бағдарламалау тілдерін анықтау;

Нәтиже:

```
{
    "Ali": {"Python", "C++"},
    "Sara": {"Java", "Python"},
    "Dina": {"Python"}
}
```

5. Кортеж элементтерін динамикалық өңдеу:

Пайдаланушыдан N кортеж мәнін енгізу сұраңыз (мысалы, (name, age)), содан кейін:

- Жасы 30-дан асқандарды шығару;
- Ең жас адамның есімін көрсету.

6. Immutable vs Mutable

Тапсырма:

- Кортеж ішіндегі тізімді (list) өзгерту.
- Кортеждің жалпы өзгермейтін қасиетін сақтай отырып, ішіндегі мәндерді қалай өзгертуге болатынын дәлелдеу.

7. Кортежден CSV жасау:

Тапсырма:

Келесі кортеж:

```
users = (("id", "name", "age"), (1, "Ali", 25), (2, "Dana", 30))
```

оны CSV жолына түрлендіру:

```
id,name,age
1,Ali,25
2,Dana,30
```

8. Бірегей пайдаланушыларды санау:

Берілген мәлімет:

```
logins = (
    ("user1", "2024-04-01"),
    ("user2", "2024-04-01"),
    ("user1", "2024-04-02"),
    ("user3", "2024-04-03"),
```

("user2", "2024-04-04"),
)
Тапсырма:
- Әрбір күнге бірегей (уникалды) қолданушы санын анықтау.

Нәтиже:
```
{
  "2024-04-01": 2,
  "2024-04-02": 1,
  "2024-04-03": 1,
  "2024-04-04": 1
}
```
9. Мәндерге қарай топтау:
```
data = (
  ("A", 100),
  ("B", 200),
  ("A", 150),
  ("C", 50),
  ("B", 300),
)
```
Тапсырма: Әрбір әріпке қатысты барлық мәндердің тізімін жасау:
Нәтиже:
```
{
  "A": [100, 150],
  "B": [200, 300],
  "C": [50]
}
```
10. Тек сандардан тұратын кортежді тексеру:
Берілген:
```
t = (1, 2.5, "a", 3)
```
Тапсырма:
- Кортеж тек сандардан тұрады ма?
- Егер иә болса, орта мәнін есептеу.

11. Кортеж ішінен ең жиі кездесетін мәнді табу;
Берілген:
```
t = (4, 2, 7, 4, 3, 4, 2, 7, 7, 7)
```
Тапсырма:

– Кортеждегі ең жиі кездесетін мәнді және оның қанша рет кездескенін анықтау.

Нәтиже: 7 кездескен саны - 4 рет

12. Кортеждің ең үлкен элементін және оның индексін анықтау;
13. Екі кортежді қосып, нәтижесінде шыққан кортежді сұрыптап, оны шығару.

3.4 Сөздіктер

Сөздік (dictionary) - бұл Python тіліндегі элементтер жинағы, мұнда әрбір элементтің бірегей кілті болады және сол кілтпен байланысты мән сақталады.

Синтаксисі:

dictionary = {кілт1: мән1, кілт2: мән2, ...}

Жүйелі жақшада ({}) үтір арқылы бірнеше элемент көрсетіледі, мұнда әрбір элемент: алдымен кілт(key), содан кейін қос нүктеден кейін оның мәні (value) жазылады.

Мысал 1:

```
users = {1: "Tom", 2: "Bob", 3: "Bill"}
```

Мұндағы users сөздігінде кілттер ретінде сандар, ал мәндер ретінде жолдар (мәтіндер) қолданылған.

Мысал 2

```
emails = {"tom@gmail.com": "Tom", "bob@gmai.com": "Bob", "sam@gmail.com": "Sam"}
```

Мұнда emails сөздігінде кілттер - электрондық пошта мекен - жайлары, ал мәндер - қолданушы есімдері.

Кілттер мен мәндер әр түрлі типте де бола алады:

```
objects = {1: "Tom", "2": True, 3: 100.6}
```

Python - да кілттер мен мәндердің типтері әртүрлі болуы мүмкін.

Бос сөздік:

```
objects = {}
```

немесе

```
objects = dict()
```

Тізімдер мен кортеждерді сөздікке түрлендіру

Сөздік пен тізім - құрылым жағынан әртүрлі болғанымен, кейбір арнайы түрдегі тізімдерді сөздікке айналдыруға болады. Бұл үшін Python-дағы dict() деген кірістірілген функция қолданылады.
Түрлендіру үшін тізімнің ішінде ішкі тізімдер (немесе кортеждер) болуы керек.
Әр ішкі тізім (немесе кортеж) екі элементтен тұруы тиіс:
- Біріншісі - кілт(key),
- Екіншісі - мән(value).

Мысал: тізімнен сөздік жасау

```
users_list = [
    ["+111123455", "Tom"],
    ["+384767557", "Bob"],
    ["+958758767", "Alice"]
]

users_dict = dict(users_list)
print(users_dict)
```

Нәтиже:

```
{"+111123455":    "Tom",    "+384767557":    "Bob",
"+958758767": "Alice"}
```

Кортежді сөздікке түрлендіру

Кортеждермен де тура осылай жасауға болады:

```
users_tuple = (
    ("+111123455", "Tom"),
    ("+384767557", "Bob"),
    ("+958758767", "Alice")
)

users_dict = dict(users_tuple)
print(users_dict)
```

Бұл да дәл сондай нәтиже береді.

Сөздік элементтерін алу және өзгерту

Сөздіктегі элементке кіру үшін сөздіктің атымен бірге квадрат жақшада кілт жазылады:

dictionary[кілт]
Мысал:
```
users = {
    "+11111111": "Tom",
    "+33333333": "Bob",
    "+55555555": "Alice"
}
# "+11111111" кілті бойынша мән алу
print(users["+11111111"])
```
Нәтиже:
```
Tom
```
```
# "+33333333" кілтінің мәнін өзгерту
users["+33333333"] = "Bob Smith"
print(users["+33333333"])
```
Нәтиже:
```
Bob Smith
```
Егер кілт жаңа болса - жаңа элемент қосылады:
```
users["+4444444"] = "Sam"
```
Бұл жолы "+4444444" деген жаңа кілт пайда болып, оған "Sam" деген мән меншіктеледі.

🕮 4 - бөлім. ЖОЛДАРМЕН ЖҰМЫС

4.1 Жолдарды енгізу және жол таңбаларына қатынау

Python тіліндегі жол (string) - тырнақшаға алынған Юникод (Unicode) таңбаларының тізбегі болып табылады. Жолды бір ('), қос ("), немесе үш (''' немесе """) тырнақша арқылы құруға болады.

Жолдарды стандартты түрде input() функциясы арқылы енгізуге болады. Ал жолдың ұзындығын (яғни ішіндегі таңбалар санын) len() функциясы арқылы анықтаймыз.

Python тіліндегі кез келген басқа нысанды (мысалы: сан, тізім, логикалық мән) жолға түрлендіруге болады. Бұл үшін str() функциясы

қолданылады. Яғни, түрлендірілетін нысанды параметр ретінде беріп, str() функциясын шақыру жеткілікті.
Мысалы:
```
message = "Hello World!"
print(message)
```
Нәтижесі:
```
Hello World!
```
Егер біз көп жолды мәтінді анықтағымыз келсе, онда мұндай мәтін үш қос немесе жалғыз тырнақшаға алынады:
```
'''
Бұл түсініктеме
'''
text = '''егер үштік жалғыз тырнақша ішіндегі мәтін айнымалыға меншіктелсе, онда ол түсініктеме емес, жол болып табылады.
'''
print(text)
```
Үштік жалғыз тырнақшаларды пайдаланған кезде, оларды түсініктемелермен шатастырмау керек: егер үштік жалғыз тырнақша ішіндегі мәтін айнымалыға меншіктелсе, онда ол түсініктеме емес, жол болып табылады.
\n: жаңа жолға ауыстырады
Мысалы:
```
text = "Message:\n\"Hello World\""
print(text)
```
Нәтижесі:
```
Message:
"Hello World"
```

Жолға мәндерді қою

Python басқа айнымалылардың мәндерін жолға ендіруге мүмкіндік береді. Ол үшін жолдың ішіндегі айнымалылар {} жақшаға орналастырылады, ал f символы бүкіл жолдың алдына қойылады:
```
userName = "Tom"
userAge = 37
user = f"name: {userName}  age: {userAge}"
```

```
print(user)
```
Нәтижесі:
```
name: Tom  age: 37
```
Бұл жағдайда {userName} орнына userName айнымалысының мәні қойылады. Сол сияқты, {userAge} орнына userAge айнымалысының мәні қойылады.

Жол таңбаларына қатынау

Жолдың жеке таңбаларына тік жақшалардағы индекс бойынша қол жеткізе аламыз:

Мысалы:
```
string = "hello world"
c0 = string[0]
print(c0)
```
Нәтиже:
```
h
c6 = string[6]
print(c6)
```
Нәтиже:

w
```
c11 = string[11]
print(c11)
```
Нәтиже:
```
қате IndexError: string index out of range
```
Индекстеу нөлден басталады, сондықтан жолдың бірінші таңбасы 0 индексіне ие болады. Ал жолда жоқ индекске қол жеткізуге тырыссақ, IndexError ерекше жағдайы орын алады. Мысалы, жоғарыдағы жағдайда жолдың ұзындығы 11 таңба, сондықтан оның таңбаларында 0-ден 10-ға дейінгі индекстер болады.

Жолдың соңынан басталатын таңбаларға қол жеткізу үшін теріс индекстерді пайдалануға болады. Сонымен, -1 индексі соңғы таңбаны, ал -2 соңынан екінші таңбаны білдіреді және т.б.:
```
string = "hello world"
c1 = string[-1]
print(c1)
```
Нәтиже:
```
d
```

```
c5 = string[-5]
print(c5)
```
Нәтиже:
```
W
```
Жолдағы таңбаларды шығару

Жолдағы барлық таңбаларды қайталау үшін for циклін пайдалануға болады:

Мысалы:
```
string = "hello world"
for char in string:
    print(char)
```
Нәтиже:
```
h
e
l
l
o

w
o
r
l
d
```
Ішкі жолды алу

Қажет болса, жолдан жеке таңбаларды ғана емес, ішкі жолды да ала аламыз. Ол үшін келесі синтаксис қолданылады:

string[:end]: 0 индексінен end(қосылмаған) индексіне дейінгі таңбалар тізбегін шығарады.

string[start:end]: таңбалар тізбегін start индекстен бастап end(қосылмаған) индекстің соңына дейін шығарады.

string[start:end:step]: step қадамы арқылы start бастапқы индекстен end (қосылмаған) соңғы индекске дейінгі таңбалар тізбегін шығарады.

Біз ішкі жолды алу үшін барлық опцияларды қолданамыз:
```
string = "hello world"
```

0 индекінен 5 индексіне дейін
```
sub_string1 = string[:5]
print(sub_string1)
```
Нәтиже:
```
hello
```

2 - ші индекстен 5 - ші индексе дейін
```
sub_string2 = string[2:5]
print(sub_string2)
```
Нәтиже:
```
llo
```

#2 - ші индекстен 9 - шы индексе дейін бір бос орын арқылы
```
sub_string3 = string[2:9:2]
print(sub_string3)
```
Нәтиже:
```
lowr
```
Жолдарды біріктіру

Жолдармен кең таралған амалдардың бірі олардың бірігуі немесе конкатенация болып табылады. Жолдарды біріктіру үшін қосу амалы қолданылады:

Мысалы:
```
name = "Tom"
surname = "Smith"
fullname = name + " " + surname
print(fullname)
```
Нәтиже:
```
Tom Smith
```
Екі жолды біріктіру оңай, бірақ жол мен санды қосу керек болған жағдайда str() функциясын қолданып, санды жолға айналдыру керек.

