Қазақстан Республикасының Білім және ғылым министірлігі

А. Байтұрсынов атындағы Қостанай өңірлік университеті

Бағдарламалық қамтамасыз ету кафедрасы

Сатмаганбетова Ж.З
Галиханов С.Г

PYTHON-ДА БАҒДАРЛАМАЛАУ

Оқу құралы

Қостанай, 2020

УДК 004.42(075.8)
ББК 32.973-018я73
С 23

Авторлар:
Сатмаганбетова Жанар Зарлыканқызы, бағдарламалық қамтамасыз ету кафедрасының аға оқытушысы
Галиханов Сункар Габитұлы, бағдарламалық қамтамасыз ету кафедрасының оқытушысы

Пікір жазғандар:
Жунусов К.М. М. Дулатов атындағы ҚИнЭУ ақпараттық технологиялар мен автоматика кафедрасының меңгерушісі, э.ғ.к., доцент
Бермагамбетов А.К., А. Байтұрсынов атындағы ҚАУ бағдарламалық қамтамасыз ету кафедрасының аға оқытушысы, т.ғ.м.
Лата В.В., А. Байтұрсынов атындағы ҚАУ бағдарламалық қамтамасыз ету кафедрасының аға оқытушысы, т.ғ.м

С 23. Сатмаганбетова Ж. З. Галиханов С.Г.
Python-да бағдарламалау. Оқу құралы. – Қостанай: А. Байтұрсынов атындағы ҚАУ, 2020.- 64 бет

Оқу құралында Python программалау тілінің негізгі түсініктері мен ерекшеліктері, Python IDE бағдарламалау ортасында жұмыс істеу принциптері, Python IDE бағдарламалау ортасын орнату және баптау, Python тілінің программалау ерекшеліктері қарастырылған. Оқу құралы 6В06103-Ақпараттық технологиялар және робототехника білім беру бағдарламасының студенттеріне арналған.

А. Байтұрсынов атындағы ҚАУ оқу - әдістемелік кеңесімен бекітілген және ұсынылған 14.12. 2020 ж. №2 хаттама

ISBN 978-601-7640-92-7

Мазмұны

Кіріспе

Python - әр түрлі типтегі қосымшаларды жасауға арналған танымал жоғары деңгейлі бағдарламалау тілі. Python машиналық оқыту және жасанды интеллект зерттеу саласында кеңінен таралған. Python - жоғары дәрежелі кодтың оқылуын және әзірлеушінің өнімділігін арттыруға мақсатталған жалпы мақсаттағы бағдарламалау тілі. Python тілі аз синтаксисті талап етеді. Бірақ сол уақытта стандартты кітапханасы үлкен көлемді пайдалы функцияларды қамтиды.

Оқу құралының негізгі мақсаты – сіздерді Python бағдарламалау тілінің негізгі түсініктерімен таныстыру. Жалпы, Python – әртүрлі мақсаттарда қолданылатын, заманауи әрі кең таралған бағдарламалау тілі. Бұл тілді меңгеру барысында, сіз бағдарламалаудың ең негізгі принциптерімен танысасыз.

Python – объектілі-бағытталған бағдарламалау тілі. Python деректер құрылымын қамтамасыз ететін және ол түрлі қосымшалар арқылы бірнеше платформаларда жұмыс істеу үшін арналған тіл. Python – қолданбалы бағдарламалар мен веб - сценарийлерді әзірлеуге ыңғайлы тілі. Python-ді Google, Intel, Cisco және Hewlett-Packard сияқты алпауыттар пайдаланады, Python тілінде танымал YouTube сайттары - VKontakte, DropBox жұмыс жасайды[1].

«Python-да бағдарламалау» оқу құралында мына мәселер қарастырылған:

- Python программалау тілінің негізгі түсініктері мен ерекшеліктерін;
- Python IDE бағдарламалау ортасында жұмыс істеу принциптерін;
- деректер құрылымын сипаттау әдістерін;
- Python IDE бағдарламалау ортасын орнату және баптау;
- қолданушы функцияларды жасауды және қолдануды;
- Python функцияларын жүктеу және шақыру мүмкіндіктері;
- интерпретацияланған бағдарламалау тілдерімен жұмыс істеу;
- Python тілінің негізгі объектілерін дұрыс жазу;
- дайын модульдерді пайдалану және өз модульдерін жасау.

Python-ға көптеген кітапханалар жазылған. Сонымен қатар, бұл бағдарламалау тілінде өте үлкен қауымдастық бар, Интернетте сіз осы тілде көптеген пайдалы материалдар мен мысалдарды таба аласыз, мамандардан тәжірибелік көмек ала аласыз.

Бүгінгі таңда, көптеген жұмыс берушілер Python тілін меңгерген мамандарды талап етеді және бұл сферадағы мамандарға үлкен жалақы беріледі.

1 Python бағдарламалау тіліне кіріспе

1.1 Python бағдарламалау тілінің шығу тарихы

Python тілін әзірлеу голланд институтының қызметкері Гвидо ван Россуммен 1980 жылдың соңында басталған. Ол оны бос уақытында жаза бастаған. 1991 жылыдың 20 ақпанында алғашқы мәтіндерін жаңалықтар топтамасында жарыққа шығара бастады[1]. Python тілі бастапқыдан объектіге бағытталған бағдарламалау тілі ретінде жобаланды.

Автор бағдарламаны 1970-жылдардағы танымал британ комедиялық «Монти Пайтон Ұшатын циркі» телешоуының құрметіне атаған. Көбі оны жыланның атымен байланыстырады. Бағдарламаның python.org сайтында (2.5 нұсқасына дейін) жыланның басы бейнеленген. Python әзірлеушісінің негізгі мақсаты - оны қолданушыға қызықты етіп жасау болды. Оны атауынан да байқауға болады. Оның бұл мақсаты бағдарламаны үйретуді ойын түрінде ұйымдастырып, ақпараттық материалдармен жабдықтағандығында. Бұл тілге деген қолданушылардың жақсы сын пікірлерінен Гвидоның дизайнерлік құрылымының да ұтымды болғанын дәлелдейді [2].

Python 2.0 нұсқасы 16 қазан, 2000 жылы шықты, және көптеген жаңа ірі мүмкіндіктерді қамтыды, онда Unicode қолдау қолданылды.

2008 жылдың 3 желтоқсанында ұзақ тестілеуден кейін Python 3000 (немесе Python 3.0) бірінші нұсқасы шыққан. Python 3000 бағдарламалау тілінде Python ескі нұсқаларымен сәйкестікті максимум сақтауға тырыса отырып, архитектурасы бойынша кемшіліктерді жойған. Қазіргі күні (Python 3.x және 2.x) екі даму бұтақтары да қолданып келе жатыр.

Python тілінің негізгі және аралық нұсқаларының уақыты[1-3]:

- Python 1.0-Қаңтар 1994;
- Python 1.5 - 31 желтоқсан 1997;
- Python 1.6-5 қыркүйек 2000;
- Python 2.0 - 16 қазан 2000;
- Python 2.1 - 17 сәуір 2001;
- Python 2.2 - 21 желтоқсан 2001;
- Python 2.3-29 шілде, 2003;
- Python 2.4 - 30 қараша 2004;
- Python 2.5-19 қыркүйек, 2006;
- Python 2.6-1 қазан, 2008;
- Python 2.7-3 шілде, 2010;
- Python 3.0-3 желтоқсан, 2008;
- Python 3.1-27 маусым 2009;
- Python 3.2-20 ақпан, 2011;
- Python 3.3-29 қыркүйек, 2012;
- Python 3.4-16 наурыз, 2014.

Python тілінің кешірек шыққандықтан оған көптеген тілдердің ықпалы болды. Мысалы келесі тілдердің[3]:

- ABC — операторларды топтаудың шегіністері, жоғары деңгейлі деректер құрылымы. (Python тілі шындығында келегенде, ABC ОББ тілін жобалауда кеткен қателіктерді түзету үшін құрылған тіл болатын);
- Modula-3 — бумалары, модульдері;
- C, C++ — біршама синтаксистік конструкциялары;
- Smalltalk — объектіге бағытталған программалау;
- Lisp — (lambda, map, reduce, filter и другие) функционалды программалаудың айрықша белгілерін;
- Fortran —массивтер, кешенді арифметика;
- Miranda — тізімдік өрнектер;
- Java — logging, unittest, threading модульдерін, xml.sax стандартты библиотекасын, finally және except-тің ескерпелерді өңдеудегі біріктірілген қолданылуын;
- Icon — генераторларын.

Python тілінің басым бөлігі (мысалы, бастапқы кодтың байт-компиляциясы) бұрынырақта басқа бағдарламалау тілдерінде іске асырылатын.

Python – бүкіл әлем бойынша түрлі мақсаттар -деректер базасын және табиғи тілде мәтінді өңдеу үшін кең таралған әмбебап тіл, ойындарға интерпретатор қосу, GUI-ді бағдарламалау және жылдам прототип құру (RAD) үшін арналған тіл. Python - Internet және WEB-қосымшаларын бағдарламалау үшін пайдаланылады. Python бай стандартты кітапханадан және модульдер жиынтығынан тұрады.

Python бағдарламалау тілінің негізгі ерекшеліктері[4]:

- xml/html файлдарымен жұмыс жасау;
- http сұраныстар жасау;
- GUI (графикалық интерфейс)
- Веб-сценарийлер құру;
- FTP-мен жұмыс жасау;
- Кескіндермен, аудио және видеомен жұмыс жасау;
- Робототехникада қолдану;
- Математикалық және ғылыми есептеулерді бағдарламалау және т.б.

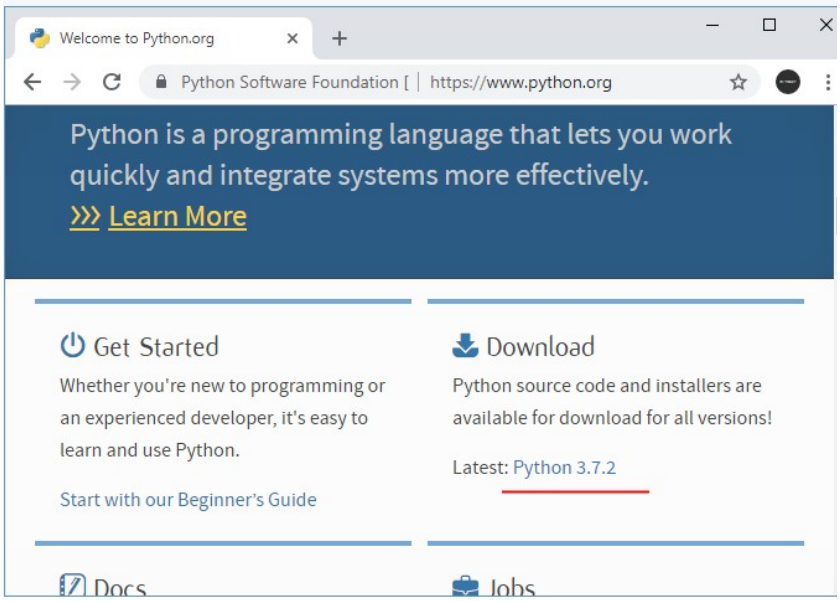
Python белгілі барлық платформаларда жұмыс істейді. Microsoft Windows порттары үшін барлық нұсқаларында (FreeBSD және Linux қоса алғанда) UNIX, Mac OS және Mac OS X, Iphone OS 2.0 немесе одан жоғары Amiga, HaikuOS, Windows Mobile, Symbian және Android[3]. Python кроссплатформалық технологияларда қолдауында.

Python бағдарламасы көптеген міндеттерді шешеді: резервті көшіру болсын, электронды поштаны оқу болсын немес қандай да ойынды құру болсын. Python бағдарламалау тілі ештеңемен шектелмегендіктен оны үлкен жобаларды пайдалануға болады.

1.2 Python-ды және IDE-ні орнату

Python бағдарламасын жүктеу тегін және еш тіркелусіз орындалады.

Алдымен оны ресми сайттан жүктеп алу керек. Python бағдарламаларын жүктеу үшін аудармашы қажет. Оны орнату үшін https://www.python.org/ сайтына өтіңіз және Жүктеулер бөліміндегі басты бетте тілдің соңғы нұсқасын жүктеуге сілтеме табамыз (Сурет 1).



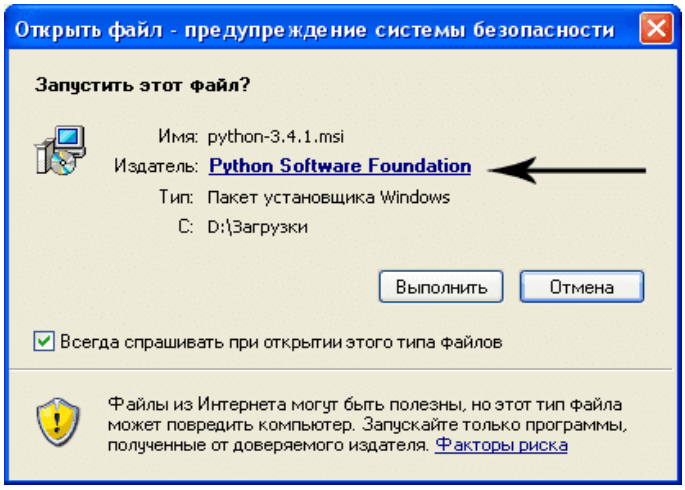
Сурет 1 - Python ресми сайты

Тілдің соңғы нұсқасының сипаттамасымен бетке сілтеме жасайық. Онда төменгі жаққа қарай әртүрлі операциялық жүйелер үшін бөлімдердің тізімін табуға болады. Қажет буманы таңдап, жүктеп алыңыз. Мысалы, менің жағдайда, бұл Windows 64-bit OS, сондықтан мен Windows x86-64 орындалатын орнатушы бумасына сілтемені таңдаймын. Жүктеп алғаннан кейін оны тарату орнатылады.