Мысалы:
```
name = "Tom"
Age = 33
info = "Name: " + name + " Age: " + str(age)
print(info)
```

Нәтиже:
```
Name: Tom Age: 33
```

4.2 Жолдарға қолданылатын функциялар

Жолды қайталау

Жолды бірнеше рет қайталау үшін көбейту амалы қолданылады:

```
print("a" * 3)
```
Нәтиже:
```
aaa
```
```
print("he" * 4)
```
Нәтиже:
```
hehehehe
```

Lower() функциясы жолды кіші әріпке түрлендіреді, ал upper() функциясы жолды бас әріпке түрлендіреді.

Мысалы:
```
# Жолды кіші әріпке түрлендіру
text1 = "Hello World"
lower_text = text1.lower()
print("Кіші әріппен:", lower_text)

# Жолды бас әріпке түрлендіру
text2 = "hello world"
upper_text = text2.upper()
print("Бас әріппен:", upper_text)
```
Нәтиже:
```
Кіші әріппен: hello world
Бас әріппен: HELLO WORLD
```
Мысалы:
```
str1 = "Tom"
str2 = "tom"
print(str1 == str2)
```
Нәтиже:
```
False
# жолдар тең емес
```

```
print(str1.lower() == str2.lower())
```
Нәтиже:
```
True
```
ord және len функциялары

Жолда Юникод таңбалары болғандықтан, Юникод кодтауындағы таңбаның сандық мәнін алу үшін ord() функциясын пайдалана аламыз:
```
print(ord("A"))
```
Нәтиже:
```
65
```
Жолдың ұзындығын алу үшін len() функциясы қолданылады:
```
string = "hello world"
length = len(string)
print(length)
```
Нәтиже:
```
11
```

4.3 Жолдарға қолданылатын негізгі әдістер

Қосымшаларда қолдана алатын жолдың негізгі әдістерін қарастырайық:

Жолды іздеу

Python-да жолда ішкі жолды табу үшін, жолдағы ішкі жолдың бірінші индексін қайтаратын және find() әдісін қолданамыз:

find(str): жолдың басынан аяғына дейін str ішкі жолын іздейді

find(str, start): start параметрі іздеу керек бастапқы индексті көрсетеді.

find(str, start, end): end параметрі іздеу баратын соңғы индексті көрсетеді.

Ішкі жол табылмаса, әдіс -1 қайтарады:
```
welcome = "Hello world! Goodbye world!"
index = welcome.find("wor")
print(index)
```
Нәтиже:
```
6
```

```
# 10-шы индекстен бастап іздеу
index = welcome.find("wor",10)
print(index)
```
Нәтиже:
```
21
```

```
# 10 шы индекстен 15 ші индексе дейін
index = welcome.find("wor",10,15)
print(index)
```
Нәтиже:
```
-1
```
Жолды ауыстыру

Жолдағы бір ішкі жолды екіншісімен ауыстыру үшін replace() әдісі қолданылады:

`replace(old, new)`: old ішкі жолды new жаңаға ауыстырады.

`replace(old, new, num)`: num параметрі old ішкі жолдың қанша рет new ге ауыстырылуы керектігін көрсетеді. Әдепкі бойынша num -1, ол әдістің бірінші нұсқасына сәйкес келеді және барлық оқиғалардың ауыстырылуын тудырады.

```
     phone = "+7-747-567-89-10"
# дефистерді бос орынмен
edited_phone = phone.replace("-", " ")
print(edited_phone)
```
Нәтиже:
```
+7 747 567 89 10
```

```
# дефистерді өшіру
edited_phone = phone.replace("-", "")
print(edited_phone)
```
Нәтиже:
```
# +77475678910
```

```
# бірінші дефисті ауыстыру
edited_phone = phone.replace("-", "", 1)
print(edited_phone)      # +7747-567-89-10
```

Жолдарды біріктіру

Жолдармен ең қарапайым амалдарды қарастырғанда, қосу амалының көмегімен жолдарды біріктіру жолы көрсетілді. Жолдарды біріктірудің тағы бір мүмкіндігі - join() әдісі: ол жолдар тізімін біріктіреді. Сонымен қатар, бұл әдіс шақырылатын ағымдағы жол бөлгіш ретінде пайдаланылады:

```
words = ["Let", "me", "speak", "from", "my", "heart", "in", "English"]

# бөлгіш – бос орын
sentence = " ".join(words)
print(sentence)
```

Нәтиже:

```
Let me speak from my heart in English
```

```
# бөлгіш – бос орын - тік сызық
sentence = " | ".join(words)
print(sentence)
```

Нәтиже:

```
Let | me | speak | from | my | heart | in | English
```

Тапсырмалар

1. Кері жол. Пайдаланушыдан жолды енгізуді сұрап, оны кері ретпен басып шығаратын бағдарлама жазу;
2. Таңбаларды санау. Пайдаланушыдан жолды қабылдайтын және сол жолдағы символдар санын басып шығаратын бағдарлама жазу;
3. Дауысты және дауыссыз дыбыстарды санау. Қолданушыдан жолды қабылдайтын, сол жолдағы дауысты және дауыссыз дыбыстардың санын есептейтін бағдарлама жазу;
4. Полиндром. Енгізілген жолдың полиндром екенін тексеретін бағдарлама жазу(бұл алға да, артқа да бірдей оқылатын сөз, сөз тіркесі, сан немесе басқа да символдар тізбегі. Мысалы, "қазақ", "тарат", "12321" - бұлар полиндромдар).

5. Ішкі жолды ауыстыру. Пайдаланушыдан жолды және екі ішкі жолды қабылдайтын бағдарламаны жазу. Содан кейін кездесетін бірінші ішкі жолдың барлығын екінші ішкі жолға ауыстыру және нәтижені басып шығару;
6. Сөзді іздеу. Пайдаланушыдан жолды және сөзді қабылдайтын, содан кейін жолда сол сөз бар-жоғын тексеретін бағдарламаны жазу;
7. Ішкі жолды шығару. Пайдаланушыдан жолды және екі индексті қабылдайтын бағдарламаны жазу. Содан кейін осы индекстер арасындағы ішкі жолды шығарып, нәтижені басып шығару;
8. Бас және кіші әріптерге түрлендіру. Жолды қабылдайтын және барлық әріптерді бас және кіші әріптерге түрлендіретін, содан кейін нәтижені басып шығаратын бағдарлама жазу;
9. Жолды бөлу. Пайдаланушыдан жолды және бөлгіш таңбаны қабылдайтын, содан кейін бөлгіш символды пайдаланып жолды ішкі жолдарға бөлетін бағдарламаны жазу;.
10. Артық бос орындарды жою. Жолды қабылдайтын және жолдың басы мен соңындағы барлық артық бос орындарды алып тастайтын және жол ішіндегі бос орындар тізбегін бір бос орынға ауыстыратын бағдарламаны жазу;
11. 2 жолды қабылдайтын және subst ішкі жолдың st жолында бар-жоғын анықтайтын search_substr(subst, st) функциясын жазу. Егер ішкі жол табылса, «Байланыс бар!» деген тіркес қайтару, әйтпесе «Өткен!» Сәйкестік екі жолдың регистріне қарамастан табылуы керек. Тапсырманы шешу үшін lower() және find() жолдық әдістерін қолдану керек. Қажетті элемент табылмаса, find() функциясы -1 мәнін қайтаратынын есте ұстаған жөн.
12. Жолдағы бірінші және соңғы әріптің индекстерін анықтау керек. Ол үшін 2 параметрді қамтитын first_last(letter, st) функциясын жазу: letter – ізделетін символ, st – жол. Жолда әріп болмаса, кортежді қайтару керек(None, None), егер бар болса, онда кортеж осы таңбаның бірінші және соңғы индексінен тұрады; Тапсырмада find() және rfind() әдістерін қолдану қажет.
13. Берілген мәтін үзіндісі негізінде ондағы ең жиі кездесетін 3 таңбаны анықтау. Бос орындарды елемеу керек (есептеу кезінде ескерілмейді). Есептеулер нәтижелерін экранға шығару үшін top3(st) функциясын жазу. Функцияның нәтижесі жол түрінде

берілген: «таңба – рет саны, символ – рет саны...». Таңбалардың санын оңай санау үшін collections модуліндегі Counter ді пайдалану ыңғайлы.

Тест тапсырмалары

1. Find () функциясы жолға қолданылады, сондықтан ол нүктелік белгіні қолдану арқылы шақырылады. Функция қайтаратын мән
"robot".find ("o")
A. 1
B. 2
C. 3
D. 4
ANSWER: A
2. Егер мән жолдан табылмаса код нәтижесі
print ("robot". find ("A"))
A. 1
B. 2
C. 0
D. -1
ANSWER: D
3. Код нәтижесі
movie = "Avatar"
print(len(movie))
A. 5
B. 1
C. 4
D. 6
ANSWER: D
4. Код нәтижесі
str = '131231323568132513'
print(str1.count ("13"))
A. 5
B. error
C. 4
D. 10

ANSWER: B
5. Код нәтижесі
print ("adcb"[: -1])
A. abcd
B. bcda
C. cbda
D. adbc
ANSWER: B
6. Шығарылатын таңба
animals = “Dog”
print(animals[0])
A. D
B. o
C. g
D. A
ANSWER: A
7. Шығарылатын таңба
notification = “New message!”
print(notification [4])
A. m
B. w
C. space
D. c
ANSWER: A
8. Шығарылатын таңба characters = “!#- ?”
print(characters [4])
A. -
B. s
C. ?
D. #
ANSWER: C
9. Код нәтижесі
x = "arctic"
print(x[2] + x[0] + x[3])
A. cat
B. arc

C. tic
D. ar
ANSWER: A
10. Код нәтижесі
vehicle = “motorbike”
print(vehicle[:5])
A. motor
B. mot
C. bike
D. tor
ANSWER: A
11. Код нәтижесі:
vehicle = “motorbike”
print(vehicle[:])
A. motor
B. bike
C. motorbike
D. tor
ANSWER: C
12. Код нәтижесі
num = “45”
print(int(num) + 3)
A. 453
B. 53
C. 48
D. 3
ANSWER: C

🕮 5 - бөлім. ФУНКЦИЯ

5.1 Функцияны сипаттау және оны шақыру

Функциялар - белгілі бір тапсырманы орындайтын және бағдарламаның басқа бөліктерінде қайта пайдалануға болатын код блогы.

Python тілінде көптеген кіріктірілген (встроенные) функциялар бар

және сонымен қатар өзіңіздің жеке функцияларыңызды да анықтауға болады.

Мысалы, print() - консольге (экранға) мәнді шығару үшін қолданылатын кірістірілген функция болып табылады.

Функцияның синтаксисі:

```
def функция_атауы ([параметрлер]):
    нұсқаулар
```

Функция сипаттамасы функция атауынан, параметрлері бар жақшалар жиынынан және қос нүктеден тұратын def өрнегінен басталады. Жақшадағы параметрлер міндетті емес. Ал келесі жолдан функцияның командалар блогы орындалады. Барлық функция нұсқаулары жолдың басынан шегінеді. Функциялар параметрлі және параметрсіз болады.

Функцияны шақыру үшін функцияның атын көрсету, содан кейін жақшаның ішінде оның барлық параметрлері үшін мәндерді беру керек.

Функцияны шақыру:

```
функция_атауы([параметрлер])
```

Мысалы, ең қарапайым параметрсіз функцияны анықтап және шақырайық:

```
def say_hello():      # say_hello функциясын анықтау
    print("Hello")

say_hello()           # say_hello функциясын шақыру
say_hello()
say_hello()
```

Мұнда say_hello функциясы қатарынан үш рет шақырылады, нәтижесінде консольдегі нәтиже:

```
Hello
Hello
Hello
```

Функция параметрлері

Функция параметрлерді қабылдай алады. Функцияға деректерді параметрлер арқылы беруге болады. Мысалы, `print()`

функциясы, ол параметрді пайдалана отырып, консольге мәндерді шығарады.
Енді функцияны параметрлермен анықтап, қолданайық:
```
def say_hello(name):
    print(f"Hello, {name}")

say_hello("Tom")
say_hello("Bob")
say_hello("Alice")
```
say_hello функциясының name параметрі бар және функцияны шақырған кезде біз осы параметрге қандай да бір мәнді бере аламыз. Функцияның ішінде біз параметрді айнымалы ретінде пайдалана аламыз, мысалы, басып шығару функциясын пайдаланып осы параметрдің мәнін консольге басып шығарамыз.
әтиже:
```
Hello, Tom
Hello, Bob
Hello, Alice
```
Параметрлердің анықталмаған саны

Жұлдызша таңбасын мәндердің белгісіз санын беруге болатын параметрді анықтау үшін пайдалануға болады. Бұл функция бірнеше мәндерді алуды қалаған кезде пайдалы, бірақ біз нақты қанша мәнді білмейміз. Мысалы, сандардың қосындысын есептейтін функцияны анықтайық:
```
def sum(*numbers):
    result = 0
    for n in numbers:
        result += n
    print(f"sum = {result}")

sum(1, 2, 3, 4, 5)
sum(3, 4, 5, 6)
```
Нәтиже:
```
sum = 15
sum = 18
```

5.2 return операторы және функциядан нәтижені қайтару

Функция нәтижені қайтара алады. Функция мәнді қайтару үшін return операторын қолданады:

```
def функция_атауы ([параметрлер]):
    нұсқаулар
    return қайтарылатын_мән
```

Мысалы, сандардың қосындысын алу:

```
def sum(a, b):
    return a + b

result = sum(4, 6)
print(f"sum(4, 6) = {result}")
```

Нәтиже:

```
sum(4, 6) = 10
print(f"sum(3, 5) = {sum(3, 5)}")
```

Нәтиже:

```
sum(3, 5) = 8
```

5.3 Айнымалылардың көріну аймағы

Функцияда айнымалылар ауқымды және жергілікті болып бөлінеді.

Ауқымды айнымалылар функциядан тыс анықталады және бағдарламадағы кез келген функцияға қол жетімді.

Мысалы:

```
name = "Tom"

def say_hi():
    print("Hello", name)

def say_bye():
    print("Good bye", name)

say_hi()
say_bye()
```

Мұнда name айнымалысы ауқымды, оны екі функция да пайдалана алады.

Жергілікті айнымалының ауқымды айнымалылардан айырмашылығы, жергілікті айнымалы функцияның ішінде анықталады және тек сол функцияда қол жетімді.

Мысалы:

```
def say_hi():
    name = "Sam"
    surname = "Johnson"
    print("Hello", name, surname)

 def say_bye():
    name = "Tom"
    print("Good bye", name)

say_hi()
say_bye()
```

Бұл жағдайда екі функцияның әрқайсысында name жергілікті айнымалысы анықталған. Бұл айнымалылар бірдей аталса да, олар екі түрлі айнымалы болып табылады, олардың әрқайсысы тек өз функциясының ішінде қол жетімді. Сондай-ақ say_hi() функциясында surname жергілікті айнымалысы анықталған, оны say_bye() функциясында пайдалана алмаймыз.

Мысал_1. N бүтін санның S цифрларының қосындысын табатын (N – енгізілетін параметр, S – шығарылатын параметр) SumDigit(N, S) функциясын жазыңдар.

```
def sumD(n):  # параметрлі функцияны анықтау
    sumD = 0  # айнымалыны 0-ге теңестіру
    while n != 0:  # n нөлге тең емес болғанша
қайталау
        sumD += n % 10  # соңғы санды қосу
        n = n // 10  # соңғы санды өшіру
    return sumD  # функцияның мәнін қайтару

# негізгі бағдарлама
print(sumD(int(input("Сан енгізіңіз: "))))
```

функцияны шақыру
Нәтиже:
```
Сан енгізіңіз: 123
6
```

Мысал_2
Ұзындығы m болатын A тізімі берілген. Осы тізімнің бірінші және соңғы элементтерінің орындарын ауыстыратын функция жазу. Тізім ұзындығы және оның элементтері пернетақтадан енгізіледі. Бастапқы және соңғы тізімді экранға шығару.
```
def zam(X):
    tmp = X[0]  # Алғашқы элементті сақтау
    X[0] = X[len(X) - 1]  # Соңғы элементті бірінші
орынға қою
    X[len(X) - 1] = tmp  # Алғашқы элементті соңғы
орынға қою

A = []  # Тізімді анықтау
m = int(input('Тізім ұзындығын енгізіңіз: '))
# Тізімнің ұзындығын енгізу
for i in range(m):
    print(f'Тізімнің {i}-ші элементін енгізіңіз:')
    A.append(int(input()))

print("Бастапқы тізім:", A)
zam(A)  # Функцияны шақыру
print("Алғашқы және соңғы элементтері ауысқан
тізім:",
```
Нәтиже:
```
Тізім ұзындығын енгізіңіз: 5
Тізімнің 0 – ші элементін енгізіңіз: 5
Тізімнің 1 – ші элементін енгізіңіз: 8
Тізімнің 2 – ші элементін енгізіңіз: 12
Тізімнің 3 – ші элементін енгізіңіз: 9
Тізімнің 4 – ші элементін енгізіңіз: 18
```
Нәтиже:

```
Бастапқы тізім: [5, 8, 12, 9, 18]
Алғашқы және соңғы элементтері ауысқан тізім:
[18, 8, 12, 9, 5]
```

Тапсырмалар

1. Квадрат теңдеудің түбірлерін есептейтін koren функциясын жазу;
2. Әртүрлі фигуралардың аудандарын есептейтін функцияны жазу. Пайдаланушы қай фигураның аумағын есептегісі келетінін көрсетеді. Осыдан кейін бастапқы деректерді енгізеді. Мысалы, тіктөртбұрыш жағдайында ұзындық пен ені.
3. Жолды кері шығару функциясын жазу;
   Мысалы. "1234abcd"
   Күтілетін нәтиже: "dcba4321"
4. X^n шамасын есептейтін STEP(x, n) функциясын $b=2.7^5 + (a+1)^{-5}$ өрнегін есептеуге қолданыңыз. x – нақты сан, n – натурал сан.
5. Екілік факториалды есептейтін бүтін типтегі функция жазу;
   N!! = 1·3·5·…·N, егер N – тақ болса; N!! = 2·4·6·…·N, егер N – жұп болса, (N > 0) - бүтін типтегі параметр. Осы функцияның көмегімен 5 бүтін санның факториалын табу;
6. $y=x^2$ функциясын есептейтін квадрат функциясын жазып, оны $z=a^2 + a^4$ өрнегін шешуге қолдану;
7. n сан берілген. Осы тізбекте 7 саны қанша рет кездесетінін анықтайтын функция жазу;
8. Жолды қабылдайтын және бас және кіші әріптердің санын есептейтін Python функциясын жазу;
9. Екі жай сан бір-бірінен 2-ге айырмашылығы болса (мысалы, 41 және 43) «егіз» деп аталады. [n, 2n] кесіндісіндегі «егіздердің» барлық жұптарын басып шығаратын функция жазу, мұндағы n – 2-ден үлкен берілген натурал сан.
10. Екі A және B матрицалары берілген. Осы матрицалардың ең үлкен элементтерін ауыстыратын бағдарлама жазу. Матрицаның ең үлкен элементін табуды функция түрінде жазу;
11. Қашықтыққа байланысты таксимен жүру құнын есептейтін функцияны жазу. Функция аргументтер ретінде аталған

параметрлерді қабылдауы керек: km – километрмен жүру қашықтығы, мин. баға – такси қызметін және жолдың алғашқы үш километрін қоса алғанда базалық тариф, price_per_km – төртіншіден бастап әрбір километрге баға.

12. Кездейсоқ құпия сөзді тудыратын функцияны жазу. Құпия сөз 48-57 (сандар), 65-90 (бас әріптер латын әріптері) және 97-122 (кіші латын әріптері) ауқымындағы 8-ден 15-ке дейінгі Юникод таңбаларын қамтуы керек. Үш құпия сөзді құрыңыз және көрсету;
13. Сандардың ЕҮОБ мәнін табу функциясын жазыңыз. Содан кейін 165, 435 және 300 сандарының ең үлкен ортақ бөлгішін тауып экранға шығару.

Тест тапсырмалары

1. Функциядағы аргумент саны
print(“Name:”, “Anna”, “Age”, 25)
A. 3
B. 1
C. 4
D. 2
ANSWER: C
2. Белгілі бір функцияны шақыру коды
A. greet()
B. print()
C. input()
D. def()
ANSWER: A
3. Функциядағы аргументтер саны

```
def bmi(weight, height):
    index = weight / (height * height)
    print(index)
```

A. 3
B. 1
C. 2
D. 5
ANSWER: C

4. rect() функциясын шақырғанда қайтарылатын мәндер саны

```
def rect(d1, d2):
    area = d1 * d2
    perimeter = 2 * d1 + 2 *d2
    price = 1000 * area
    return area, perimeter, price
```

A. 1
B. 3
C. 2
D. 5

ANSWER: B

5. Шақырылған бұл функция қайтаратын мән

```
def rect(d1, d2):
    area = 0
    return area
    area = d1 * d2
```

A. 2
B. 0
C. 3
D. 6

ANSWER: A

6. Шақырылған бұл функция қайтаратын мән

```
def greet(name = “Guest”):
    print(“Welcome”, name)
greet()
```

A. қате
B. Welcome Guest
C. Welcome
D. Welcome name
E. name “Guest”

ANSWER: B

7. Код нәтижесі

```
def f(n):
    return (str(n))
print(f(5%2) *3)
```

A. 3

B. 6
C. 1
D. 111
ANSWER: D
8. Кодтың нәтижесі
fib = {1: 1, 2: 1, 3: 2, 4: 3}
print (fib.get(4, 0) + fib.get (7, 5))
A. 2
B. 4
C. 6
D. 8
ANSWER: D
9. Код нәтижесі
```
def fib (x):
      if x = = 0 or x = = 1 :
        return 1
      else:
        return fib (x-1) + fib (x-2)
print (fib (4))
```
A. 2
B. 1
C. 0
D. 5
ANSWER: D
10. Кодтың нәтижесі
```
def func(**kwargs):
      print (kwargs("zero"])
func(a = 0, zero = 8)
```
A. 0
B. 2
C. 4
D. 8
ANSWER: D

🕮 6 – бөлім. ФАЙЛДАРМЕН ЖҰМЫС

Файлдарды өңдеу кез келген веб-қосымшаның маңызды бөлігі болып табылады. Python көптеген әртүрлі файл түрлерін қолдайды, бірақ оларды екі түрге бөлуге болады: мәтіндік және екілік. Мәтіндік файлдар, мысалы, cvs, txt, html кеңейтімі бар файлдар, жалпы алғанда, ақпаратты мәтін түрінде сақтайтын кез келген файлдар. Екілік файлдар кескіндер, аудио және бейне файлдар.