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		02a75015f7cd845e27b85192bb0ca4cb	22897802	SIG
XZ compressed source tarball	Source release		df6ec36011808205beda239c72f947cb	17042320	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	d8ff07973bc9c009de80c269fd7efcca	34405674	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	0fc95e9f6d6b4881f3b499da338a9a80	27766090	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	f81568590bef56e5997e63b434664d58	7025085	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	ff258093f0b3953c886192dec9f52763	26140976	SIG
Windows x86-64 web-based	Windows	for AMD64/EM64T/x64	8de2335249d84fe1eeb61ec25858bd82	1362888	SIG

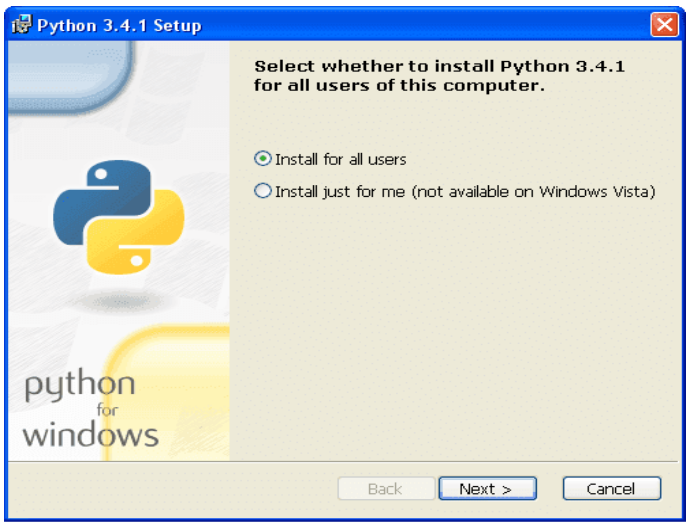
Сурет 2 - Жүктеуге болатын файлдар тізімі

Python тілі жүктелгенше күтеміз. Жүктелген файлды ашу керек. Шығарушы қатарында Python Software Foundation жазуын көрсеңіздер, онда дұрыс таңдалды. Басқа жазу тұрса, ондай файлды ашпаған жөн.



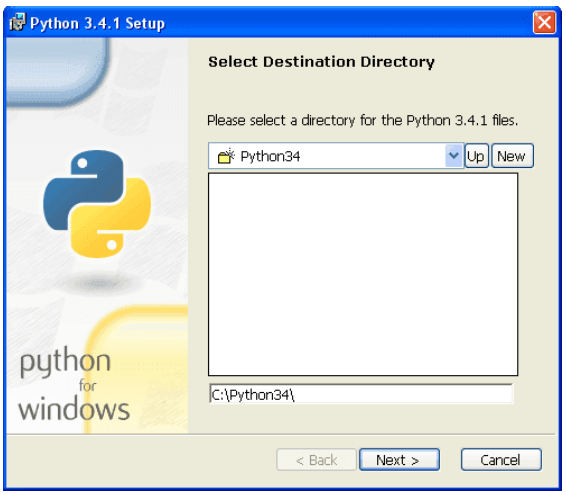
Сурет 3 - Хабарлама терезесі

Барлық қолданушыларға немесе тек бір қолданушыға орнатуды орындаймыз (өз қалауларыңызбен орнату керек).



Сурет 4 - Орнату жағдайын таңдау терезесі

Орнатуға арналған буманы таңдау керек. Дискіден кез келген буманы таңдауға болады.



Сурет 5 - Орнататын буманы таңдау терезесі

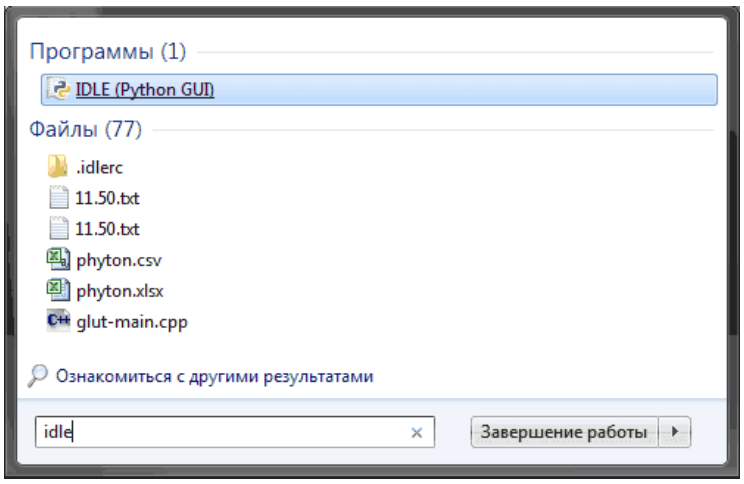
Компонентерін таңдауға болады. Егер таңдау білмеген жағдайда үнсіз келісім бойынша қалдыру керек.

Сурет 6 - Компоненттерді таңдау терезесі

Python бағдарламасының орнатылғанын күту керек. Finish батырмасын басқан соң бағдарлама орнатылды деп есептеу керек. Бұл нұсқада IDLE «зірлеу ортасы ендірілген. Енді алғашқа бағдарламаны кез келген мәтіндік редакторда немесе IDLE әзірлеу ортасында жазуға болады.

Бағдарламалау ортасы ретінде IDLE python 3.4 GUI қолдануға да болады(басқа әзірлеу ортасы да қабылдануы мүмкін).

Python бағдарламасын іске қосқан соң IDLE ортасын ашу керек (Python бағдарламалау ортасы).

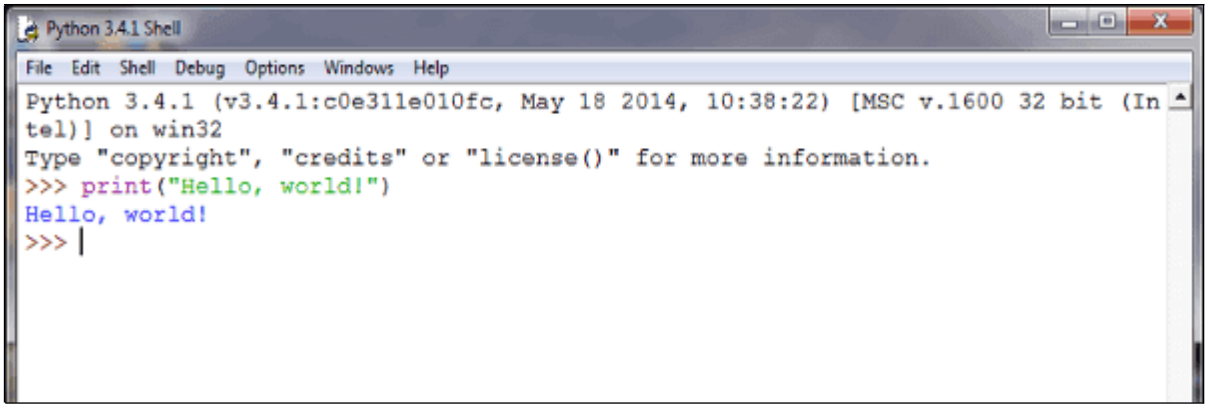


Сурет 7 - IDLE ортасын іске қосу терезесі

IDLE ортасы бастапқыда интерактивті режімде ашылады. Кейіннен программаны жазуды бастауға болады. Python тілінде "hello world" сөз тіркесін жазу үшін тек бір ғана жолдың жазылуы жеткілікті:

```
print("Hello world!")
```

Бұл кодты IDLE ортасына енгізіп Enter батырмасына шертеміз.Нәтиже келесі суретте бейнеленген:



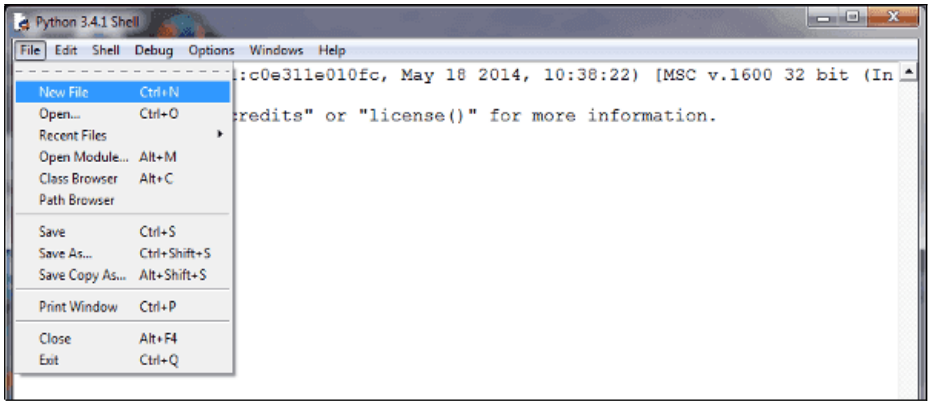
Сурет 8 - IDLE python

IDLE – дің негізгі беті интерпретатордың беті болып табылады. Ол интерактивті тәртіпте бағдарламалауға мүмкіндік береді, яғни команданы енгізгеннен кейін ол бірден [Enter] тетігін басқан соң жүзеге асырылады.

Әрбір таза жол >>> таңбасынан басталады, орындалудың нәтижесі бірден оператор жолының астында көрсетіледі. Толық жұмыс істеу үшін және нәтижелерді файлға сақтау үшін File мәзірінен NewFile тармағын таңдаңыз. Содан кейін ашылған бетте бағдарлама кодын таңдауға болады.

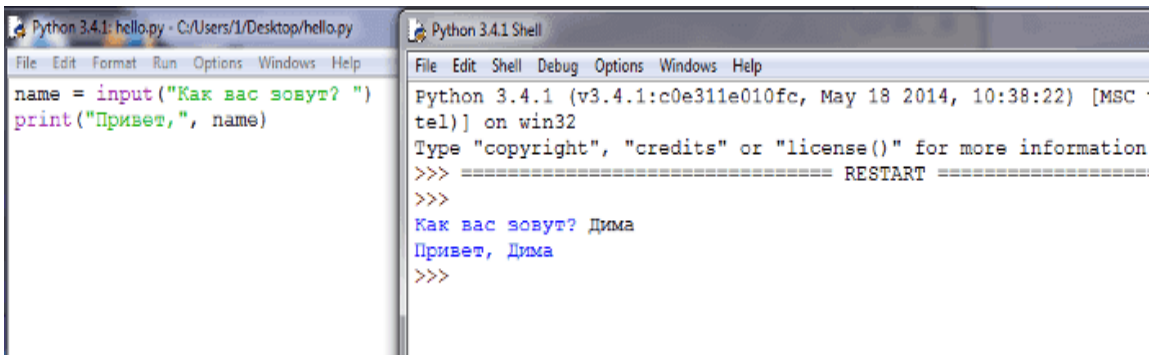
Негізінде интерактивті режім ең негізгісі болып табылмайды, сондықтан көбінде бағдарламалық кодтарды файлға жазып, файлды тексеруге жіберіп

отырып жұмыс жаслады. IDLE интерактивті режімінде жаңа файл құру үшін File → New File (немесе Ctrl + N пернелер комбинациясын басу керек).



Сурет 9 - Жаңа файл құру терезесі

Функционалды пернелерден F5 (немесе мәзірден IDLE Run → Run Module командасын таңдап) басу және шыққан нәтиженің дұрыстығын тексеру керек. Келесі суретте бейнеленген скриншотта оң бағанында нәтиже, сол жақ бағанында жазған бағдарлама көрсетіледі.



Сурет 10 - Бағдарламаның нәтижесін шығару терезесі

Python интерпретаторы онда интерактивті режим, дереу орындалады бар, және нәтижесі (REPL) көрсетіледі. Бұл режимді бағдарлама оны пайдаланар алдында интерактивті кез келген аймақ кодын тексеру, немесе жай ғана функцияларын үлкен жиынтығы бар калькулятор ретінде пайдалануға болады деп есептейді тәжірибелі бағдарламашылар.

Интерактивті режимде Python-дағы байланыс осылай көрсетіледі:

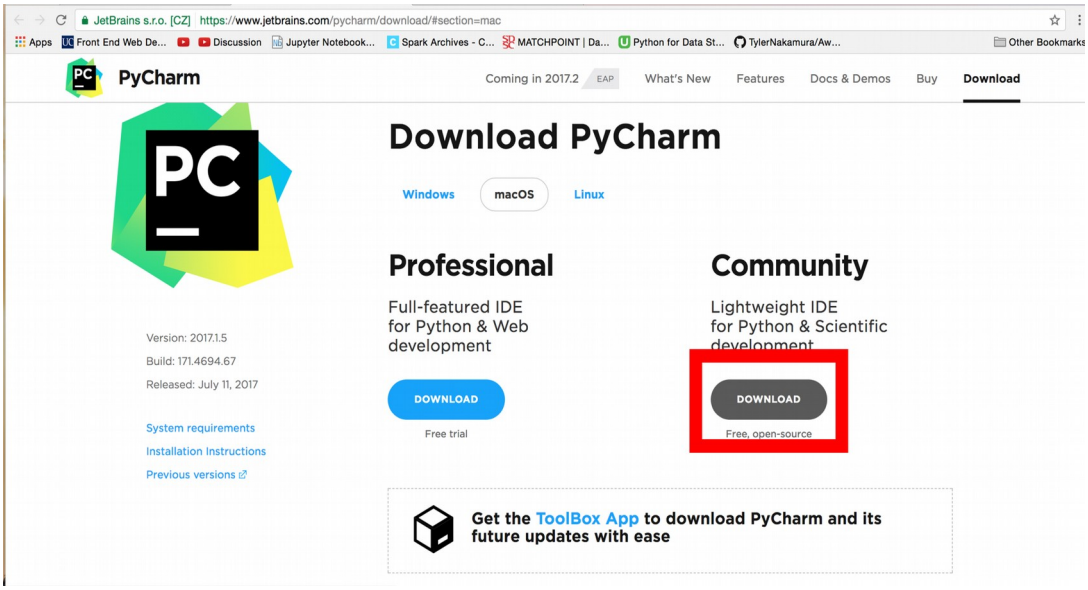
```
>>> 2 ** 100           # дәреже
1267650600228229401496703205376L
>>> from math import *
>>> sin(pi * 0.5)
>>> help(sorted)
Help on built-infunction sorted in module __builtin__:
sorted(...)
sorted(iterable, cmp=None, key=None,reverse=False) --> new sorted list
```

Интерактивті режимде PDB() (көмек үшін деп аталатын) көмек жүйесі қол жетімді. Анықтамалық жүйесі олар құжаттама жолдың қамтамасыз етілді, тек егер модульдермен функциялар үшін жұмыс істейді.