Файл түріне байланысты онымен жұмыс істеу сәл өзгеше болуы мүмкін.

Файлдармен жұмыс істеу кезінде белгілі бір әрекеттер тізбегі сақталу керек:

1. open() әдісі арқылы файлды ашу;
2. read() әдісі арқылы файлды оқу немесе write() әдісі арқылы файлға жазу;
3. close() әдісі арқылы файлды жабу

Файлды ашу және жабу

Файлмен жұмыс істеуді бастау үшін open() функциясы арқылы ашу керек:

```
open(file, mode)
```

Функцияның бірінші параметрі файлға жолды көрсетеді. Файл жолы абсолютті болуы мүмкін, яғни диск әрпінен басталады, мысалы, `C://somedir/somefile.txt`. Немесе салыстырмалы болуы мүмкін, мысалы, somedir/somefile.txt - бұл жағдайда файл Python скриптің іске қосылған орнына қатысты ізделеді.

Екінші аргумент mode - режим файлды ашу режимін біз онымен не істейтінімізге байланысты орнатады. 4 жалпы режим бар:.

- r (Read). Файл оқу үшін ашылады. Егер файл табылмаса, FileNotFoundError ерекше жағдайы шығарылады.
- w (Write). Файл жазу үшін ашылады. Егер файл жоқ болса, ол құрылады. Егер ұқсас файл бұрыннан бар болса, ол жаңадан құрылады және тиісінше ондағы ескі деректер жойылады.
- a(Append). Файлды жазғанға дейін ашу. Егер файл жоқ болса, ол құрылады. Егер ұқсас файл бұрыннан бар болса, онда деректер оның соңына жазылады.

– b (Binary). Екілік файлдармен жұмыс істеу үшін қолданылады. Басқа режимдермен бірге қолданылады - w немесе r, мысалы, rb (екілік файлдарды оқу) және wb (екілік файлдарды жазу).

Мысал_1

```
f = open("demofile.txt", "r")
print(f.read())
```

Егер файл басқа жерде болса, файлға жолды көрсету керек, мысалы:

Файлды басқа жерде ашу

Мысал_2

```
f = open("D:\\myfiles\welcome.txt", "r")
print(f.read())
```

Файлдың тек бөліктерін оқу

Үнсіз read() әдісі барлық мәтінді қайтарады, бірақ қанша таңбаны қайтарғыңыз келетінін де көрсете аласыз:

Мысал_3

Файлдың алғашқы 5 таңбасын қайтару:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Жолдарды оқу

readline() әдісі арқылы бір жолды қайтаруға болады.

Мысал_4

Файлдың бір жолын оқу:

```
f = open("demofile.txt", "r")
print(f.readline())
```

Файлды жабу

Мысал_5

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

Бар файлға жазу

Мысал_6

«demofile2.txt» файлын ашып, оған мазмұнды қосу:

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
```

```
f.close()
```
Мысал_7
"demofile3.txt" файлын ашып, мазмұнын қайта жазыңыз:
```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```
Жаңа файл құру
Мысал_8
«myfile.txt» деп аталатын файлды құру:
```
f = open("myfile.txt", "x")
```
Нәтиже: жаңа бос файл құрылды!
Файлды жою
Файлды жою үшін ОЖ модулін импорттап, оның os.remove() функциясын іске қосу керек:
Мысал_9
"demofile.txt" файлын жойыңыз:
```
import os
os.remove("demofile.txt")
```
Қатені болдырмау үшін файлды жоймас бұрын оның бар-жоғын тексеруге болады:
Мысал_10
Файлдың бар - жоғын тексеріңіз, содан кейін оны жойыңыз:
```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```
Қалтаманы жою
Қалтаманы толығымен жою үшін os.rmdir() әдісін пайдалану:
Мысал_11
«myfolder» қалтамасын жойыңыз:
```
import os
os.rmdir("myfolder")
```
Ескерту. Тек бос қалтамаларды жоюға болады.
Мысал 1:
1 - қадам: Файлға жазу (write)

Файлды жазу режимінде ашамыз
```
with open("my_file.txt", "w") as file:
    file.write("Сәлем, әлем!\n")
    file.write("Бұл - Python-да файлмен жұмыс істеу
мысалы.\n")
    file.write("Файлға бірнеше жол жаза аламыз.")
```
Нәтиже:
```
"my_file.txt" файлында мынадай мазмұн пайда болады:
Сәлем, әлем!
Бұл - Python-да файлмен жұмыс істеу мысалы.
Файлға бірнеше жол жаза аламыз.
```
2 - қадам: Файлдан оқу (read)
Файлды оқу режимінде ашамыз
```
with open("my_file.txt", "r") as file:
    content = file.read()
    print("Файлдың ішіндегі мәтін:\n")
    print(content)
```
Нәтиже:
Файлдың ішіндегі мәтін:

```
Сәлем, әлем!
Бұл - Python-да файлмен жұмыс істеу мысалы.
Файлға бірнеше жол жаза аламыз.
```
3 - қадам: Файл соңына қосу (append)
Файлдың соңына жаңа жол қосамыз
```
with open("my_file.txt", "a") as file:
    file.write("\nБұл — жаңадан қосылған жол.")
```
Жаңартылған файл мазмұны:
```
Сәлем, әлем!
Бұл - Python-да файлмен жұмыс істеу мысалы.
Файлға бірнеше жол жаза аламыз.
Бұл - жаңадан қосылған жол.
```
Мысал 2
Мәтіндік файлдан жолдарды оқып, жолдар мен сөздердің санын шығару:
Берілген файлда бірнеше жол бар. Сол файлдан:

- Жалпы жол санын (line count),
- Жалпы сөз санын (word count),
- Жалпы әріп санын (character count) есептеп шығар.

```
# Файлды жазу
with open("text_sample.txt", "w") as file:
    file.write("Сәлем, бұл бірінші жол.\n")
    file.write("Python тілі өте қызықты.\n")
    file.write("Файлдармен жұмыс істеп үйреніп
жатырмыз.\n")
# Есептеу
line_count = 0
word_count = 0
char_count = 0
with open("text_sample.txt", "r") as file:
    for line in file:
        line_count += 1
        words = line.split()
        word_count += len(words)
        char_count += len(line)

print(f"Жолдар саны: {line_count}")
print(f"Сөздер саны: {word_count}")
print(f"Әріптер (таңбалар) саны: {char_count}")
```

Нәтиже:

```
Жолдар саны: 3
Сөздер саны: 12
Әріптер (таңбалар) саны: 108
```

Ескерту: әріптер санына бос орындар мен \n символдары да кіреді.

 **Тапсырмалар**

1. users.txt файлы берілген. Әр жолда қолданушы аты мен электронды пошта жазылған, мысалы:
   Aidos, aidos@gmail.com
   Dana, dana@mail.ru

Файлды оқып, әр қолданушының атын және поштасын бөлек шығару;

2. log.txt файлында веб - сервер журналдары сақталған. Мысалы:
   [2024-04-20 10:32:01] GET /home
   [2024-04-20 10:32:15] POST /login
   Тек POST сұрауларды тауып, оларды санау және шығару;
3. data.txt файлы берілген, кейбір жолдар бос болуы мүмкін. Файлдағы бос емес жолдардың санын есептеу;
4. Файлда әр жолда бір бүтін сан берілген:
   23
   45
   12
   89
   Барлық сандарды оқып, олардың:
   - Қосындысын;
   - Орташа мәнін;
   - Максимум және минимумын табу;
5. Файл ішінде үлкен мәтін берілген. Мәтінді сөздерге бөліп, ең жиі кездесетін 5 сөзді шығару;
6. input.txt файлында әртүрлі мәліметтер бар. Тек "error" сөзін қамтитын жолдарды errors.txt деген жаңа файлға көшіру;
7. Қолданушы енгізген файл атауы бойынша деректерді оқып, басқа файлға дәл сол күйінде жазу. Қарапайым "көшіру" логикасын жазу;
8. students.csv файлында білім алушылар туралы ақпарат бар:
   name,age,grade
   Ali,17,A
   Dana,16,B
   CSV - файлды оқып, әр білім алушыны жеке шығару: аты, жасы, бағасы.
9. poem.txt деген файлда өлең жазылған. Файлдағы әр жолды нөмірлеп, шығару;
10. text.txt ішінде бірнеше сөйлем бар. Нүкте (.) арқылы сөйлем санын анықтау (бірнеше нүкте қатар келуі мүмкін екенін ескеріп)

11. part1.txt, part2.txt, part3.txt деген 3 файл берілген. Барлық файлдардың мазмұнын оқып, бір combined.txt файлына ретімен жазу;
12. data.txt файлы берілген, ішінде бірнеше жол бар. Файлдағы жолдарды соңынан басына қарай шығару (яғни ең соңғы жол бірінші болып шығады)
13. access.log деген файлда веб-сервердің логтары сақталған. Әр жолда IP-адрес пен сұраныс түрі (GET, POST, т.б.) бар:
    192.168.0.10 - GET /index.html
    10.0.0.5 - POST /login
    192.168.0.10 - GET /about
    - Файлдан барлық бірегей (уникальный) IP-адрестерді табу;
    - Олардың қайсысы қанша рет кездесті – санау;
    - Ең жиі кездескен IP - адресті анықтау;

Тест тапсырмалары

1. “Hello world!” сөзін файлға жазу коды
A. file.write(“Hello world!”)
B. write(file, “Hello world!”)
C. write(“Hello world!”, file)
D. write(“Hello”, file)
ANSWER: A
2. Python - да рұқсат етілетін файл атауы
A. find_sum_1
B. 1check_condition
C. 0_check_condition
D. E. #check_condition
ANSWER: A
3. Readlines () әдісі қайтарады
A. b
B. a
C. w
D. r
ANSWER: B
4. Файлдың readlines() әдісі қайтаратын деректер түрі

A. жолдар тізімі
B. бүтін сандар тізімі
C. жолдар кортежі
D. жиындар
ANSWER: A
5. Мәтіндік файлдың барлық мазмұнын жол ретінде оқу үшін пайдалануға болатын файл әдісі
A. readlines()
B. read_file_to_str()
C. readline()
D. read()
ANSWER: D
6. Файлдармен жұмыс істеуді әмбебап ету үшін python бағдарлама - ларында файл жолдарында __ қою ұсынылады
A. кері слеш (\)
B. тура слеш (/)
C. үтір (,)
D. нүкте (.)
ANSWER: B
7. jack_russell.png атты екілік файл орындалатын бағдарламамен бір бумада орналасқан. Бұл файлды оқу үшін ашу
A. open('jack_russell.png', 'rb')
B. open('jack_russell.png', 'r')
C. open('jack_russell.png', 'wb')
D. open('jack_russell.png')
ANSWER: A
8. Файлдың атын түрінен ажырататын белгі
A. үтір
B. нүкте
C. нүктелі үтір
D. қос нүкте
ANSWER: B
9. _____ файл түрінің мазмұнын Блокнот сияқты мәтіндік редакторда көруге болады

A. Мәтіндік
B. Екілік
C. Оқылатын
D. Орысша
ANSWER: A
10. Файлды бағдарламада пайдаланбас бұрын, ол ____ болуы керек
A. шифрланған
B. ашық
C. жабық
D. форматталған
ANSWER: B

🕮 7. ҚАТЕЛЕР МЕН ЕРЕКШЕ ЖАҒДАЙЛАРДЫ ӨҢДЕУ

7.1 Ерекше жағдайларды өңдеу операторлары

try ... except ... finally конструкциясы

Python тілінде бағдарламалау кезінде біз екі түрлі қателікпен кездесеміз.

1. Синтаксистік қателер (**syntax error**).

Бұл қателер бағдарламалау тілі синтаксисінің ережелері бұзылған кезде пайда болады. Мысалы, жақшаларды немесе қос нүктелерді ұмытқан кезде.

Егер сіз PyCharm сияқты ортада жұмыс істесеңіз, ол синтаксистік қателерді өзі анықтап, оларды ерекшелей алады.

2. Орындау кезіндегі қателер *(***runtime error***)*.

Бұл қателер бағдарлама орындалып жатқанда пайда болады. Мұндай қателерді **ерекше жағдайлар** деп атайды.

Мысалы, жолды (string) санға (integer) түрлендіру мысалын алайық:

```
string = "5"
number = int(string)
print(number)
```

Бұл скрипт сәтті орындалады, себебі "5" деген жол санға түрлене алады. Бірақ басқа мысалды қарастырайық:

```
string = "hello"
```

```
number = int(string)
print(number)
```
Мұнда бағдарлама орындалу кезінде **ValueError** деген ерекше жағдай орындалады, өйткені "hello" жолын санға айналдыру мүмкін емес:
```
ValueError: invalid literal for int() with base
10: 'hello'
```
Бір қарағанда, "hello" жолы сан емес екені анық, бірақ біз қолданушы енгізетін деректермен жұмыс істегенде, дәл осындай күтпеген мәндер келуі мүмкін:
```
string = input("Сан енгізіңіз: ")
number = int(string)
print(number)
```

7.2 Ерекше жағдайлар түрлері

Python тілінде ерекше жағдайлардың(исключения) бірнеше негізгі түрлері бар.

Төменде ең жиі кездесетін және бағдарламалауда маңызды рөл атқаратын негізгі ерекше жағдайлар (қателік түрлері) берілген:

Қате түрі	**Сипаттамасы**
NameError	Атау (айнымалы, функция) табылмаған кезде шығатын қате
TypeError	Қате типтермен амалдар орындағанда шығады (мысалы, сан мен жолды қосу)
ValueError	Айнымалының мәні дұрыс типте болса да, мағынасы сәйкес болмағанда пайда болады (мысалы, "hello" → int)
IndexError	Индекс тізім немесе жол шегінен тыс болса, шығатын қате
KeyError	Сөздікте (dictionary) жоқ кілтке жүгінгенде шығады
ZeroDivisionError	Сан нөлге бөлінген кезде шығатын қате
FileNotFoundError	Ашылмақшы болған файл табылмаған жағдайда пайда болады

Қате түрі	Сипаттамасы
AttributeError	Объектіде сұралған қасиет (атрибут) болмаған кезде шығатын қате
ImportError	Модуль немесе оның ішіндегі бірдеңе табылмаған кезде пайда болады

Егер қолданушы дұрыс емес мән енгізсе, бағдарлама қателікпен тоқтап қалады**.** Мұндай жағдайлардың алдын алу үшін Python тілінде **try ... except** конструкциясы қолданылады.

Синтаксисі:

```
try:
   # Негізгі нұсқаулар (қате шығуы мүмкін код)
except [Қате түрі]:
   # Қате шыққанда орындалатын код
```

- try блогына қате беруі мүмкін код жазылады;
- Егер осы кодта қате туындаса, орындау тоқтап, except блогына өтеді;
- except кілттік сөзімен бірге нақты қате түрін (мысалы, ValueError, KeyError) көрсетуге болады.

Мысал:

```
try:
    number = int(input("Сан енгізіңіз: "))
    print("Енгізілген сан:", number)
except:
    print("Түрлендіру сәтсіз аяқталды")
print("Бағдарлама аяқталды")
```

Қате енгізсек:

Нәтиже:

```
Сан енгізіңіз: hello
Түрлендіру сәтсіз аяқталды
Бағдарлама аяқталды
```

Дұрыс сан енгізсек:

```
Сан енгізіңіз: 22
Енгізілген сан: 22
Бағдарлама аяқталды
```

finally блогы

Ерекше жағдайларды өңдеу кезінде finally деген қосымша блокты да қолдануға болады. Бұл блоктың ерекшелігі - ол қате шықса да, шықпаса да міндетті түрде орындалады.

Мысал:

```
try:
    number = int(input("Сан енгізіңіз: "))
    print("Енгізілген сан:", number)
except:
    print("Түрлендіру сәтсіз аяқталды")
finally:
    print("try блогының жұмысы аяқталды")
print("Бағдарлама аяқталды")
```

Нәтиже:

- Егер дұрыс сан енгізілсе → except орындалмайды, бірақ finally бәрібір орындалады.
- Егер қате енгізілсе → except орындалады, және finally де орындалады.

Ескерту:

finally блогы қателерді өңдемейді**,** ол тек соңғы әрекеттер үшін (мысалы, файлды жабу, ресурстарды босату) қолданылады.

Мысал 1

```
try:
    x = int(input("Бөлінетін санды енгізіңіз: "))
    y = int(input("Бөлгішті енгізіңіз: "))
    result = x / y
    print("Нәтиже:", result)
except ZeroDivisionError:
    print("Қате: Нөлге бөлуге болмайды!")
```

Нәтиже:

```
Қате: Нөлге бөлуге болмайды!
```

Мысал 2

```
try:
# Жолды санға айналдыру
    age = int("он сегіз")
    print("Жасы:", age)
```

```
except ValueError:
    print("Қате: Жасыңызды тек санмен енгізіңіз!")
```
Нәтиже:
```
Қате: Жасыңызды тек санмен енгізіңіз!
```
Мысал 3
```
try:
    fruits = ["алма", "банан", "алмұрт"]
    print(fruits[5])
# Индекс 5 - жоқ элемент индексі
except IndexError:
    print("Қате: Мұндай индекс жоқ!")
```
Нәтиже:
```
Қате: Мұндай индекс жоқ!
```
Мысал 4
```
try:
    student = {"аты": "Алия", "жасы": 18}
    print(student["мамандығы"])  # Мұндай кілт жоқ
except KeyError:
    print("Қате: Мұндай кілт сөздікте жоқ!")
```
Нәтиже:
```
Қате: Мұндай кілт сөздікте жоқ!
```
Мысал 5
```
try:
    file = open("мәліметтер.txt", "r")
    content = file.read()
    print(content)
    file.close()
except FileNotFoundError:
    print("Қате:       'мәліметтер.txt'       файлы
табылмады!")
```
Нәтиже:
```
Қате: 'мәліметтер.txt' файлы табылмады!
```

Тапсырмалар

1. Калькулятор + қатені өңдеу.
Пайдаланушыдан екі сан мен амал (+, -, *, /) енгізуін сұраңыз. Барлық мүмкін болатын қателерді (ValueError, ZeroDivisionError, т.б.) өңдеу;
2. Пайдаланушы деректерін тексеру.
Пайдаланушы логин және жасы бар сөздікті енгізеді. Егер кілт жетіспесе - KeyError, егер жас сан болмаса - ValueError өңдеу;
3. Динамикалық тізім индекстері.
Пайдаланушы кез келген ұзындықтағы тізім құрады. Индекс бойынша элемент сұрағанда, IndexError туындаса, элементті қосуға мүмкіндік беру;
4. Өз қателігіңізді raise арқылы жасау.
Функцияда пайдаланушы теріс сан енгізсе, өзіңіз raise ValueError("Теріс санға рұқсат жоқ") қатесін жасаңыз, және оны try...except арқылы өңдеу;
5. Базадағы қолданушыны табу.
Қолданушылар базасы сөздік түрінде берілген. Ізделетін логин болмағанда - KeyError. Логин мен пароль дұрыс болмаса - өзіңіз raise арқылы Exception("Қате логин немесе пароль") шығару;
6. Тізімнен ең үлкен санды табу.
Пайдаланушы сандар тізімін енгізеді. Барлық мәндер сан екеніне көз жеткізу. Егер бірі сан болмаса - қатені өңдеу, және қайта енгізуді сұрау;
7. Файлға жазу және жабу (finally)
Мәліметті файлға жазып, finally блогында файлды жабу. Егер жазу кезінде қате шықса - қатені өңдеу;
8. Нақты уақыттағы өңдеу(while True)
Пайдаланушыдан қайта - қайта мән сұралады. Егер ол exit десе - цикл тоқтайды. Қате мән енгізілсе - қатені өңдеу, қайта сұрау;
9. Функция ішінде бірнеше ерекше жағдай
Бір функцияда бірнеше мүмкін қатені өңдеу: файл ашу, дерек оқу, түрлендіру, бөлу, т.б.

10. Сандарды сұрыптау

Пайдаланушы тізім енгізеді (input().split()). Барлығы сан болмаған жағдайда - қате өңделеді. Соңында сұрыптап көрсету.

11. Лог файл жасау

Программада қате шыққан кезде, қате туралы хабарды error_log.txt файлына жазу (мысалы: except Exception as e: → log.write(str(e))).

12. Бот үшін жауап генерациялау

Сценарийде қолданушы сұрақ қояды, ал бот жауап береді. Егер сұрақ бос болса немесе дұрыс форматта болмаса - қате өңделіп, “Сұрағыңызды дұрыс қойыңыз” деген жауап шығарылады.

13. Сізде students.txt атты файл бар деп елестетіңіз. Бұл файлда әр жолда студенттің аты және бағалары (мысалы: Айжан 4 5 3 5). Бағдарлама осы файлды оқып, әр студенттің орташа бағасын есептеп көрсету.

– Егер файл табылмаса - FileNotFoundError өңделеді.
– Егер бір студенттің жолында баға орнына мәтін жазылса - ValueError өңделіп, ол студент өткізіледі.
– finally блогында "Бағдарлама аяқталды" деген хабар шығарылады.

Мысал файл мазмұны (students.txt):

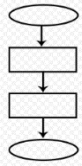
Айжан 4 5 3 5
Нұржан 5 5 бес 4
Санжар 3 4 2

Нәтиже:

Айжан - GPA: 4.25
Нұржан - Қате: бағалар сан емес!
Санжар - GPA: 3.00
Бағдарлама аяқталды

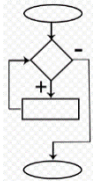
Аралық аттестаттау тапсырмалары

1. Алгоритм түрі



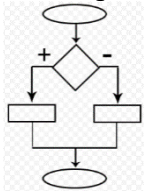
A. Тармақталған
B. Сызықтық
C. Циклдік
D. Аралас
ANSWER: B

2. Алгоритм түрі



A. Тармақталған
B. Сызықтық
C. Циклдік
D. Аралас
ANSWER: C

3. Алгоритм түрі



A. Тармақталған
B. Сызықтық
C. Циклдік
D. Аралас
ANSWER: A

4. Дұрыс жазылған меншіктеу операторын таңда
A. 10 = x

B. y = 7,8
C. a = 5
D. a == b + x
ANSWER: C
5. Енгізу операторы
A. input()
B. print()
C. int()
D. random()
ANSWER: A
6. Алдыңғы шартты цикл операторы
A. while
B. for
C. if
D. print
ANSWER: A
7. Python – да қалдық табу амалы
A. %
B. +
C. /
D. *
ANSWER: A
8. Функция формальды параметрлері жазылады
A. Функция тақырыбында
B. Функция денесінде
C. Функцияны шақырғанда
D. Негізгі программада
ANSWER: A
9. len() функциясы қызметі
A. Жолдың ұзындығын қайтарады
B. Кездейсоқ санды қайтарады
C. Таңба нөмірін қайтарады
D. Санның модулін қайтарады
ANSWER: A
10. Бағдарламаға модульді қосу тәсілі
A. import math

B. import math()
C. import (math)
D. import.math
ANSWER: A
11. Python бағдарламалау тілін құрушы
A. Гвидо Ван Россум
B. Дэвид Паттерсон
C. Эрвин Дональд Кнут
D. Джеймс Артур Гослинг
ANSWER: A
12. a = 345. print(a//100) командасы нәтижесі
A. 3
B. 5
C. 4
D. 34
ANSWER: A
13. int(input()) қызметі
A. Пернетақтадан сан енгізіледі
B. Экранға хабарлама шығады
C. Пернетақтадан бүтін сан енгізіледі
D. Пернетақтадан таңбалар енгізіледі
ANSWER: C
14. Тік жақшаға алынған және үтірмен бөлінген тізімдердегі мәндер ___деп аталады
A. жолдар
B. сандар
C. элементтер
D. индексаторлар
ANSWER: C
15. Белгілі бір мәселені шешетін бағдарламаның жеке функционалды тәуелсіз бөлігі _____ деп аталады
A. блок
B. параметр
C. функция
D. өрнек
ANSWER: C

16. Функция сипаттамасындағы алғашқы жол
A. кіріспе
B. инициализация
C. дене
D. тақырып
ANSWER: D
17. Фунциядағы бағдарламалық код бөлігі
A. кіріспе
B. инициализация
C. денесі
D. тақырып
ANSWER: C
18. Функцияны орындау үшін
A. шақырады
B. импорттайды
C. анықтайды
D. экспорттайды
ANSWER: A
19. Python – да шартты оператор
A. if
B. do
C. while
D. for
ANSWER: A
20. Python - дағы логикалық оператор "және"
A. &&
B. ||
C. !
D. and
ANSWER: D
21. Функция денесінде сипатталатын параметр
A. ауқымды
B. жасырынған
C. формальды
D. жергілікті
ANSWER: D

22. Python - дағы логикалық оператор "немесе"
A. &&
B. ||
C. !
D. or
ANSWER: D
23. Python - дағы логикалық оператор "емес"
A. &&
B. ||
C. not
D. and
ANSWER: C
24. Python. "тең емес салыстыру амалы"
A. = =
B. =
C. !=
D. >
ANSWER: C
25. Python тіліндегі range() функциясындағы "start " параметрінің бастапқы мәні
A. 0
B. 1
C. -1
D. 10
ANSWER: A
26. for циклін үзу операторы
A. skip
B. next
C. break
D. stop
ANSWER: C
27. 0.00.0 ден 1.01.0 дейінгі аралықтағы нақты кездейсоқ сандарды қайтаратын функция
A. random()
B. randint()
C. uniform()

D. random_integer()
E. randomInteger()
ANSWER: A
28. range(3, 10, 2) функциясының мәні
A. [3, 5, 7, 9]
B. [2, 4, 6, 8]
C. [3, 6, 9]
D. [2, 5, 8]
ANSWER: A
29. range(5) функциясының мәні
A. [0, 1, 2, 3, 4]
B. [1, 2, 3, 4, 5]
C. [0, 1, 2, 3, 4, 5]
D. [1, 2, 3, 4, 5, 6]
ANSWER: A
30. Python - да while циклінің келесі итерациясына өту командасы
A. next
B. continue
C. skip
D. pass
ANSWER: B
31. x айнымалысы 0 және 55 аралығында екенін тексеретін логикалық өрнекті таңдаңыз
A. x > 0 or < 5
B. x > 0 or x < 5
C. x > 0 and x < 55
D. x > 0 and < 5
ANSWER: C
32. Python тіліндегі түсініктемелерді білдіретін таңба
A. //
B. --
C. /*
D. #
ANSWER: D
33. Python - да бос тізім
A. array[]

B. array()
C. []
D. {}
ANSWER: C
34. Python - да тізім ұзындығын алу функциясы
A. length()
B. size()
C. count()
D. len()
ANSWER: D
35. Код нәтижесі
zeros = [0] * 10
print(len(zeros))
A. 10
B. l2
C. 0
D. 100
ANSWER: A
36. Код нәтижесі
words = ['Hello', 'Python']
print('-'.join(words))
A. Hello-Python
B. Hello-Python-
C. HelloPython-
D. -HelloPython
ANSWER: A
37. Төмендегі код нәтижесі
numbers = [10, 20, 30, 40]
del numbers[0:6]
print(numbers)
A. [20, 40]
B.[30, 40]
C.[10, 20]
D.[]
ANSWER: D
38. Төмендегі код нәтижесі

list1 = [[1, 8, 9], [4, 8, 12, 16], [0, 2, 7]]
print(list1[0][1] + list1[1][2] + list1[2][2])
A. 27
B. 5
C. 13
D. 38
ANSWER: A
39. Төмендегі код нәтижесі
list1 = [[1, 8, 7, 4], [1, 3, 4, 5, 6], [2, 7, 2], [2, 6, 7, 8]]
print(max(list1))
A.[1, 8, 7, 4]
B.[2, 7, 2]
C.[2, 6, 7, 8]
D.[1, 3, 4, 5, 6]
ANSWER: B
40. Төмендегі код нәтижесі
matrix = [[1, 2, 8, 0],
[-4, 1, 9, 4],
[41, 71, 2, -2]]
print(matrix[2][3])
A. 9
B. -2
C. 4
D. 2
ANSWER: B
41. Бір матрицаны басқа басқа матрицаға көбейтуге болады
A. бірінші матрицадағы бағандар саны екінші матрицадағы бағандар санымен бірдей
B. бірінші матрицадағы бағандар саны екінші матрицадағы жолдар санымен бірдей
C. бірінші матрицадағы жолдар саны екінші матрицадағы жолдар санымен бірдей
D. бірінші матрицадағы жолдар саны екінші матрицадағы бағандар санымен бірдей
ANSWER: B

42. Жақшаға алынған және үтірмен бөлінген кортеждердегі мәндер _____ деп аталады
A. сандар
B. мәндер
C. элементтер
D. индексаторлар
ANSWER: C
43. Кортеждердің пайда болуының негізгі себептері
A. өзгермейтіндігі
B. жылдамдық (өнімділік)
C. қауіпсіздік
D. барлық жауаптар дұрыс
ANSWER: D
44. numbers кортежіндегі элементтер саны
numbers = (3, 5, 7, 9)
A. 4
B. 0
C. 5
D. 3
ANSWER: A
45.Төмендегі код нәтижесі
names = ('Michael', 'John', 'Freddie')
print(names)
A. 'Michael', 'John', 'Freddie'
B. ('Michael', 'John', 'Freddie')
C. [''Michael'', ''John'', ''Freddie'']
D. (Michael, John, Freddie)
ANSWER: B
46. Төмендегі код нәтижесі
colors = ('red', 'green', 'blue')
colors[0] = 'black'
print(colors)
A. қате туындайды
B. ('red', 'green', 'blue')
C. ('black', 'green', 'blue')
D. ('blue', 'black', 'green')

ANSWER: A
47. Төмендегі бағдарламалық код нәтижесі
```
a = (3, 4, 5)
for i in range(3):
    a[i] += 3
print(sum(a))
```
A. 12
B. 21
C. 9
D. қате
ANSWER: D
48. numbers кортежіндегі 337 санының индексі
```
numbers = (0, 1090, 7, 21, 17, 337, 22)
```
A. 5
B. 4
C. 6
D. 7
ANSWER: A
49. Кірістірілген кортеж үшін numbers[0][1][1] тең болады
```
numbers = ((0, (9, 2)), (1, (4, 6, 3), (5, 2, 3), 8, 3))
```
A. 9
B. 2
C. 4
D. 1
ANSWER: B
50. Тізімнің бірінші элементіне қатынау индексі
A. 0
B. 1
C. -1
D. 2
ANSWER: A
55. range(5) функциясының мәні
A. [0, 1, 2, 3, 4]
B. [1, 2, 3, 4, 5]
C. [0, 1, 2, 3, 4, 5]
D. [1, 2, 3, 4, 5, 6]

ANSWER: A
52. Бұл сандар туралы түсінік ежелгі дәуірде объектілерді санау қажеттілігіне байланысты пайда болды. Бұл сандардың аты
A. комплекс
B. бүтін
C. натурал
D. жай
ANSWER: C
53. Натурал емес сан
A. 5
B. 2
C. 10
D. -3
ANSWER: D
54. Нақты сан
A. -3
B. 10
C. 5
D. 1.2
ANSWER: D
55. Python – да түсініктеме белгісі
A. //
B. --
C. /*
D. #
ANSWER: D
56. Python – да бос тізім
A. array[]
B. array()
C. []
D. {}
ANSWER: C
57. Python – да тізім ұзындығын анықтау
A. length()
B. size()
C. count()

D. len()
ANSWER: D
58. Python - да тізімнің бірінші элементіне қатынау индексі
A. 0
B. 1
C. -1
D. start
ANSWER: A
59.Төмендегі код нәтижесі
myset = set([1, 2, 2, 3, 4, 4, 4])
print(len(myset))
A. 7
B. 4
C. 5
D. 6
ANSWER: B
60. Төмендегі код нәтижесі
myset = set('ъъ эээ ююююю ыыыыя')
print(len(myset))
A. 5
B. 15
C. 14
D. 4
ANSWER: A
61. Төмендегі код нәтижесі
myset = {'Yellow', 'Orange', 'Black'}
print(myset[1])
A. Yellow
B. Orange
C. Black
D. қате
ANSWER: D
62. myset жиынына item жаңа элемент элементінің мәнін қосу жолын таңдаңыз
A. myset.add(item)
B. myset.update(item)

C. myset.append(item)
D. myset.insert(item)
ANSWER: A
63. Төмендегі код нәтижесі
myset = {'Yellow', 'Orange', 'Black'}
myset.add('Blue')
myset.add('Orange')
print(myset)
A.{'Yellow', 'Black'}
B.{'Blue', 'Orange', 'Yellow', 'Black'}
C.{'Yellow', 'Orange', 'Black'}
D.{'Blue', 'Orange', 'Yellow', 'Orange', 'Black'}
ANSWER: B
64. Жиынның discard() және remove() әдістерінің айырмашылығы
A. remove() әдісі егер элемент жиында табылмаса, ерекше жағдайды шығарады
B. бірдей жұмыс істеу
C. егер элемент жиында табылмаса, discard() әдісі ерекше жағдайды шығарады
D. A және C опциялары дұрыс
ANSWER: A
65. Төмендегі код нәтижесі
myset = {'python'}
item = myset.pop()
print(item, len(myset))
A. python 1
B. python 0
C. p5
D. python 2
ANSWER: A
66. Төмендегі код нәтижесі
myset = set('python')
item = myset.pop()
print(item, len(myset))
A. p 5
B. python 1

C. p 0
D. p 6
ANSWER: A
67. Төмендегі код нәтижесі
```
myset = set()
item = myset.pop()
print(item)
```
A. 0
B. -1
C. set()
D. қате
ANSWER: D
68. Төмендегі код нәтижесі
```
myset = set()
for i in range(10):
    if i % 2 == 0:
        myset.add('even')
    else:
        myset.add('odd')
print(len(myset))
```
A. 1
B. 2
C. 4
D. 5
ANSWER: B
69. Төмендегі код нәтижесі
```
set1 = {10, 20, 30, 40}
set2 = set(range(50))
print(set1.issubset(set2))
```
A. True
B. False
C. boolean
D. set2
ANSWER: A
70. Төмендегі код нәтижесі
```
set1 = set('Stepik')
```

set2 = set('stepik')
print(set1.issubset(set2))
A.True
B. False
C. boolean
D. қате
ANSWER: B
71. Төмендегі код нәтижесі
word = 'beegeek'
set1 = set(word*3)
set2 = set(word[::-1]*2 + 'stepik')
print(set1 < set2)
A. False
B. True
C. қате
D. boolean
ANSWER: B
72. Төмендегі код нәтижесі
set1 = {1, 2, 3, 4, 5, 6, 7, 8}
list1 = [1, 2, 3, 4, 5]
print(set1.issuperset(list1))
A. False
B. True
C. boolean
D. [1, 2, 3, 4, 5]
ANSWER: B
73. Төмендегі код нәтижесі
set1 = {'q', 'w', 'e', 'r', 't', 'y'}
list1 = ['y', 'w', 'r']
print(set1 >= list1)
A. False
B. True
C. қате
D. boolean
ANSWER: C
74. Төмендегі код нәтижесі

set1 = set(range(1, 10))
set2 = set(range(10, 20))
print(set1.isdisjoint(set2))
A. False
B. True
C. қате
D. boolean
ANSWER: B
75. Python - да тізім соңына бірнеше элементтерді қосу
A. append()
B. insert()
C. add()
D. push()
ANSWER: A
76. Төмендегі код нәтижесі
set1 = {1, 2, 3, 4, 5}
set2 = frozenset(range(1, 6))
print(set1 == set2)
A. қате
B. False
C. True
D. equal
ANSWER: C
77. Сөздіктер (dict типі) _____ болып табылады
A. өзгермелі
B. өзгермейтін
C. жиын
D. тізімдер
ANSWER: A
78. Төмендегі код нәтижесі
stuff = {1: 'aaa', 2: 'ббб', 3: 'ввв'}
print(stuff[3])
A. ббб
B. ввв
C. aaa
D. 2

ANSWER: B
79. Төмендегі код нәтижесі
my_dict = dict([('first', 1), ('second', 2), ('third', 3)])
print(my_dict)
A. [('first', 100), ('second', 200), ('third', 300)]
B. {1: 'first', 2: 'second', 3: 'third'}
C. {'first': 1, 'second': 2, 'third': 3}
D. {first: 1, second: 2, third: 3}
ANSWER: C
80. Төмендегі код нәтижесі
my_dict = dict.fromkeys(['a', 'b', 'c'], -1)
print(my_dict['b'])
A. -1
B. c
C. b
D. a
ANSWER: A
81. Төмендегі код нәтижесі
my_dict = dict.fromkeys(['a', 'b', 'c'], -1)
print(my_dict['d'])
A. a
B. қате
C. b
D. c
ANSWER: B
82. Төмендегі код нәтижесі
my_dict = {1: [0, 1], 2: [2, 3], 3: [4, 5]}
print(my_dict[2][1])
A. [0, 1], [2, 3]
B. [2, 3]
C. 3
D. 5
ANSWER: C
83. Сөздік элементінің кілт бөлігін көрсетіңіз: өзгермейтін болуы керек мән
A. кілт

B. мән
C. тізім
D. идентификатор
ANSWER: A
84. Төмендегі код нәтижесі
stuff = {1:'ааа', 2:'ббб', 3:'ввв', 4:'ггг'}
print(len(stuff))
A. 1
B. 3
C. 2
D. 4
ANSWER: D
85. Сөздікте кілттің бар - жоғын тексеру үшін қолданылатын оператор
A. ?
B. in
C. &
D. ^
ANSWER: B
86. Төмендегі код нәтижесі
dict1 = {'key1':1, 'key2':2}
dict2 = {'key2':2, 'key1':1}
print(dict1 == dict2)
A. False
B. True
C. қате
D. 11
ANSWER: B
87. Төмендегі код нәтижесі
my_dict = {'foo': 100, 'bar': 200, 'baz': 300}
print(my_dict['bar':'baz'])
A. (200, 300)
B. [200, 300]
C. 200 300
D. қате
ANSWER: D

88. Барлық сөздік кілттерін және олардың байланысты мәндерін кортеждер тізбегі ретінде қайтаратын сөздік әдісі
A. keys()
B. items()
C. values()
D. get()
ANSWER: B
89. Сөздіктен оның кілті негізінде элементті жоюға арналған әдіс
A. erase()
B. delete()
C. remove()
D. pop()
ANSWER: D
90. Python - да тізім элементінін индексі бойынша өшіру
A. delete()
B. remove()
C. pop()
D. discard()
ANSWER: C
91. Python - да тізім элементтерін өсу реті бойынша сұрыптау
A. sort()
B. order()
C. arrange()
D. organize()
ANSWER: A
92. Python - да тізімнің соңғы элементін алу
A. last()
B. end()
C. tail()
D. [-1]
ANSWER: D
93. Төмендегі код нәтижесі
dct = {'понедельник': 1, 'вторник': 2, 'среда': 3}
print(dct.get('понедельник', 'Не найдено'))
A. 3
B. 2

C. 6
D. 1
ANSWER: D
94. Төмендегі код нәтижесі
dct = {'понедельник': 1, 'вторник': 2, 'среда': 3}
print(dct.get('пятница', 'Не найдено'))
A. 5
B. 2
C. 6
D. 4
ANSWER: C
95. Python - да екі өлшемді тізімді сипаттау
A. [[ ]]
B. [( )]
C. {{ }}
D. [{ }]
ANSWER: A
96. students сөздігінен Темурдің жасын алудың дұрыс жолын көрсетіңіз
students = {1: {'name': 'Тимур', 'age': '28', 'sex': 'Male'},
2: {'name': 'Руслан', 'age': '22', 'sex': 'Male'},
3: {'name': 'Соня', 'age': '25', 'sex': 'Female'}}
A. students[0][1]
B. students[0]['age']
C. students[1]['age']
D. students['Тимур']['age']
ANSWER: C
97. marks сөздігінен тарихты бағалауды шығарудың дұрыс нұсқасын көрсетіңіз
```
marks = {
  'class':{
    'student':{
      'name':'Rosaly',
      'marks':{
        'physics':70,
        'history':80
      }
```