IDE-ні орнату

Интеграцияланған өңдеу ортасы (ағылш. Integrated Development Environment) - бағдарламалық қамтамасыз етуді әзірлеу үшін бағдарламашылар пайдаланатын бағдарламалық құралдар жүйесі. Бұл бағдарлама арқылы сіз бағдарламалаумен ыңғайлы түрде айналысып, жаңа мүмкіндіктерге ие боласыз.

Python тіліне арналған көптеген сапалы IDE-лер бар, соның ішінде "PyCharm" программасын ұсынамыз. Бұл IDE қолданыста ыңғайлы әрі тиімді болып табылады. PyCharm-ды орнату үшін: https://www.jetbrains.com/ru-ru/pycharm/ нұсқауына өтіп, "Community" сөзінің астындағы "Download" батырмасын шертіңіз. Сонда программа автоматты түрде жүктеледі. PyCharm-нің Community нұсқасы тегін болып табылады.



Сурет 11 - PyCharm-ды жүктеу

PyCharm-да код жазуды бастау үшін, "Create New Project" батырмасын шертіп, проектке ат беріңіз. Одан кейін тінтуірдің он батырмасын шертіп, New -> Python File батырмаларын басыңыз. Python файлына ат беріп, "OK" батырмасын басыңыз. Міне, енді сіз PyCharm-да кодты тере аласыз! Жазылған кодты жүзеге асыру үшін, үстіндегі жасыл үшбұрышты шертіңіз.

Python бағдарламасын іске қосу тәсілдері.

Алдымен мәтіндік редакторда берілген бағдарламалау тіліндегі өрнек жиынтығы бар сценарий жазамыз. Біз осы сценарийді аудармашыға жібереміз. Аудармашы кодты аралық байт кодқа аударады, содан кейін виртуалды машина алынған кодты операциялық жүйе арқылы орындалатын нұсқаулар жиынына аударады(Сурет 12).

Сурет 12 - Python бағдарламасының орындалу процессі

Бұл жерде формальді кодты аудармашы арқылы байт кодына аудару және виртуалды машина бойынша байтекодты машиналық нұсқаулар жиынтығына аудару екі түрлі процесті білдіреді, бірақ іс жүзінде олар аудармашыда біріктіріледі. Тасымалдау және платформалық тәуелсіздік. Windows, Mac OS, Linux сияқты операциялық жүйе маңызды емес, бізде барлық операциялық жүйелерде интерпретатормен жұмыс істейтін сценарий жазу керек. Автоматты жады басқару. Динамикалық теру.

Python және C++ тілінде жазылған бір бағдарлама кодының фрагменттерін салыстырайық. Нәтижесі Python - да жазылған кодтың қаншалықты аз екенін көрсетеді:

Python

```
1    print("Hello, World!")
```

C++

```
1    #include
2    void main()
3    {
4    cout << "Hello, World!" << endl;
5    }
```

1.3 Негізгі операторлар мен амалдар

Тағайындау операторы

Тағайындау операторы = таңбасы болып табылады. Оператор стандартты түрде орындалады: алдымен теңдік белгісінің оң жағындағы өрнектер есептеледі, содан кейін алынған мән теңдік белгісінің сол жағында көрсетілген айнымалыға жазылады.

```
A = 3.14
print(type(A)) # float
A = 'Hello'
print(type(A)) # str
a = b = c = 0
a += 1     # a = a + 1
c = 5//2   # int
d = 5/2    # float
b = c**2   # b = c^2 (дәрежесі)
a, b = b, a # мәндер алмасу a=b, b=a
```

Python-дағы түсініктемелер

```
# Бір жолдық түсініктеме # -дан басталады

'''
Сіз осындай блоктық түсініктеме пайдалана аласыз
(жалпы, Python-да блоктық түсініктеме жоқ
'''
```

Python тілінің синтаксисі. Негізгі ережелері[2]:

- Жолдың соңы интрукцияның соңы болып табылады (нүктелі үір қажет емес);
- Ендірілген инструкциялар шегіністер өлшемі (көлемі) бойынша блоктарға біріктіріледі. Шегініс кезкелген болуы мүмкін, бастысы бір ендірілген блокқа бірдей шегініс өлшемі қолданылса болғаны. Кодтың оқылымы туралы да ұмытпау керек. 1 бос орын шегініс - ол ешқандай шешім болып табылмайды. Тым болмағанда 4 бос орын шегініс немесе табуляция белгісін қолданған дұрыс болады.
- Python тілінде әр инструкциялар бір шаблонға сәйкес жазылады, негізгі инструкция қос нүктемен аяқталған соң, оның соңынан инструкцияның ендірілген блогы шегініс арқылы орналасады. Келесі суретте жазылуы бейнеленген:

Бірнеше арнайы жағдайлар бар. Олар:

1. Кейбір жағдайда бірнеше инструкцияны бір жолға нүктелі үтір арқылы жазуға болады:

```
a = 1; b = 2; print(a, b)
```

Бұндайды көп қолдануға болмайды, әрқашан да оқылым туралы есте сақтау керек.

2. Бір инструкцияны бірнеше жолға да бөліп жазуға болады. Ол үшін бірнеше доғал, квардатты немесе жүйелі жақшаны қолдану керек.

```
if (a == 1 and b == 2 and
    c == 3 and d == 4):
    print('spam' * 3)
```

3. Құрамдас интсрукцияның денесі сол негізгі дененің жазылған жолына орналасуы мүмкін, егер негізгі инструкцияның денесі құрамдас денені қамтымаса. Мысалы:

```
if x > y: print(x)
```

Бағдарламалау тілінің толық синтаксисін түсіну үшін көптеген мысалдарды қарастыру керек.

Деректерді енгізу және шығару

Деректерді шығару print операторының көмегімен жүзеге асырылады:

```
1    a = 1
2    b = 2
3    print(a)
4    print(a + b)
5    print('сумма = ', a + b)
```

Нұсқауларды бір жолға жазу мүмкіндігі бар. Ол үшін оларды ; арқылы бөлу керек. Алайда, мұндай әдісті жиі қолдануға болмайды, бұл оқу барысында ыңғайсыздықтар тударады:

```
1    a = 1; b = 2; print(a)
2    print(a + b)
3    print('сумма = ', a + b)
```

Print функциясы үшін шығару элементтері арасында сепаратор – айырғыш берілуі мүмкін:

```
1    x=2
2    y=5
3    print ( x, "+", y, "=", x+y, sep = " " )
```

Нәтижесі элементтер арасында бос орын қалдырылып көрсетіледі: 2 + 5 = 7

Өзгертіліп шығару үшін format қолданылады:

```
1    x = 11
2    print ( "{:4d}".format(x) )
```

Нәтижесінде 11 саны шығарылады, ал оның алдында екі бос орын болады, себебі шығару үшін төрт белгі орнын қолдану керектігі көрсетілген.

Немесе бірнеше аргументтермен:

```
1    x = 2
2    print ( "{:4d}{:4d}{:4d}".format (x, x+x, x*x) )
```

Нәтижесінде әр мән белгісі 4 белгі орны есебінен шығарылады.

Деректерді енгізу input операторының көмегімен жүзеге асырылады:

```
1    a = input()
2    print(a)
```

Функцияның жақшасында енгізілген деректерге пікір-хабарлама көрсетуге болады:

```
a = input ("Введите количество: ")
```

Input функциясы енгізу деректерін таңбалар ретінде қабылдайды. Сондықтан, бүтін мәнді қабылдау үшін int() функциясын пайдалану керек:

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

Арифметикалық амалдар

```
x + y     Қосу
x - y     Азайту
x * y     Көбейту
x / y     Бөлу
x // y    Бүтін бөлу
x % y     Бүтін бөлуден қалған қалдық
x**y      Дәрежеге шығару
  -x      Сан белгісін ауыстыру
```

Python-да әртүрлі математикалық операциялар бар. Бұл сабақта біз ең негізгі амалдарды қарастырамыз (қосу, азайту, көбейту және бөлу). Ең алдымен, қосу мен азайту амалдарын қарастырайық. Қосу мен азайту амалдарын орындау үшін, біз тиісінше + пен - амалдарын қолданамыз. Мысалы,

```
print(2 + 2)
print(10 - 6)
```

Көріп тұрғаныңыздай, бұл операторлар орындалуда қарапайым болып табылады. Енді, көбейту мен бөлу амалдарын қарастырайық. Python-да көбейту * символы арқылы белгіленеді, ал бөлу / символы арқылы белгіленеді.

Мысалы,

```
print(15 * 2),
print(6 / 3).
```

Python-да осы негізгі амалдардан басқа, тағы бірнеше амалдар бар. Python-дағы барлық математикалық амалдар математикалық заңдарға толық бағынады. Мысалы, жақшалар амалдың қай бөлігін бірінші кезекте орындау керектігін көрсету үшін қолданылады. Python әрқашан ең алдымен жақшаның ішіндегі амалдарды орындайды, тек одан соң келесі амалдарға көшеді.

Мысалы,

```
print((5 + 5) * 2)
print(3 + (4 * 5))
print(10 / (6 - 1)).
```

Осыған қоса, біз сандық мән берілген айнымалылармен математикалық операцияларды орындай аламыз. Бірақ, мәтінді айнымалыларды біз бір-біріне тек қоса аламыз. Бұл түсінік – конкатенация деп аталады.

Биттік операциялар. Бүтін сандармен де биттік операцияларды орындауға болады.

Биттік операциялар тізімі

x | y	Биттік немесе
x ^ y	Биттік «алып тастағыш» немесе
x & y	Биттік және
x << n	Солға биттік жылжу
x >> y	Оңға биттік жылжу
~x	Биттер инверсиясы

Қосымша әдістер. int.bit_length() - белгісі мен жетекші нөлдерін алып тастағандағы екілік жүйеде санды беру үшін қажетті биттер саны. Төменде көрсетілгендей:

```
>>> n = -37
>>> bin(n)                  //     '-0b100101'
>>> n.bit_length()          // 6
```

int.to_bytes(length, byteorder, *, signed=False) – осы санды беретін байттар жолын қайтарады.

```
>>> (1024).to_bytes(2, byteorder='big')    // b'\x04\x00'
>>> (1024).to_bytes(10, byteorder='big')
b'\x00\x00\x00\x00\x00\x00\x00\x00\x04\x00'
>>> (-1024).to_bytes(10, byteorder='big', signed=True)
b'\xff\xff\xff\xff\xff\xff\xff\xff\xfc\x00'
>>> x = 1000
>>> x.to_bytes((x.bit_length() // 8) + 1, byteorder='little')
b'\xe8\x03'
```

Келесі classmethod int.from_bytes(bytes, byteorder, *, signed=False) әдісі аталған байттар жолынан сандарды қайтарады.

```
>>> int.from_bytes(b'\x00\x10', byteorder='big')                  //  16
>>> int.from_bytes(b'\x00\x10', byteorder='little')               // 4096
>>> int.from_bytes(b'\xfc\x00', byteorder='big', signed=True)     // -1024
>>> int.from_bytes(b'\xfc\x00', byteorder='big', signed=False)    // 64512
>>> int.from_bytes([255, 0, 0], byteorder='big')                  //  16711680
```

Санау жүйелері. Сандар тек ғана ондық санау жүйесінде ғана емес сонымен қатар, басқа да санау жүйелерінде беріледі. Мысалы, компьютерде екілік санау жүйесі қолданылады. 19 саны екілік санау жүйесінде 10011 деп бейнеленеді. Кейбір жағдайда бір санау жүйесінен екіншіге көшу қажет болады. Оны орындауға Python тілі бірнеше функцияларды ұсынады:

- int([object], [санау жүйесінің негізі]) – ондық санау жүйесіндегі бүтінге айналдыру. Бұл жерде үнсіз келісім бойынша ондық санау жүйесі қолданылады, бірақ негізін 2-ден 36-ға дейін таңдай отырып кез келген санау жүйесінде беруге болады.
- bin(x) – бүтін санды екілік жолға айналдыру
- hex(x) - бүтін санды он алтылық жолға айналдыру
- oct(x) - бүтін санды сегіздік жолға айналдыру

Санау жүйесіне мысалдар

```
>>> a = int('19') # жоды санаға айналдыру
>>> b = int('19.5')  # жол бүтін сан болып табылмайды
Traceback (most recent call last):
  File "", line 1, in
ValueError: invalid literal for int() with base 10: '19.5'
>>> c = int(19.5)  # жылжымалы нүктесі бар санға қолданылғанда бөлшек
бөлігін алып тастайды
>>> print(a, c)
19 19
>>> bin(19)
'0b10011'
>>> oct(19)
'0o23'
>>> hex(19)
'0x13'
>>> 0b10011  # сандық тұрақтылықарды ослай да жазуға болады.
19
>>> int('10011', 2)
19
>>> int('0b10011', 2)
19
```

Нақты сандар (float). Нақты сандар да бүтін сандардағы сықты операцияларды қолдайды, бірақ сандарды компьютерде бергендіктен нақты сандар нақты болмауы және қателіктерге соқтыруы мүмкін және оның өзі қателіктерге әкелуі мүмкін.

```
>>> 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1
0.9999999999999999
```

Жоғары нақтылыққа қол жеткізу үшін басқа да объектілерді (Мысалы: Decimal және Fraction) қолдануға болады.