```
    }
  }
}
```
A. marks['class']['Rosaly']['marks']['history']
B. marks['class'][0]['marks']['history']
C. marks['class']['student']['marks'][1]
D. marks['class']['student']['marks']['history']
ANSWER: D
98. Python - да екі өлшемді тізімнің жолдар саны
A. rows()
B. row_count()
C. len()
D. count()
ANSWER: B
99. Python - да екі өлшемді тізімнің бағаналар саны
A. columns()
B. col_count()
C. len()
D. count()
ANSWER: B
100. Python - да файлды ашуға арналған функция
A. open
B. read
C. write
D. create
ANSWER: A
101. Python - да жолдағы жолды іздеу әдісі
A. splices
B. find
C. rfind
D. replace
ANSWER: B
102. Python - да жолдағы жолды ауыстыру әдісі
A. splices
B. find
C. rfind

D. replace
ANSWER: D
103. Жолды жолдарға бөлу әдісі
A. splices
B. find
C. rfind
D. split()
ANSWER: D
104. Бірнеше жолдарды бір жолға біріктіру әдісі
A. join()
B. find()
C. rfind()
D. replace()
ANSWER: A
105. Код нәтижесі
a = 82 // 3 ** 2 % 7
print(a)
A. 2
B. 3
C. 7
D. 82
ANSWER: A
106. Код нәтижесі
a = 4
print(a, 'a')
A. 44
B. aa
C. 4a
D. a4
ANSWER: C
108. Төмендегі код нәтижесі
my_dict = {1: [0, 1], 2: [2, 3], 3: [4, 5]}
print(my_dict[2][1])
A. [0, 1], [2, 3]
B. [2, 3]
C. 3

D. 5
ANSWER: C
109. float санын = = көмегімен салыстыруға болады
A. ия
B. жоқ
C. =
D. >
ANSWER: B
110.Төмендегі код нәтижесі
```
from decimal import *
num = Decimal(0.1) + Decimal(0.1) + Decimal(0.1) - Decimal(0.3)
if num == 0:
   print('YES')
else:
   print('NO')
```
A. YES
B. YES YES
C. NO
D. NO NO
ANSWER: C
111. Төмендегі код нәтижесі
```
from decimal import *
num = Decimal('0.1') + Decimal('0.1') + Decimal('0.1') - Decimal('0.3')
if num == 0:
   print('YES')
else:
   print('NO')
```
A. YES
B. NO
C. print
D. else
ANSWER: A
112. Код нәтижесі
```
def swap(a, b):
   a, b = b, a
a = 4
```