Сонымен қатар, нақты типтер ұзақ арифметиканы қолдамайды.

```
>>> a = 3 ** 1000
>>> a + 0.1
Traceback (most recent call last):
```

File "", line 1, in
OverflowError: int too large to convert to float
Сандармен қарапайым мысалдар:

```
>>> c = 150
>>> d = 12.9
>>> c + d
162.9
>>> p = abs(d - c)  # сан модулі
>>> print(p)
137.1
>>> round(p)  # Дөңгелектеу
137
```

Қосымша әдістері:

- float.as_integer_ratio() - қатынасы осы санға тең болатын жұп сандар.
- float.is_integer() - мән бүтін сан бола ма.
- float.hex() – float-ты hex-ке түрлендіреді (он алтылық жүйеде).
- classmethod float.fromhex(s) - он алтылық жолдағы float.

```
>>> (10.5).hex()
'0x1.5000000000000p+3'
>>> float.fromhex('0x1.5000000000000p+3')
10.5
```

Python тілінде сандармен жұмысқа қатысты стандартты өрнектерден басқа бірнеше қажетті модульдер бар. Келесі math модулі күрделі математикалық функцияларды қамтамасыз етеді. Оның қолданылуы төмендегідей.

```
>>> import math
>>> math.pi          // 3.141592653589793
>>> math.sqrt(85)    // 9.219544457292887
```

Келесі random модулі - кездейсоқ сандар генераторы мен кездейсоқ функцияны таңдауды жүзеге асырады.

```
>>> import random
>>> random.random()          // 0.15651968855132303
```

Комплексті сандар (complex). Python тіліне сонымен қатар, комплекті сандар енгізілген.

Комплекс сандардың программада қолданылуы көрсетілген.

```
>>> x = complex(1, 2)
>>> print(x)
(1+2j)
>>> y = complex(3, 4)
>>> print(y)
(3+4j)
>>> z = x + y
>>> print(x)
(1+2j)
>>> print(z)
(4+6j)
```

```
>>> z = x * y
>>> print(z)
(-5+10j)
>>> z = x / y
>>> print(z)
(0.44+0.08j)
>>> print(x.conjugate())  # түйіндес сан
(1-2j)
>>> print(x.imag)  # жорамал бөлігі
2.0
>>> print(x.real)  # нақты бөлігі
1.0
>>> print(x > y)  # Комплексті санды салыстыруға болмайды
Traceback (most recent call last):
  File "", line 1, in
TypeError: unorderable types: complex() > complex()
>>> print(x == y)  # бірақ теңдікке тексеруге болады
False
>>> abs(3 + 4j)  # комплексті сан модулі
5.0
>>> pow(3 + 4j, 2)  # санның дәрежесін шығару
(-7+24j)
```

Комплексті сандармен жұмыс жасау үшін сонымен қатар, cmath модулі де қолданылады.

1.4 Деректер типтері

Python-да деректердің бірнеше типі бар. Біз әр типті бөлек қарастырамыз. Бірақ, көп жағдайда бір түрмен жұмыс істеу жеткіліксіз, себебі күрделі жобаларда деректердің бір түрінен басқаларына ауысу қажеттілігі пайда болады.

Ең алдымен, сандарды қарастырайық. Python-да сандардың екі типі бар: “Integer” – бүтін сан, және “Float” – бөлшек сандар (нүктемен жазылады). Бүтін сандар арқылы, мысалы адам санын белгілеуге болады, ал бөлшек сандар арқылы, салмақты белгілеуге болады.

Бүтін сандарға мысалдар келтірсек: people = 150, num = 1, population = 2000000. Бөлшек сандарға мысалдар келтірсек: weight = 50.5, float_num = 1.5, gpa = 4.9 .

Келесі тип – мәтінді жолдар. Python-да дара немесе қос тырнақшаға алынған кез-келген мәтін үзіндісі мәтінді жол болып табылады. Мәтінді жолдар әріптерден, символдардан, цифрлардан, бос орындардан, нүкте және үтір белгілерінен тұра алады. Мәтінді жолдарға мысалдар келтірсек: a = “Coding is fun!”, string = ‘String’, letters = ‘abc’, nums = ‘123’, symbol = ‘a’. Осындай айнымалыларды құрғанда, мәтінді жолдарды әрқашан тырнақшаларға алуға ұмытпаңыз!

Келесі деректер типі – логикалық деректер типі немесе Буль айнымалылары. Осы айнымалылардың мәндері әрқашан не ақиқат (True), не жалған (False) болады. Екі сөз де бас әріппен жазылады.

Деректер типін анықтау үшін type() функциясы қолданылады. Мысалы, print(type(24)) командасын терсек, экранға мәні шығады (int – integer, яғни 24 – бүтін сан). Осы функция арқылы басқа деректердің типтерін анықтап көріңіздер.

Python-да деректердің әртүрлі типтерін бірге араластыруға да болады. Егер біз код жазуда әртүрлі деректер типтерін түрлендірмей, бірге қолдансақ, қате пайда болады. Мысалы, мәтінді жол мен бүтін санды араластырып көрейік. Apple = ‘3’ және num = 1 айнымалыларын алайық. Оларды бірге араластыру үшін келесі команданы жазу керек: print(int(Apple) + num). Яғни, бұл жағдайда біз int() функциясы арқылы ‘3’ мәтінді жолын 3 санына айналдырдық.

Ал, сандарды мәтінді жолдарға айналдыру үшін, str() функциясын қолдану қажет. Мысалы, print(str(1) + ‘2’). Бұл жағдайда, экранға 12 мәні шығады, себебі мәтінді жолдар бір-біріне сандар сияқты қосылмай, оңай тілмен айтқанда, бір-біріне кілейленеді. Яғни, бұл жағдайда, ‘1’ мен ‘2’ бір-біріне әріптер сияқты қосылып кетті. Одан басқа, Python-да float() (бөлшек сандарға айналдыру үшін) және bool() (сандарды True немесе False айнымалыларына айналдыру үшін, 0 – False, 1 мен қалған сандар – True) функциялары бар.

Айнымалылар

Бағдарламалауда біз айнымалыны қорапша ретінде қарастыра аламыз. Бұл қорапшаның ішіне біз әртүрлі мәндерді енгізе аламыз. Мәндер ретінде бүтін немесе бөлшек сандар, мәтінді жолдар, символдар және т.б. объектілер бола алады (деректер типі келесі сабақта қарастырылады). Яғни, айнымалылар программадағы ақпараттың орналасу орнын анықтау үшін қолданылады.

Айнымалыны құру үшін, ең алдымен оның атын жазу керек. Одан кейін, тең белгісін қойып, айнымалыға мән беру қажет. Айнымалының мәні ретінде сандар, әріптер, сөздер және т.б. объектілер бола алады. Егер айнымалының мәні "None" сөзі болса, айнымалы бос болып есептелінеді.

Айнымалының аты тек қана сандардан, әріптерден және сызық таңбаларынан тұра алады. Және де, айнымалының аты сандардан бастала алмайды. Дұрыс айнымалыларға мысал келтірсек:

a = 5, string = "hello"

num = 10 boolean = True

mans_name = "Alex"

Бұрыс айнымалыларға мысал келтірсек:

333num = 3

$money = 1000

discount% = 10

Айнымалының мәнін экранға шығару үшін, print(айнымалының аты) командасын теру қажет.

Типтерді түрлендіру

Питонда типтерді түрлендіру қалай қолданылатындығының мысалдарын қарастырайық:

```
1    a = 1.7
2    a=str(a)    #таңбалы жолға түрлендіру
3    print(a)    # '1.7'
```

```
1   x = 1.7
2   x=int(x) # бүтінге түрлендіру
3   print(x) 1
```

```
1   y=1
2   y=float(y) # бөлшекке түрлендіру
3   print(y)   # 1.0
```

1.5 Python негізгі стандартты модульдері

Модульдік тәсілге сәйкес бағдарламалау үлкен міндет бірнеше ұсақ болып бөлінеді, олардың әрқайсысын (Идеалда) жеке модуль шешеді. Әр түрлі әдістемелерде модуль өлшеміне әртүрлі шектеулер беріледі, алайда бағдарламаның модульдік құрылымын құру кезінде модульдердің композициясын құру маңызды, ол олардың арасындағы байланысты барынша азайтуға мүмкіндік береді.

Өз элементтері арасында көптеген байланыстары бар сыныптар мен функциялар жиынтығы бір модульде қисынды орналастыру болар еді. Тағы бір пайдалы ескерту бар: модульдерді қайта жазудан гөрі пайдалану оңай болуы керек. Бұл дегеніміз, модуль ыңғайлы интерфейс болуы керек: функциялар жиынтығы, сыныптар және тұрақты, ол өз пайдаланушыларына ұсынады.

Python тілінде бір мәселеге арналған модульдер жиынтығын пакетке қоюға болады. Мұндай пакеттің жақсы мысалы-XML пакеті, онда XML өңдеудің әр түрлі аспектілеріне арналған модульдер жинақталған[3,9].

Python бағдарламасында модуль Модуль-нысан-модуль, Оның атрибуттары модульде анықталған атаулар болып табылады:

```
>>> import datetime
>>> d1 = datetime.date(2020, 11, 20)
```

Бұл мысалда datetime модулі импортталады. Import операторының жұмысы нәтижесінде ағымдағы атау кеңістігінде datetime атымен объект пайда болады. Python бағдарламасына модульді қосу import операторының көмегімен жүзеге асырылады . Оның екі нысаны бар : import және from-import:

```
import os
import pre as re
from sys import argv, environ
from string import *
```

Бірінші пішін көмегімен ағымдағы көріну аймағымен тек модуль объектісіне сілтеме жасайтын атау ғана байланысады, ал екіншісін пайдаланған кезде модуль объектілерінің көрсетілген аттары (немесе қолданылса, барлық аттары *) ағымдағы көріну аймағымен байланысады. Импорттау кезінде, as арқылы элементтің атын өзгертуге болады. Бірінші жағдайда модуль атаулары кеңістігі бөлек атауда қалады және модульден нақты атауына кіру үшін нүктені қолдану қажет. Екінші жағдайда аттар ағымдағы модульде анықталғандай қолданылады:

```
os.system("dir")
digits = re.compile("\d+")
print argv[0], environ
```

Модульді қайта импорттау әлдеқайда жылдам жүреді , себебі Модульдер интерпретатормен кэштеледі. Жүктелген модульді reload функциясы арқылы тағы да жүктеуге болады (мысалы, модуль дискіде өзгерсе)():

```
import mymodule
. . .
reload(mymodule)
```

Бірақ бұл жағдайда модульдің ескі нұсқасынан сыныптардың даналары болып табылатын барлық объектілер өз мінез-құлқын өзгертпейді.

Python-да *математикалық модуль* math деп аталады, және оны қосу үшін, import math командасын теру қажет.

Python ішіндегі кіріктірілген математикалық модуль математикалық, тригонометриялық және логарифмдік операцияларды орындау үшін бірқатар функцияларды ұсынады. Модульдің кейбір негізгі функциялары[3,5,9]:

- pow (num, pow): санды нөмірді қуат қуатын көтеру
- sqrt (num): санның шаршы түбірі
- ceil (num): санды ең жақын жоғарғы бүтін санға дейін дөңгелектеу
- floor (num): санды ең жақын ең кіші бүтін санға дейін дөңгелектеңіз
- factorial (num): факторлық сандар
- degrees (rad): радианнан градусқа дейін айналдыру
- radians (grad): дәрежелерден радианға дейін өзгеру
- cos (rad): радионың бұрышын косинус
- sin (rad): радианның бұрыштық синусасы
- tan (рад): радианға бұрыштық тангенс
- acos (rad): радиандықтардың бұрышының доғаның косинасы
- asin (rad): радианның бұрышының арксионы
- atan (rad): радианға бұрыштың арктангені
- log (n, base): база негізіндегі n санының логарифмі
- log10 (n): ондық логарифм

Бұл модуль бойынша кейбір мысалдарды қарастырайық:

```
>>> import math
>>> print(math.pi) – Пи саны 3.141592653589793
```

>>> print(math.e) – е саны 2.718281828459045
>>> print(math.sqrt(4)) sqrt – шаршы түбір 2.0
>>> print(math.sin(1)) – синус 0.8414709848078965
>>> print(math.cos(1)) – косинус 0.5403023058681398
>>> print(math.tan(0)) – тангенс 0.0
>>> print(math.ceil(3.4)) – ең жақын үлкен санға дөңгелектеу 4
>>> print(math.floor(1.6)) – төменге дөңгелектеу 1
>>> print(math.factorial(5)) – саның факториалын анықтау 120

random модуль кездейсоқ сандардың пайда болуын бақылайды. Оның негізгі функциялары:

- random (): 0.0-ден 1.0-ге дейінгі кездейсоқ санды жасайды
- randint (): белгілі бір ауқымнан кездейсоқ санды қайтарады
- randrange (): сандардың белгілі бір жиынтығынан кездейсоқ санды қайтарады
- shuffle (): тізімін араластырады
- choice (): кездейсоқ тізім элементін қайтарады

Кездейсоқ () функциясы 0,0-дан 1,0-ге дейін кездейсоқ өзгермелі нүкте нөмірін қайтарады. Егер бізге үлкен ауқымнан нөмір қажет болса, 0-ден 100-ге дейін айта беріңіз, онда кездейсоқ функцияның нәтижесін тиісінше 100-ге көбейтуге болады.

```
import random
number = random.random()
print(number)
number = random.random() * 100
print(number)
```

Randint (min, max) функциясы екі және ең жоғары мәндердің арасындағы кездейсоқ бүтіндігін қайтарады.

```
import random
number = random.randint(20, 35)  # значение от 20 до 35
print(number)
```

Randrange () функциясы сандардың белгілі бір жиынтығынан кездейсоқ бүтін санды қайтарады. Оның үш түрі бар:

- randrange (тоқтату): 0-ден қашықтыққа дейінгі диапазон кездейсоқ шамасы шығарылатын сандардың жиынтығы ретінде пайдаланылады
- randrange (бастау, тоқтату): сандардың жиынтығы саннан басталатын санға дейінгі диапазонды білдіреді
- randrange (бастау, тоқтату, қадам): сандардың жиынтығы саннан басталатын санға дейінгі диапазонды білдіреді, әр қадам алдыңғы сатыдан өзгеше

```
import random
number = random.randrange(10)
print(number)
number = random.randrange(2, 10)  print(number)
```

```
number = random.randrange(2, 10, 2)
print(number)
```

Кездейсоқ модуліндегі тізімдермен жұмыс істеу үшін екі функция анықталады: shuffle () функциясы тізімді кездейсоқ таңдайды және choice () функциясы тізімнен бір кездейсоқ элементті қайтарады:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8]
random.shuffle(numbers)
print(numbers)      # 1
random_number = random.choice(numbers)
print(random_number)
```

Python-да белгілі бір мәдениет үшін пішімдеу мәселесін шешу үшін *жергілікті модуль* орнатылған.

Бірінші параметр функция қолданылатын санатты - сандарға, валюталарға немесе екі нөмірге және валюталарға көрсетеді. Параметр үшін мән ретінде келесі тұрақты мәндердің біреуін жібере аламыз:

- LC_ALL: барлық санаттарға локализация қолданылады - сандарды, валюталарды, күндерді және т.б. пішімдеу.
- LC_NUMERIC: санға локализацияны қолданады
- LC_MONETARY: валюталарға локализацияны қолданады
- LC_TIME: күн мен уақытқа локализацияны қолданады
- LC_CTYPE: таңбаларды жоғарғы немесе кіші әріпке аударғанда оқшаулауды қолданады.
- LC_COLLIATE: жолдарды салыстыру кезінде жергілікті тілді қолданады

Сандар мен валюталарды пішімдеу үшін тікелей жергілікті модуль екі функцияны қамтамасыз етеді:

- currency (num): пішім валютасы
- format (str, num): жолдың орнына толтырғыштың нөмірін ауыстырады

Келесі толтырғыштар пайдаланылады:

- d: бүтін сандар үшін
- f: өзгермелі нүкте нөмірлері үшін
- e: экспоненциалды белгілеу үшін

1.6 Шартты оператор

Python тілінде синтаксистің қызықты ерекшелігі бар: кодта операторлық жақшалар жоқ (begin..end, {..олардың орнына операторлар қандай да бір конструкцияның ішінде орындағанын көрсетеді.

Python-да if else шартты нұсқаулығының стандартты жазбасы келесідей:

```
if шарт1:
    оператор1
elif шарт2:
    оператор2
else:
```

```
        оператор3

    if x > 0:
      if x < 2:
    else:
      оператор
```

Python-да салыстыру белгісі екі белгі == ретінде жазылады:

```
if x < 0:
  блок1
elif x == 0: # салыстыру
  блок2
else:
  блок3
```

Шартты пайдаланудың басқа мысалы:

```
if x < 0:
    print('мало')
elif x = 0:
    print('средне')
else:
    print('много')
```

Мысалы,

```
    num = int( input ('Please Enter A Number: ' ) )
    if num > 5 :
    print ('Number Exceeds 5' )
    elif num < 5 :
    print( 'Number is Less Than 5' )
    else :
    print ('Number Is 5' )
```

Күрделі жағдайлар

Қос теңсіздікті пайдалану рұқсат етілген:

```
  if 0 < x < 2:
    if 0 < y < 2:
  else:
    оператор
```

elif шартын пайдалану үлгісі:

```
  if x < 0:
      print('мало')
  elif -0 <= x <= 0:
      print('средне')
  else:
      print('много')
```

Сонымен қатар, логикалық оператор AND (және):

```
if x >= 30 and x <= 40:
 ...
    if num > 7 and num < 9 :
```

```
print( 'Number is 8' )
if num = 1 or num = 3 :
print( 'Number Is 1 or 3' )
```

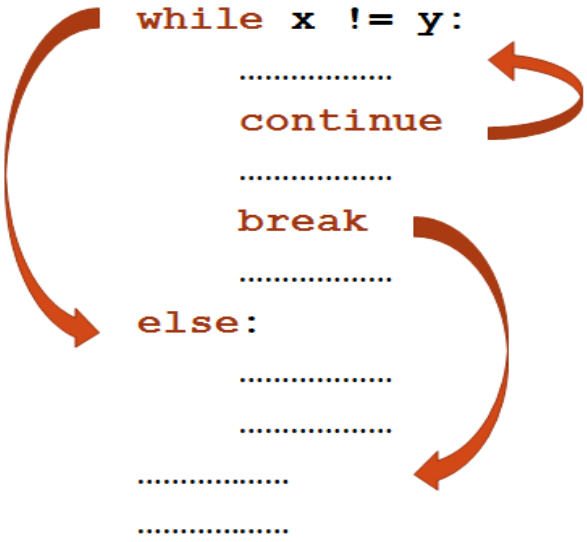
1.7 Циклдер

WHILE циклі

while циклі - шарт алдындағы цикл. Қолдану мысалы:

```
i = 5
while i < 15:
    print(i)
    i = i + 2   #   5 7 9 11 13
```

break және continue операторлары



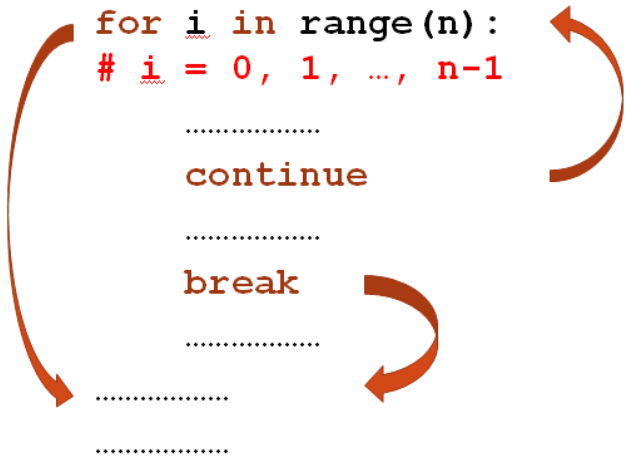
Break операторы-циклден шығу, Continue операторы-ағымдағы цикл итерациясынан шығу

Мысалы

```
a=0
while a!=10:
    a=a+1
    if a==5:
        continue
    print (a)
    if a==7:
        break
print ("всё!")
```

FOR циклі

Python-дағы for циклында келесідей синтаксисі бар:



Мысал:

Екілік дәрежесін 1-ден 10-ға дейін шығару.

```
for x in range(1,11):
  print ( 2**x ) # 2 4 8 16 ... 1024
```

Цикл санауышының қадамын өзгертуге болады:

```
for x in range(1,11,2):
  print ( 2**x )
```

тағы бір мысалы:

```
for i in 'hello world':
     if i == 'o':
         continue
     print(i, end='') # hell  wrld
```

Break қолдану мысалы:

```
for i in 'hello world':
     if i == 'l':
         break
     print(i, end='') # he
```

Мысал-тапсырма, бағдарламаның нәтижесін анықта

```
a=0
n=10
for i in range(1,n+1,2):
    a=a+1
    if a==5:
        continue
    print (a)
    if a==7:
```

```
        break
print ("всё!")
```

1.8 Python ішкі бағдарламалар құрылымы

Python-да ішкі бағдарламалар құрылымдары процедура, функция түрінде ұйымдастырылады. *Python – дағы процедура(функция), бағдарламалар жазудың негізі болып табылады.*

Функциялар(процудуралар) нақты тапсырманы орындайтын және бағдарламаның басқа бөліктерінде қайта пайдалануға болатын код блогын білдіреді[2,3].

Жалпы синтаксисі:

```
def П-Ф_атау ([параметрлер]):
    инструкциялар
```

Функция(процудур) анықтамасы функцияның атауынан, параметрлері бар жақшалар жиынынан және қос нүктеден тұратын def – сөзінен басталады. Жақшадағы параметрлер қосымша емес. Келесі жолдан функция(процудура) орындайтын нұсқаулар блогы пайда болады. Барлық нұсқаулықтар жолдың басынан шығады.

Процедураның синтаксисін қарастырайық:

```
def Err():  # процедура анықтау
  print ("Қате: деректер дұрыс емес" )
n = int (input('введите положительное число'))
if n < 0:
  Err() #  процедура шақыру
```

- Процедура - кейбір әрекеттерді орындайтын көмекші алгоритм
- Бұл шақыруға болатын бағдарламаның шағын ішкі бөлігі
- Процедураны анықтау def қызметтік сөзінен басталады.
- Процедураны шақыру оның аты бойынша жүзеге асырылады, одан кейін дөңгелек жақшалар, мысалы, Err().
- Бір бағдарламада бір процедураның көптеген шақырулары болуы мүмкін.
- Процедураларды пайдалану кодты қысқартады және ыңғайлылықты арттырады.

Параметрлі процедура

Python-да процедура параметрлері қалай қолданылатынын қарастырайық.

Мысал: көрсетілген таңбаны 60 рет басып шығаратын процедура жазу керек (символ пернетақтадан енгізу).

```
1   def printChar(s):
```

```
2      print (s)
3    sim = input('введите символ')
4    printChar(sim) # алғашқы шақыру
5    printChar('*') # келесі шақыру,  * шығару
```

Жергілікті және жаһандық айнымалылар

```
1  x = 3        # жаһандық айнымалы
2  def pr():    # параметрсіз процедура
3    print (x) # жаһандық айнымалы мәнін шығару
4  pr()
```

```
1  x = 3  # жаһандық айнымалы
2  def pr(a): # параметрлі процедура
3    a = 4 # жергілікті айнымалы
4    print (a) # 4
5  pr(x) # жаһандық айнымалы параметріне жіберу (3)
```

Python тілінің функциясын құру мысалын қарастырайық.

Мысалы:

```
def sumD(n):
  sum = 0
  while n!= 0:
     sum += n % 10
     n = n // 10
  return sum
print (sumD(1075))
```

- Функция – негізгі бағдарлама бөлігі.
- Процедура секілді, функция шақыру кезінде анықталуы керек
- Процедурдан айырмашылығы қайтару мәндерінде
- Функцияның мәнін қайтару үшін, `return` пайдаланыңыз.
- Функция мәнін шығару өз атымен `(sumD(1075))` бірге жүреді.

Python күрделі математикалық өрнектерді құрастырудан басқа, функциялардың шақыру нәтижелерін басқа функциялардың аргументтері ретінде қосымша айнымалыларды пайдаланбай жіберуге мүмкіндік береді

Бағдарламаларды жазу кезінде әр түрлі объектілерді түрлендіру қажет. Өйткені біз тек сандық объектілермен ғана таныстық, сондықтан оларды түрлендіру үшін функцияларды қарастырамыз.

int() аргументтер берілмеген жағдайда 18 немесе 0 жолынан жасалған бүтін санды объектті қайтарады. float() саннан немесе жолдан жасалған өзгермелі нүктелі санды қайтарады.

Функциялар жұмысының сипаттамасын қайдан алуға болады? Бағдарламашылар бұл үшін құжаттаманы пайдаланады. Python функциясына

арналған құжаттама help() функциясының аты кіретін функцияның көмегімен туындауы мүмкін:

>>> help(abs)

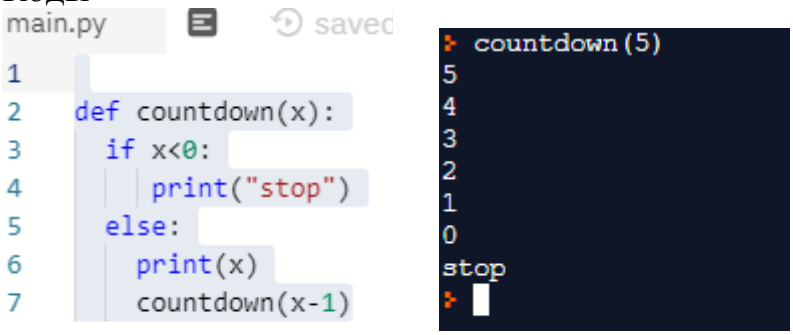
Рекурсия –өзін шақыратын функкция командалар жинағы.Амалдардың ішінде командалар қайталанса немесе функция өз-өзі шақыратын функция рекурсия деп аталады. Мысалы, Бірінші функцияда және екінші функция бар болғаны экранға енгізу арқылы сөздер шығарылады.

Ол рекурсия яғни шексіз орындалатын функция

```
def f1():
  print('f1 orindaldy')
def f2():
  print('f2 orindaldy')
  f1()
def f3():
  print('f3 orindaldy')
      f3()
```

Рекурсияя тоқтату үшін шарт керек. Шарт арқылы рекурсияға мысал қарастыру. Осы мысалда шарт 0 ден кіші болған жағдайда `"stop"` сөзі шығарылады, ал оған дейін рекурсия функция жүзеге аса береді

Коды



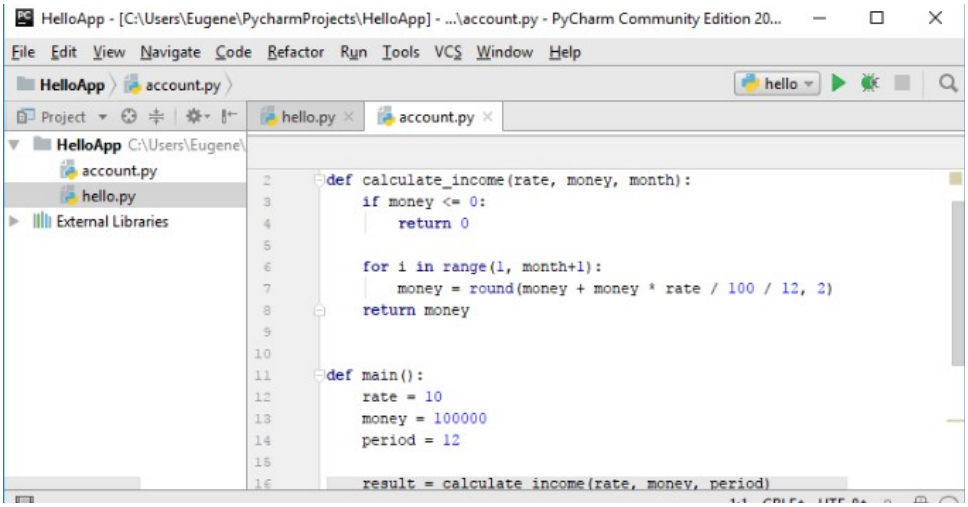
Келеси біз математикалық есептерді рекурсия функция арқылы көрейік. Әсіресе факториал есептері шығару мақсатында ең бастысы return қайталау операторы арқылы және шарт жүргізу арқылы жүзеге асады

Коды