```
b = 3
swap(a, b)
print(a - b)
```
A. -1
B. 4
C. 1
D. 3
ANSWER: C
113. Python да рұқсат етілетін файл атауы
A. find_sum_1
B. 1check_condition
C. 123check@
d. #check_condition
ANSWER: A
114. Төмендегі код нәтижесі
```
from fractions import *
num = Fraction(7, 71)
if num * 71 == 7:
   print('YES')
else:
   print('NO')
```
A. YES
B. NO
C. NO NO
D. YES YES
ANSWER: A
115. Өте тез анимацияны жүзеге асыруға арналған команда
A. turtle.speed(1)
B. turtle.speed(10)
C. turtle.speed(0)
D. turtle.speed(2)
ANSWER: B
116. readlines() Файлдың readlines() әдісі қайтаратын деректер түрі
A. жолдар тізімі
B. бүтін сандар тізімі
C. жолдар кортежі

D. жолдар
ANSWER: A
117. Мәтіндік файлдың барлық мазмұнын жол ретінде оқу үшін пайдалануға болатын файл әдісі
A. readlines()
B. read_file_to_str()
C. readline()
D. read()
ANSWER: D
118. Файлдармен жұмыс істеуді әмбебап ету үшін python бағдарламаларында файл жолдарында _____ қою ұсынылады
A. кері слеш (\)
B. тура слеш (/)
C. үтір (,)
D. нүкте (.)
ANSWER: B
119. jack_russell.png атты екілік файл орындалатын бағдарламамен бір бумада орналасқан. Бұл файлды оқу үшін ашу
A. open('jack_russell.png', 'rb')
B. open('jack_russell.png', 'r')
C. open('jack_russell.png', 'wb')
D. open('jack_russell.png')
ANSWER: A
120. Келесі функция _____ аргументтер санын қабылдайды
def func(x, y, *args):
A. екі және көп
B. бір және көп
C. үш және көп
D. бес және көп
ANSWER: A
121. Файлдың атын түрінен ажырататын белгі
A. үтір
B. нүкте
C. нүктелі үтір
D. қос нүкте
ANSWER: B

122. Төмендегі код нәтижесі
```
matrix  = [[1, 2, 8, 0],
          [-4, 1, 9, 4],
          [41, 71, 2, -2]]
print(matrix[2][3])
```
A. 9
B. -2
C .4
D. 2
E. 6
ANSWER: B
124. Файлды бағдарламада пайдаланбас бұрын, ол _____ болуы керек
A. шифрланған
B. ашық
C. жабық
D. форматталған
ANSWER: B
125. Функционалды тілдерде қайталанатын әрекеттерді жүзеге асыру
A. цикл көмегімен
B. рекурсия көмегімен
C. goto көмегімен
D. шарт көмегімен
ANSWER: B
126. Атау арқылы белгіленген және компьютердің ұзақ мерзімді жадында біртұтас тұтас күйде сақталатын ақпараттар _____ деп аталады
A. бума
B. бағдарлама
C. құжат
D. файл
ANSWER: D
127. Python - да тізім элементтерін өсу реті бойынша сұрыптау
A. sort()
B. order()
C. arrange()
D. organize()

ANSWER: A
128. Python – да тізімнің соңғы элементін алу
A. last()
B. end()
C. tail()
D. [-1]
ANSWER: D
129. Код нәтижесі
a = 4
print(a, 'a')
A .44
B. aa
C. 4a
D. a4
ANSWER: C
130. Төмендегі код нәтижесі
my_dict = {1: [0, 1], 2: [2, 3], 3: [4, 5]}
print(my_dict[2][1])
A. [0, 1], [2, 3]
B. [2, 3]
C. 3
D. 5
ANSWER: C

Қорытынды

"PYTHON тілінде бағдарламалау" оқу - әдістемелік құралы білім алушылар мен бағдарламалауды үйренушілерге Python тілінің негіздерін тиімді меңгеруге арналған кешенді құрал болып табылады. Бұл құралда Python тілінің синтаксисі, деректер типтері, басқару құрылымдары, тізімдер, кортеждер және сөздіктер, жолдармен жұмыс, функциялар, файлдармен жұмыс, қателер мен ерекше жағдайларды өңдеу жан - жақты қарастырылды. Сонымен қатар, оқу құралы практикалық тапсырмалар мен мысалдар арқылы теорияны бекітуге мүмкіндік береді.

Python тілі қазіргі заманғы бағдарламалаудың маңызды бағыттарының бірі ретінде қарастырылады, оның қарапайымдылығы мен икемділігі білім алушыларға және мамандарға бағдарламалаудың негіздерін тез әрі сапалы үйренуге жол ашады. Оқу құралының әдістемелік негіздері оқытушыларға сабақты тиімді ұйымдастыруға көмектеседі және студенттердің логикалық ойлау қабілетін дамытуға ықпал етеді.

Жалпы алғанда, бұл оқу-әдістемелік құрал Python бағдарламалау тілін меңгеруде алғашқы қадамдарды жасау үшін өте қолайлы, әрі болашақта күрделі жобаларда қолдануға қажетті білім мен дағдыларды қалыптастыруға бағытталған.

Әдебиеттер

1. Python ресми құжаты
   https://docs.python.org/3/
2. Sololearn - Python интерактивті курсы
   https://www.sololearn.com/learn/courses/python-for-beginners
3. W3Schools - Python Tutorials
   https://www.w3schools.com/python/
4. Metanit – Python (орыс тілінде)
   https://metanit.com/python/
5. Маврин П. "Программирование на Python. Легкий старт", 2022г.
6. Васильев А.Н. Программирование на PYTHON в примерах и задачах, Москва 2022г.
7. Кравченко А. "Python для начинающих: просто о важном", 2023г.
8. Иванов А. - Современный Python 3.11+, 2023г.
9. Eric Matthes Python Crash Course (3rd Edition), 2024
10. Курстар: Coursera, Udemy, YouTube

Қосымшалар

Терминдер глоссарийі

Қазақша	Орысша	English
Орнату (инсталляция)	Установка	Installation
Орнатушы файл (.exe)	Установочный файл (.exe)	Installer (.exe)
Интерпретатор	Интерпретатор	Interpreter
IDE (әзірлеу ортасы)	IDE (среда разработки)	IDE (Integrated Dev Env)
Қолданба	Приложение	Application
Консоль / Терминал	Консоль / Терминал	Console / Terminal
Скрипт	Скрипт	Script
Интерпретатор	Интерпретатор	Interpreter
Пәрмен жолы	Командная строка	Command Line / Terminal
Пайдаланушы интерфейсі	Пользовательский интерфейс	User Interface
Модуль	Модуль	Module
Жоба (проект)	Проект	Project
Файл	Файл	File
Python файлы (.py)	Python файл (.py)	Python file (.py)
Код терезесі	Окно кода	Code editor / window
Орындау	Запуск	Run
Консоль	Консоль	Console
Хабарлама (нәтиже)	Сообщение / результат	Output
Енгізу (пайдаланушыдан)	Ввод (от пользователя)	Input
Шығару (экранға шығару)	Вывод (на экран)	Output

Қазақша	Орысша	English
Айнымалы	Переменная	Variable
Жол (мәтін)	Строка	String
Мәлімет түрін түрлендіру	Преобразование типа данных	Type Conversion
Түсініктеме	Комментарий	Comment
Айнымалы	Переменная	Variable
Деректер типі	Тип данных	Data Type
Бүтін сан	Целое число	Integer (int)
Нақты сан (бөлшек)	Вещественное число	Float (float)
Жол (мәтін)	Строка	String (str)
Логикалық мән	Логическое значение	Boolean (bool)
Тізім	Список	List (list)
Сөздік	Словарь	Dictionary (dict)
Айнымалыны жариялау	Объявление переменной	Variable declaration
Шартты оператор	Условный оператор	Conditional operator
if операторы	Оператор if	if statement
else блогы	Блок else	else block
elif (басқа шарт)	elif (иначе если)	elif (else if)
Тернарлы оператор	Тернарный оператор	Ternary operator
Шарт	Условие	Condition
Шартты өрнек	Условное выражение	Conditional expression
Логикалық оператор	Логический оператор	Logical operator (and, or)
Логикалық амалдар	Логические операции	Logical operations
and - ЖӘНЕ	and - И	and - Logical AND
or - НЕМЕСЕ	or - ИЛИ	or - Logical OR

Қазақша	Орысша	English
not - ЕМЕС	not - НЕ	not - Logical NOT
Шарт	Условие	Condition
Логикалық мән	Логическое значение	Boolean value
Салыстыру операторы	Оператор сравнения	Comparison operator
Күрделі шарт	Сложное условие	Compound condition
Цикл	Цикл	Loop
while циклі	Цикл while	while loop
for циклі	Цикл for	for loop
Шарт	Условие	Condition
Итерация	Итерация	Iteration
break операторы	Оператор break	break statement
continue операторы	Оператор continue	continue statement
Цикл денесі	Тело цикла	Loop body
Шексіз цикл	Бесконечный цикл	Infinite loop
Итератор	Итератор	Iterator
Диапазон (Range)	Диапазон (Range)	Range
Тізім	Список	List
Кортеж	Кортеж	Tuple
Сөздік	Словарь	Dictionary
Тізім элементі	Элемент списка	List element
Кортеж элементі	Элемент кортежа	Tuple element
Кілт және мән	Ключ и значение	Key and value
Мәтін кілті (сөздік үшін)	Строковый ключ (для словаря)	String key (for dictionary)
Қол жеткізу	Доступ	Access
Индекстеу	Индексация	Indexing
Ішкі тізім	Вложенный список	Nested list

Қазақша	Орысша	English
Ішкі кортеж	Вложенный кортеж	Nested tuple
Тізімнің ұзындығы	Длина списка	List length
Кортеждің ұзындығы	Длина кортежа	Tuple length
Жол	Строка	String
Жолды біріктіру	Конкатенация строки	String concatenation
Жолдың ұзындығы	Длина строки	String length
Жолды бөлу	Разделение строки	String split
Жолды өңдеу	Обработка строки	String manipulation
Жолды іздеу	Поиск в строке	String search
Жолдың ішінде символ іздеу	Поиск символа в строке	Character search in string
Жолды алмастыру	Замена в строке	String replace
Жолды үлкен әріпке өзгерту	Преобразование строки в верхний регистр	Convert string to uppercase
Жолды кіші әріпке өзгерту	Преобразование строки в нижний регистр	Convert string to lowercase
Жолды тазалау	Очищение строки	String trimming
Жолды қиып алу	Обрезка строки	String slicing
Форматтау	Форматирование строки	String formatting
Жолды қосу	Добавление строки	String append
Жолды салыстыру	Сравнение строк	String comparison
Функцияны анықтау	Определение функции	Function definition
Функцияны шақыру	Вызов функции	Function call
Аргумент	Аргумент	Argument
Параметр	Параметр	Parameter
Қайтару мәні	Возвращаемое значение	Return value
Қайту операторы	Оператор возврата	Return statement

Қазақша	Орысша	English
Функцияның денесі	Тело функции	Function body
Жергілікті айнымалылар	Локальные переменные	Local variables
Глобалды айнымалылар	Глобальные переменные	Global variables
Рекурсия	Рекурсия	Recursion
Қазақша	Орысша	Ағылшынша
Файл	Файл	File
Файлды ашу	Открыть файл	Open file
Файлды жабу	Закрыть файл	Close file
Файлды оқу	Чтение файла	Read file
Файлға жазу	Запись в файл	Write to file
Файлды сақтау	Сохранить файл	Save file
Файлдың жолы	Путь к файлу	File path
Файлдың кеңейтілуі	Расширение файла	File extension
Файлдың өлшемі	Размер файла	File size
Файлды көшіру	Копирование файла	Copy file
Файлды жылжыту	Перемещение файла	Move file
Файлды жою	Удаление файла	Delete file
Файлдың атауы	Имя файла	File name
Файлдың мазмұнын оқу	Чтение содержимого файла	Read file content
Файлды ашу режимі	Режим открытия файла	File open mode
Файлды жазу режимі	Режим записи в файл	File write mode
Бинарлық файл	Бинарный файл	Binary file
Мәтіндік файл	Текстовый файл	Text file
Файлдың атын өзгерту	Переименование файла	Rename file
Қате	Ошибка	Error
Ерекше жағдай	Исключение	Exception

Қазақша	Орысша	English
Қателерді өңдеу	Обработка ошибок	Error handling
Ерекше жағдайды өңдеу	Обработка исключений	Exception handling
Қате хабарламасы	Сообщение об ошибке	Error message
Қате аумағы (try-except)	Блок try-except	try-except block
Қателерді тексеру	Проверка ошибок	Error checking
Ерекше жағдайды ұстау (catch)	Ловля исключений	Catch an exception
Қате туралы хабарлама	Сообщение об исключении	Exception message
Қате болған кезде тоқтату	Остановка при ошибке	Stop on error
Ерекше жағдайды туындату	Генерация исключения	Throw an exception