```
def fact(n):
  if(n==1):
     return 1
  else:
    return n*fact(n-1)
```

Python модулі басқа бағдарламаларда қайта пайдалануға болатын кодпен бөлек файлды білдіреді. Модуль жасау үшін модульді білдіретін *.py кеңейтімімен нақты файлды жасау керек. Файл атауы модульдің атауын көрсетеді. Содан кейін осы файлда бір немесе бірнеше функцияларды анықтау

керек. Басты бағдарлама файлын hello.py деп атаңыз. Біз оған сыртқы модульдерді қосқымыз келеді. Бұл әрекетті орындау үшін алдымен жаңа модуль анықтаймыз, account.py деп аталатын жаңа файлды құру керек. Егер PyCharm немесе басқа IDE пайдалансаңыз, онда екі файл да бір жобаға орналастырылады.



Сурет 13 - Модуль жасау

Python бағдарламалау кезінде қателердің екі түрі кездеседі. Бірінші түрі - синтаксистік қате. Олар бастапқы кодты жазу кезінде бағдарламалау тілінің синтаксисін бұзу нәтижесінде пайда болады. Егер мұндай қателер болса, бағдарламаны құрастыруға болмайды. Кез-келген әзірлеу ортасында жұмыс істегенде, мысалы, PyCharm-те, IDE өзі синтаксистік қателерді бақылай алады және оларды қандай да бір жолмен бөле алады.

Қателердің екінші түрі - орындалу қатесі. Олар жасалған бағдарламада оны орындау кезінде пайда болады. Мұндай қателерге ерекше жағдайлар да жатады.

2 Python бағдарламалау негіздері

2.1 Мәтінді жолдар

Python-да мәтінді жолдар үлкен қолданыс табады. Мысалы, әртүрлі жолдарды бір-бірімен біріктіруге болады немесе олардың жеке бөліктерін шығаруға болады. Мәтінді жол әріптерден, цифрлардан, таңбалардан немесе бос орындардан тұра алады.

Мысал ретінде екі айнымалыны алайық:

```
>>> a = "I am "
>>> b = "human"
>>> print(a + b)
I am human
```

Жолдағы жеке символдарды жолдан кесіп алуға, қолдануға және экранға шығаруға болатындай әрбір символдың өзіндік реттік нөмірі бар. Python-да санақ 0-ден басталады. Яғни, символдың ретін (позициясын) санауды 0-ден бастайды. Жолдың бірінші символы - 0, екінші символы - 1, үшінші символы - 2 және әрі қарай осылай кете береді.

Мысал қарастырайық:

```
C O M P U T E R
0 1 2 3 4 5 6 7
```

```
>>> c = "COMPUTER"
>>> print(c[0])  -  C
>>> print(c[1])  - O
>>> print(c[2]) - M
>>> print(c[7]) - R
```

Және де, біз теріс индекстерді қолдана аламыз. Мысалы, [-1] - жолдың ең соңғы символы (c[-1] = "R"), [-2] - соңғының алдындағы символ, яғни рет керісінше кетеді.

Python-да жұп тырнақшаларының ішінде дара тырнақшаларды жазуға болады (апостроф ретінде) немесе керісінше жұп тырнақшаларды дара тырнақшалардың ішіне жазуға болады

```
a = "It's cool"
b = 'it is "funny" '
```

Бірақ, бір тырнақшалар түрінің ішіне бірдей тырнақшаларды қойсаңыз, программа қате болады. Қате болмау үшін, бэкслеш (\ ) символын қолдану қажет:

```
>>> print('It\'s a cloudy day')  //   It's a cloudy day
```

Жол стандартты input() ***функциясымен есептеледі:***

```
  a=input()
  print(a)
```

Жолдар операциялары

Екі жолды біріктіру (конкатенация), жолды көбейту операциясы:

```
a="па"
b="рад"
print(a+b) # парад
a="кар"
print (a*4) # каркаркаркар
```

Массивтермен жұмыс істеу (индекс 0-тен басталады):

```
a="парад"
print (a[2]) # p
```

Жол ұзындығы-len() функциясы:

```
a="парад"
print (len(a)) # 5
```

Кескіндер/жолдағы символдар фрагменттерін кесу/
Кескін шығару операторы:
[X:Y] // X-кесудің басталу индексі, ал Y-оның аяқталуы

```
tday = 'morning, afternoon, night'
tday[0:7] # 'morning'
```

Python-да кесу қалай қолданылады:

```
s = 'spameggs'
s[3:5]    # 'me'
s[2:-2]   # 'ameg'
s[-4:-2]  # 'eg'
s[:6]     # 'spameg'
s[1:]     # 'pameggs'
s[:]      # 'spameggs'
```

Қадамды орнатуға болады:

```
s = 'spameggs'
s[::-1]       # 'sggemaps'
s[3:5:-1]     # ''
s[2::2]       # 'aeg'
```

Мысал: Үш еселік индекстері бар таңбаларды жолдан алып тастаңыз. Тапсырманы циклді пайдалана отырып орындауға болады:

```
s = 'spameggs'
x=3
l=len(s)//3
for i in range(l):
  print(s[x:x+1:3])    # m g
  x+=3
```

Немесе жай кескінді пайдалана аласыз:

```
s = 'spameggs'
print(s[1::3])
```

Жол әдістері

join - жолдарды бөлінген бөлгіш арқылы қосу

```
s="hello"
s1="-".join(s)
s1 # 'h-e-l-l-o'
lst=['11','22','33']
lst="-".join(lst)# '11-22-33'
```

s1 жолында ішкі жолақтардың кездесулерінің саны. Нәтижесі Сан болып табылады. I іздеуді бастау және j іздеуді аяқтау орнын көрсетуге болады:

```
s1="abrakadabra"; s1.count('ab') # 2
s1.count('ab',1) # 1
s1.count('ab',1,-3) # 0 , т.к. s1[1:-3]='brakada'
```

s1.find (s[, i, j]) — s ішкі жолының s1 жолына бірінші (сол жақ санағанда) кіру позициясы анықталады. Нәтижесі Сан болып табылады. I және j іздеу аймағының басталуы мен аяқталуын анықтайды:

```
s1="abrakadabra"; s1.find('br') # 1
```

s1.replace (s2, s3 [, n]) — жаңа жол құрылады, онда бастапқы жолдың S2 фрагменті (қосымша жол) s3 фрагментіне ауыстырылады. Қосымша дәлел n ауысым санын көрсетеді:

```
s1="breKeKeKeKs"; ss=s1.replace('Ke','XoXo',2)
ss # breXoXoXoXoKeKs
```

Жолдарды пішімдеу

Python жолдарды пішімдеу форматтау мағынасын білдіреді.

Format әдісі:

```
'Hello, {}!'.format('Vasya') # 'Hello, Vasya!'  Бір орын
```

Әдістің дәлелі-бағдарламаны орындау кезінде фигуралық жақшалардың орнына қойылатын мәтін-қою.

```
'{0}{1}{0}'.format('abra', 'cad') # 'abracadabra'   бірнеше орын
```

Ауыстырулар нөмірленеді, форматы әдісінің дәлелдері жақшалардағы реттік сандарға сәйкес ауыстыру позицияларын толтырады.

Бірнеше алмастырулармен басқа пішімдеу параметрі:

```
'Coordinates:{latitude},{longitude}'.format(latitude='37.24N',longitude='-115.81W')
'Coordinates: 37.24N, -115.81W'
```

2.2 Python тізімдері

Python тізімдер, топтамалар және деректер жиынымен жұмыс істеу үшін сөздіктер сияқты кіріктірілген түрлерін ұсынады.

Python-дағы массивтер(тізімдер), басқа бағдарламалау тілдерінде болғандай, ортақ атаумен бірдей типтегі элементтердің белгілі бір санын

құрайды және әр элементтің өз индексі бар. Алайда, Python тілінде "массив"сияқты құрылым жоқ. Массивтермен жұмыс істеу үшін тізімдер қолданылады. Тізімдер төртбұрышты [] жақшаларға салынған әртүрлі объекттен(мәндер, деректер) тұратын және бір-бірінен үтірмен бөлінген реттелген жүйе болып табылады.
Мысалы, сандардың тізімін анықтаймыз:

```
numbers =[1, 2, 3, 4, 5]
```

Тізімді құру үшін list () конструкторын пайдалануға болады:

```
numbers1 = []   бос тізім
numbers2 = list()
```

Екі тізбелік анықтамалар бірдей - олар бос тізім жасайды. Тізімді құруға арналған тізім конструкторы басқа тізімді қабылдай алады:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers2 = list(numbers)
```

Тізімдерді "+ " белгісі арқылы қоюға (конкатенациялауға) болады»:

```
l = [1, 3] + [4, 23] + [5]
  # [1, 3, 4, 23, 5]
```

Тізім элементтеріне сілтеме жасау үшін тізімдегі элементтің нөмірін көрсететін индекстерді пайдалануыңыз қажет. Индекстер нөлден басталады. Яғни, екінші элементте 1 индексі болады. Соңынан элементтерге сілтеме жасау үшін -1-ден бастап теріс индекстерді қолдана аласыз. Яғни, соңғы элементте -1 индексі болады, соңғы, бірақ біреуі -2, және т.б. болады.

Тізімдермен жұмыс істеудің әдістері мен функциялары[1,5]:

- append (item): элементті тізімнің соңына қосады
- insert (элемент индекс): элементті тізімге тізімге қосады
- remove (item): item элементін жояды. Элементтің бірінші пайда болуы ғана жойылады. Егер элемент табылмаса, ValueError ерекшелігін шығарады.
- clear (): тізімдегі барлық элементтерді алып тастаңыз
- index (item): элемент элементінің индексін қайтарады. Егер элемент табылмаса, ValueError ерекшелігін шығарады.
- pop ([index]): элементті индекс индексінде жояды және қайтарады. Егер индекс өтпесе, ол соңғы элементті ғана жояды.
- count (item): тізімдегі элементтің қайталану санын қайтарады
- sort([key]): элементтерді сұрыптайды. Әдепкі бойынша, өсу тәртібімен сұрыпталады. Бірақ негізгі параметрмен сұрыптау функциясын бере аламыз.
- reverse (): тізімдегі барлық элементтерді кері тәртіпте орналастырады
- len (list): тізім ұзындығын қайтарады
- sorted(list, [key]): сұрыпталған тізім қайтарады

- min (list): ең кіші тізім элементін қайтарады
- max (list): ең үлкен тізім элементін қайтарады

Жолдар тізімге өте ұқсас элементтер тізбегін білдіреді, тек қана бұл өзгермелі үлгі болып табылады. Сондықтан, біз топтамаға элементтерді қосып немесе алып тастай алмаймыз, оны өзгерте алмаймыз.

Жақша жақшалар жасау үшін пайдаланылады, онда оның мәндері үтірлермен бөлінеді:

```
user = ("Tom", 23)
print(user)
```

Сондай-ақ, жолдарды анықтау үшін жақшаларды пайдаланбастан, үтірлермен бөлінген мәндерді жай ғана тізімдей аламыз:

```
user = "Tom", 23
print(user)
```

Тізім –кез келген типтегі элементтердің жинағы.Белгілі бір жинаққа
Сандық тізім: [10,20,30,40]
Жолдық тізім: [‘tip’,’hip’,’uio’]
Аралас: [‘spam’,2,452,12,’sabak’]

```
List () функциясын пайдаланып тізімді алыңыз
l = list ('spisok')  # 'spisok' - жол
print(l) #['s', 'p', 'i', 's', 'o', 'k']
```

```
Split () функциясын пайдаланып тізімді жасау
 stroka ="Hello, world"  # жол
 lst=stroka.split(",") # тізім
```

Python-да тізімді генераторлардың көмегімен жасауға болады:

```
 l = [1]*10
 # [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Екінші күрделі әдіс:

```
 l = [i for i in range(10)]
 # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Немесе мынадай мысал***:***

```
 c = [c * 3 for c in 'list']
 print (c) # ['lll', 'iii', 'sss', 'ttt']
```

Мысал: Тізім генераторын пайдаланып, тізімді 0-ден 9-ға дейінгі сандардың квадраттарымен толтырамыз:

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

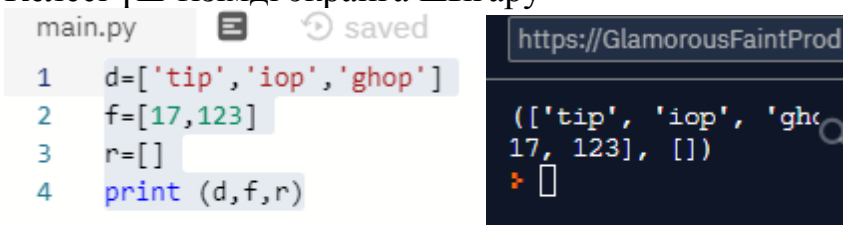
Тағыда мысал:

```
l = [(i+1)+i for i in range(10)]
print(l) # [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

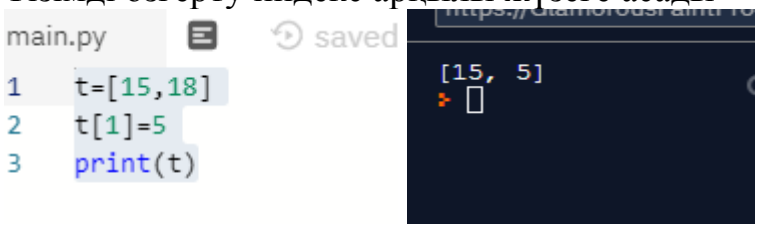
Тізімдегі кездейсоқ сандар:

```
from random import randint
l = [randint(10,80) for x in range(10)]
```

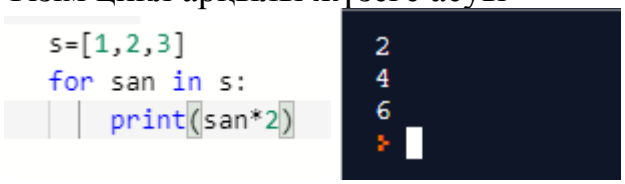
Келесі үш тізімді экранға шығару



Тізімді өзгерту индекс арқылы жүзеге асады



Тізім цикл арқылы жүзеге асуы



Екі еселендіру тізім элементтері

```
t=[1,2,3]
range(0,3)
for i in range(len(t)):
    t[i]=t[i]*2
```

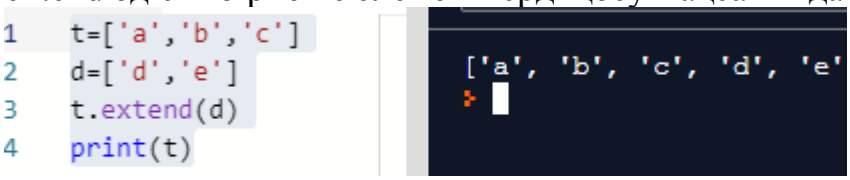
```
+ Біріктіру:
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print (c)
[1, 2, 3, 4, 5, 6]
```

Тізім бөліктері яғни сізге қажет элементтерді индекс арқылы t[:4] жүзеге асты

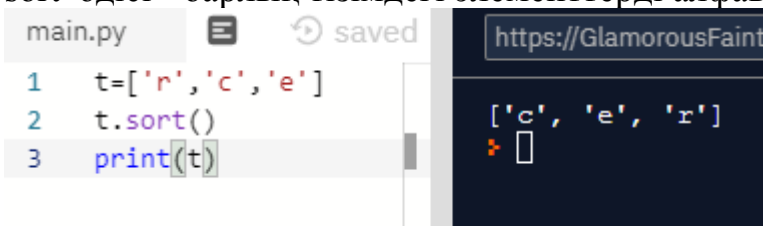
```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> t[1:3]
['b', 'c']
>>> t[:4]
['a', 'b', 'c', 'd']
>>> t[3:]
['d', 'e', 'f']
```

Append әдісі – бір элементті тізімге қосу

extend әдісі - бірнеше элементтерді қосу мақсатында



sort әдісі - барлық тізімдегі элементтерді алфавит бойынша ауыстырады



remove and del әдістері - арқылы тізімнің кез келген элементтерін өшіре аласыз

Python тізімді енгізу

Тізім элементтерін енгізу үшін range циклі қолданылады:

```
for i in range(N):
  print ( "L[", i, "]=", sep = "", end = "" )
  L[i] = int( input() )
```

Тізімді енгізу және шығару:

```
L = [ int(input()) for i in range(N) ]
print (L)

# немесе
for i in range(N):
  print ( L[i], end = " " )
```

2.3 Кортеждер мен Сөздіктер

Кортеждер

Әдетте, тізімдерді деректерді реттеу үшін қолданады, ал оларды сақтау үшін кортеждерді немесе сөздіктерді қолданады. Бұл сабақта біз кортеждерді қарастырамыз. Бағдарламалауда көп элементтері бар деректер типі контейнер деп аталады.

Жалпы, кортеж дегеніміз - аздап тізімдерге ұқсас, дөңгелек жақшаларға алынатын және өзгермейтін элементтерді сақтайтын деректер құрылымы. Яғни, кортежде орналасқан элементтерді өзгертуге болмайды, өзгертіп көрсеңіз, қате шығады. Мысал қарастырайық:

```
>>> person = ("Bob", 20, 183)
>>> print(person)
('Bob', 20, 183)
```

Бұл мысалда біз адам жайындағы ақпаратты сақтайтын кортежді құрдық және элементтерді экранға шығардық. Кортеждің элементтерін де индекс арқылы экранға шығаруға болады (мысалы, print(person[1]) -> 20).

Егер біз кортеж элементтерін өзгертіп көрсек программа қатені шығарады:

```
>>> person[2] = 175
>>> print(person)
Traceback (most recent call last):
File "pyshell#9", line 1, in module
person[2] = 175
TypeError: 'tuple' object does not support item assignment
```

Сөздіктер

Сөздіктер кортеждерге ұқсас, бірақ оларда кілт индекстер және мәндер болады. Яғни сөздіктегі әрбір элементтің мәні және кілті бар. Сөздікте элементтердің ретпен орналасуы қажет емес және олардың мазмұны өзгертілуі мүмкін. Мысал қарастырайық:

```
>>> mark = {"Alikhan" : "A", "Sanzhar" : "A", "Arman" : "C"}
>>> print(mark)       //  {'Alikhan': 'A', 'Sanzhar': 'A', 'Arman': 'C'}
```

Сөздіктің мәндерін шығару үшін, олардың кілттерінің аттарын жазу керек:

```
>>> print(mark["Alikhan"]        // 'A'
```

Және де, біз кілттердің мәндерін өзгерте аламыз немесе оларды жоя аламыз:

```
>>> mark['Arman'] = "A"
>>> print(mark)            // {'Alikhan': 'A', 'Sanzhar': 'A', 'Arman': 'A'}
```

```
>>> del mark["Sanzhar"]
>>> print(mark)
{'Alikhan': 'A', 'Arman': 'A'}
```
- Бұл мысалда, біз "del" командасы арқылы 'Sanzhar' кілтін жойдық.

2.4 Файлдармен жұмыс

Python әр түрлі файл түрлерін қолдайды, бірақ шартты түрде оларды екі түрге бөлуге болады: мәтін және екілік. Мәтіндік файлдарға мысалы, cvs, txt, html, кеңейтімі бар файлдар жатады. Жалпы мәтіндік пішінде ақпаратты сақтайтын кез келген файлдар. Бинарлық файлдар - суреттер, аудио және бейне файлдар және т.б. Файл түріне байланысты онымен жұмыс істеу сәл өзгеше болуы мүмкін.

Файлдармен жұмыс істеу кезінде сіз белгілі бір операциялардың кезектілігін қадағалауыңыз керек:

- Open() әдісімен файлды ашу
- Read() әдісін пайдаланып файлды оқу немесе write () әдісін қолданып файлға жазу
- Close() әдісі арқылы файлды жабыңыз

Файлмен жұмыс істеуді бастау үшін келесі ресми анықтамасы бар open () функциясы арқылы ашылуы керек:

open(file, mode)

Функцияның бірінші параметрі файлға жол ұсынады, мысалы

C: //somedir/somefile.txt.

Немесе сіз салыстырмалы болуы мүмкін, мысалы, somedir / somefile.txt - бұл жағдайда, файл жұмыс істейтін Python сценарийінің орнын іздейді.

Екінші аргумент - бұл режим файлды ашуға арналған режимді, немен айналысатынымызға байланысты. 4 жалпы режим бар:

- r (оқу). Файл оқу үшін ашылады. Егер файл табылмаса, онда FileNotFoundError лақтырылады.
- w (Жазу). Файл жазу үшін ашылды. Егер файл жоқ болса, ол жасалады. Егер мұндай файл бұрыннан бар болса, ол қайтадан жасалады, және, тиісінше, ескі деректер жойылады.
- a (Қосу). Файл қайта жазу үшін ашылады. Егер файл жоқ болса, ол жасалады. Егер мұндай файл бұрыннан бар болса, деректер аяқталғанға дейін жазылады.
- b (екілік). Екілік файлдармен жұмыс істеу үшін пайдаланылады. Басқа режимдермен бірге пайдаланылады - w немесе r.

Файлмен жұмыс істеуді аяқтағаннан кейін close() әдісімен жабуыңыз керек. Бұл әдіс файлмен байланысты барлық ресурстарды босатады. Файлды ашқан кезде немесе онымен жұмыс істеу кезінде әртүрлі жағдайларды кездестіруіміз мүмкін, мысалы, оған рұқсат жоқ және т.б. Бұл жағдайда бағдарлама қатеге түседі және сәйкесінше файл жабық болмайды.

Кодта бұл келесідей:

```
Fin = open ( "input.txt" )
Fout = open ( "output.txt", "w" )
Fout.close()
Fin.close()
```

Мәтіндік файлға жазу. Жазу үшін мәтіндік файлды ашу үшін, w (қайта жазу) немесе (қайта жазу) режимін қолдану керек. Содан кейін жазуға арналған жазу (str) әдісі пайдаланылады, оған жазылған жол беріледі. Айта кету керек, бұл жазылған жол, сондықтан сізге сандарды, басқа түрдегі деректерді жазу қажет болса, оларды алдымен жолға түрлендіру керек.

Файлды оқу үшін ол r (Оқу) режимі арқылы ашылады, содан кейін оның мазмұнын түрлі әдістермен оқуға болады:

- readline (): файлдан бір жолды оқиды
- read (): файлдың бүкіл мазмұнын бір жолда оқиды

- readlines (): файлдың барлық жолдарын тізімге оқиды

Әр жолды оқуға readline() әдісін пайдаланбайтынымызға қарамастан, бірақ файлды іздегенде, бұл әдіс әр жаңа жолды алу үшін автоматты түрде шақырылады.

```
str1 = Fin.readline() # str1 = 1
str2 = Fin.readline() # str2 = 2
```

Мысал: Файлда екі сан жазылған.
input файлы.txt:12 17
жауабы: 27

1. тәсіл: Int функциясы жол мәнін санға түрлендіреді

```
Fin = open ( "D:/input.txt" )
str = Fin.readline().split()
x, y = int(str[0]), int(str[1])
print(x+y)
```

2. тәсіл:

```
...
x, y = [int(i) for i in s]
print(x+y)
```

Python write әдісі файлға жолды жазу үшін қолданылады:

```
Fout = open ( "D:/out.txt","w" )
Fout.write ("hello")
```

Файлға жазуды белгілі бір пайдалану арқылы жүзеге асыруға болады. Бұл жағдайда {:d} үлгілерінің орнына format (алдымен x, содан кейін y, одан кейін x+y) әдісінің параметрлерінің мәндері дәйекті түрде қойылады.

```
Fout.write ( "{:d} + {:d} = {:d}\n".format(x, y, x+y) )
```

EOF (файл соңы болса) while циклін тәсілі тізімге жолдарды қосу:

```
while True:
  str = Fin.readline()
  if not str: break

Fin = open ( "input.txt" )
lst = Fin.readlines()
for str in lst:
   print ( str, end = "" )
Fin.close()
```

Python үшін қолайлы әдіс:

```
for str in open ( "input.txt" ):
  print ( str, end = "" )
```

Ақпаратты ыңғайлы пішімде сақтайтын жалпы файл пішімдерінің бірі csv пішімі. Csv файлындағы әрбір жолда үтірмен бөлінген бөлек бағандардан тұратын бөлек жазба немесе жол көрсетіледі. Осылайша, формат «үтірмен бөлінген мәндер» деп аталады. Csv пішімі мәтіндік файл пішімі болса да, Python онымен жұмыс істеуді жеңілдету үшін арнайы орнатылған csv модулін ұсынады.

Файлға екі өлшемді тізім жазылады - шын мәнінде, әр жол бір пайдаланушы болып табылатын кесте. Әр пайдаланушыда екі өріс бар: аты мен жас. Яғни, шын мәнінде, үш жолдың және екі бағанның кестесі.

Файл жазу үшін ашылған кезде, үшінші параметр newline = «» мәнін анықтайды - бос жол операциялық жүйеге қарамастан, файлдан сызықтарды дұрыс оқуға мүмкіндік береді.

Жазу үшін csv.writer(file) функциясымен қайтарылатын жазушы объектісін алу керек. Бұл функцияға ашық файл беріледі. Және нақты жазба writer.writerows әдісі арқылы жасалады (пайдаланушылар) Бұл әдіс жолдардың жиынтығын алады. Біздің жағдайда бұл екі өлшемді тізім.

Егер сізге бір өлшемді тізімді қосу қажет болса, мысалы, [«Sam», 31], онда бұл жағдайда writer.writerow(user) әдісін шақыруға болады

Жоғарыда келтірілген мысалда әр жазба немесе жол жеке тізім, мысалы, [«Sam», 31]. Сонымен қатар, csv модулінде сөздіктермен жұмыс істеу үшін арнайы қосымша мүмкіндіктер бар. Атап айтқанда, csv.DictWriter() функциясы файлға жазуға мүмкіндік беретін жазушы нысанын қайтарады. Csv.DictReader() функциясы файлдан оқу үшін оқу құралының нысанын қайтарады. Мысалы:

```
import csv
FILENAME = "users.csv"
users = [
    {"name": "Tom", "age": 28},
    {"name": "Alice", "age": 23},
    {"name": "Bob", "age": 34} ]
with open(FILENAME, "w", newline="") as file:
    columns = ["name", "age"]
    writer = csv.DictWriter(file, fieldnames=columns)
    writer.writeheader()
    writer.writerows(users)
    user = {"name" : "Sam", "age": 41}
     writer.writerow(user)
with open(FILENAME, "r", newline="") as file:
    reader = csv.DictReader(file)
    for row in reader:
        print(row["name"], "-", row["age"])
```

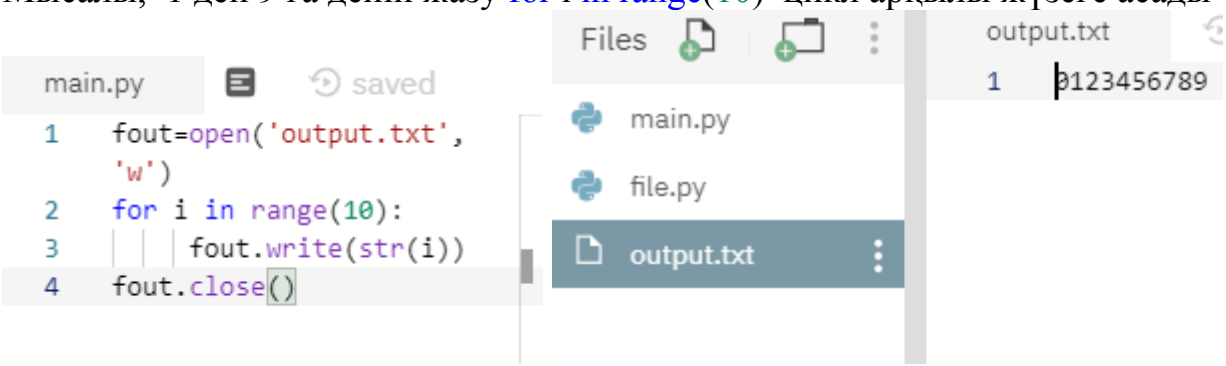
Жолдар writeerow() және writerows() әдістерімен жазылады. Бірақ енді әр жол жеке сөздік болып табылады, сонымен қатар, баған тақырыптары writeheader() әдісі арқылы жазылады және бағандар жиынтығы csv.DictWriter әдісіне екінші параметр ретінде беріледі.

Екілік файлдар, мәтіндік файлдардан өзгеше, ақпаратты байт жиынтығы түрінде сақтайды. Python-мен олармен жұмыс істеу үшін сізде кіріктірме қондырылған модуль қажет. Бұл модуль екі әдісті ұсынады:

- dump (obj, file): obj екілік файлға жазады
- load(file): екілік файлдан деректерді нысанға оқиды

```
import pickle
FILENAME = "user.dat"
name = "Tom"
age = 19
with open(FILENAME, "wb") as file:
    pickle.dump(name, file)
    pickle.dump(age, file)
with open(FILENAME, "rb") as file:
    name = pickle.load(file)
    age = pickle.load(file)
    print("Имя:", name, "\tВозраст:", age)
```

Мысалы, 1 ден 9 ға дейін жазу for i in range(10) цикл арқылы жүзеге асады

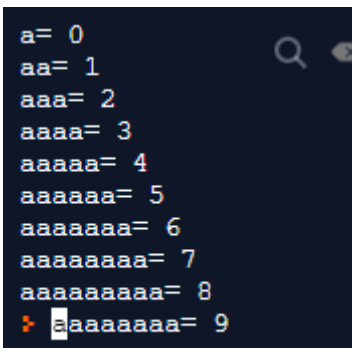


Келесі біздің қарастыратынымыз сандардың аралары бос орын болғанын қаласаңыз fout.write(str(i)+' ') осыны қолдануға болады

```
fout=open('output.txt','w')
for i in range(10):
    fout.write(str(i)+' ')
fout.close()
```

Форматтау арқылы қолдану

```
fout=open('output.txt','w')
s='a'
for i in range(10):
    print('%s= %d' % (s,i))
    s=s +'a'
fout.write(s)
fout.close()
```



2.5 Python графикасы

Графикалық кітапханаларға шолу

Python тіліндегі бағдарламалар үшін пайдаланушының графикалық интерфейсін (GUI, Graphical User Interface) құру графикалық интерфейс компоненттерінің тиісті кітапханаларының көмегімен немесе кальканы ағылшын тілінен, виджет кітапханаларын пайдалана отырып болады.

Келесі тізім толық емес, бірақ бар шешімдердің алуан түрлілігін көрсетеді:

- Tkinter көп платформалы пакет компоненттердің орналасуын жақсы басқаруға ие. Интерфейс әртүрлі платформаларда (Unix, Windows, Macintosh) бірдей көрінеді. Python стандартты жеткізіліміне кіреді. Құжаттама ретінде Фредрик Лунд жазған "An Introduction to Tkinter" ("Tkinter-ге кіріспе") нұсқаулығын пайдалануға болады[13]:
- wxPython wxwidgets көп ақылы кітапханасында салынған (бұрын wxWindows деп аталды). Барлық платформалар үшін Ана болып көрінеді, белсенді жетілдірілуде, GL қолдауы жүзеге асырылды. Барлық негізгі платформалар үшін бар. Мүмкін, болашақ Python нұсқаларында Tkinter орын алады. Сайт: http://www.wxpython.org/
- PyGTK gtk+ және Gnome үшін визуалды компоненттер жиынтығы. Тек GTK платформасы үшін.
- PyQT / PyKDE Qt (UNIX немесе Windows астында) немесе KDE қолданатын адамдар үшін жақсы пакеттер.
- Pythonwin MFC айналасында салынған, win32all пакетінде қабықпен бірге жеткізіледі; тек Windows үшін.
- pyFLTK Xforms аналогы, OpenGL қолдауы. Windows және Unix платформалары үшін бар. Сайт: http://pyfltk.sourceforge.net/
- AWT, JFC, Swing Jython бірге жеткізіледі, ал Jython үшін Java пайдаланатын құралдар қол жетімді. Java платформасын қолдайды.
- anygui Python бағдарламасына арналған графикалық интерфейсті құру үшін төменгі платформадан тәуелсіз пакет. Сайт: http://anygui.sourceforge.net/
- Pythoncard HyperCard/MetaCard идеологиясы бойынша ұқсас графикалық интерфейс құрастырушысы. WxPython базасында жасалған[14].

Python-дан Қол жетімді түрлі графикалық кітапханаларға арналған өзекті сілтемелер тізімін мына мекен-жай бойынша табуға болады[15].

Кітапхана банк-клиент. Мысалы, PythonCard Wxpython пайдаланады, мысалы, Linux платформасында wxwindows көп платформалды GUI-кітапханасына негізделген, ол өз кезегінде GTK+ немесе Motif - ға негізделген,

ал сол да X Window шығару үшін қолданылады. Айтпақшы, Motif үшін Python-да өз байланыстары бар.

Дәрісте TCL сценарий тілі үшін белгілі графикалық пакет - TCL/Tk үшін орама болып табылатын Tkinter пакеті қарастырылады. Бұл пакеттің мысалында пайдаланушының графикалық интерфейсін құрудың негізгі принциптерін үйрену оңай.

Графикалық интерфейс туралы

Жалпы мақсаттағы барлық заманауи графикалық интерфейстер WIMP - Window, Icon, Menu, Pointer (терезе, иконка, мәзір, көрсеткіш) моделі бойынша құрылады. Терезелердің ішінде Графикалық интерфейстің элементтері бейнеленеді, олар қысқа виджеттер деп аталады (widget - штучка). Мәзір терезенің әр түрлі бөліктерінде орналасуы мүмкін, бірақ олардың мінез-құлықтары бірдей: олар алдын ала анықталған әрекеттер жиынтығынан әрекетті таңдау үшін қызмет етеді.

Графикалық интерфейсті пайдаланушы" түсіндіреді " қажетті әрекеттерді компьютерлік бағдарламаға көрсеткіш көмегімен түсіндіреді. Әдетте көрсеткіш тінтуір меңзері немесе джойстик болып табылады, бірақ басқа да "сілтегіш" құрылғылар бар. Көмегімен иконок графикалық интерфейс иеленеді тәуелсіздігі тілі және кейбір жағдайларда мүмкіндік береді тез бағдарлай интерфейс.

Графикалық интерфейстің негізгі міндеті пайдаланушы мен компьютер арасындағы коммуникацияны жеңілдету болып табылады. Бұл туралы интерфейсті жобалау кезінде үнемі есте сақтау керек. Программисте (немесе дизайнерде) бар құралдарды қолдану графикалық интерфейсті жасау кезінде әрбір нақты жағдайда пайдаланушыға ыңғайлы виджеттерді таңдай отырып, минимумға дейін қою керек. Сонымен қатар, ең аз таң қаларлық принципін ұстану пайдалы: интерфейс формасынан оның мінез-құлқы түсінікті болуы керек. Нашар ойластырылған интерфейс, тіпті интерфейстің қасбетінде тиімді алгоритм жасырылса да, пайдаланушының бағдарламадан сезінуін бұзады. Интерфейс пайдаланушының типтік әрекеттері үшін ыңғайлы болуы керек.

Көптеген қолданбалар үшін мұндай әрекеттер "шеберлер" (wizards) деп аталатын экрандардың жеке серияларына бөлінген. Алайда, егер қосымша-пайдаланушы өзіне қажетті шешімдерді құра алатын конструктор болса, типтік әрекет шешімді құру болып табылады. Типтік әрекеттерді анықтау оңай емес, сондықтан компромиссом "шеберлер" бар гибрид болуы мүмкін және өз құрылыстары үшін жақсы мүмкіндіктер. Дегенмен, графикалық интерфейс барлық жағдайларда ең тиімді интерфейс емес. Көптеген пәндік салалар үшін шешімді белгілі бір формальды тілде немесе сценарий тілінде алгоритм арқылы мәлімдеу оңайырақ.

Tk Негіздері

Графикалық интерфейсі бар кез келген бағдарламаның негізгі ерекшелігі-интерактивтілік. Бағдарлама жай ғана бір нәрсе (пакеттік режимде) өзінің іске

қосылуынан аяғына дейін санайды: оның әрекеттері пайдаланушының араласуына байланысты. Іс жүзінде, графикалық қосымша оқиғаларды өңдеудің шексіз циклын орындайды. Графикалық интерфейсті іске асыратын бағдарлама оқиғалы-бағытталған. Ол өзінің ішкі жағдайына сәйкес өңделетін оқиғалар интерфейсінен күтеді.

Бұл оқиғалар графикалық интерфейс элементтерінде (виджеттерде) пайда болады және осы виджеттерге бекітілген өңдеушілермен өңделеді. Виджеттердің өздері көптеген қасиеттерге ие(түсі, өлшемі, орналасуы), тиісті иерархияға (бір виджет басқасының иесі болуы мүмкін), жай-күйіне қол жеткізу әдістері бар.

Виджеттердің орналасуы (басқа виджеттердің ішінде) орналасу менеджерлері деп аталады. Виджет орналасу менеджерінің ережелері бойынша орынға орнатылады. Бұл ережелер виджет координаттарын ғана емес, оның өлшемдерін де анықтай алады. Tk-да орналасқан менеджерлердің үш түрі бар: қарапайым ораушы (pack), тор (grid) және еркін орналасуы (place).

Бірақ бұл графикалық бағдарлама үшін жеткіліксіз. Себебі, графикалық бағдарламадағы кейбір виджеттер белгілі бір жолмен өзара байланысты болуы керек. Мысалы, жылжыту жолағы мәтіндік виджетпен өзара байланысты болуы мүмкін: жолақты пайдаланған кезде виджеттегі мәтін қозғалуы тиіс және керісінше, мәтін бойынша жылжыған кезде жолақтың ағымдағы жағдайын көрсетуі тиіс. TK виджеттері арасындағы байланыс үшін виджеттер мен параметрлерді бір-біріне беретін айнымалылар қолданылады.

Виджеттер сыныптары

Tk кітапханасында графикалық интерфейсті құру үшін виджеттердің келесі сыныптары таңдалып алынды (алфавиттік ретпен):

- Button (түйме) кейбір әрекеттерді шақыру (белгілі бір пәрменді орындау) үшін қарапайым түйме.
- Canvas (сурет) графикалық примитивтерді шығару негізі.
- Checkbutton (құсбелгі) оған басқан кезде екі күй арасында ауыса алатын түйме.
- Entry (енгізу өрісі) мәтін жолын енгізуге болатын көлденең өріс.
- Frame (Рамка) басқа көрнекі компоненттерді қамтитын Виджет.
- Label (жазу) Виджет мәтінді немесе графикалық суретті көрсете алады.
- Listbox (тізім) пайдаланушы бір немесе бірнеше элементтерді таңдай алатын тізімді тікбұрышты рамка.
- Menu (мәзір) қалқымалы (popup) және төмен түсетін (pulldown) мәзірлерді жасауға болатын Элемент.
- Menubutton (мәзір түймесі) төмендеу мәзірі бар түйме.
- Message (хабар) жазуларға ұқсас, бірақ ұзын жолдарды бұрауға және орналасу менеджерінің талабы бойынша өлшемін өзгертуге мүмкіндік береді.

- Radiobutton (селекторлы түйме) балама мәндердің бірін ұсынуға арналған түйме. Мұндай түймелер әдетте топта әрекет етеді. Басқан кезде бір түйме тобы таңдалған, бұдан бұрын "секіріп".
- Scale (Шкала) белгілі бір диапазонда жылжу арқылы сандық мәнді орындау үшін қызмет етеді.
- Scrollbar (айналдыру жолағы) айналдыру жолағы басқа виджеттерде айналдыру көлемін көрсету үшін қолданылады. Мүмкін тік және көлденең.
- Text (пішімделген мәтін) бұл тікбұрышты виджет әр түрлі стильдерді пайдаланып мәтінді өңдеуге және пішімдеуге, мәтінге суреттер мен тіпті терезелерді енгізуге мүмкіндік береді.
- Toplevel (жоғарғы деңгей терезесі) бөлек терезе ретінде көрсетіледі және басқа виджеттер ішінде болады.

Бұл сыныптардың барлығы бір - бірімен мұрагерлік қатынастары жоқ-олар тең. Бұл жинақ көптеген жағдайларда интерфейсті құру үшін жеткілікті.

Қазіргі заманғы графикалық интерфейс жүйесінде пернетақта мен тінтуірмен байланысты және қандай да бір виджеттің" аумағында" болып жатқан түрлі оқиғаларды бақылау мүмкіндігі бар. Оқиға елінде мәтіндік жол - үш элементтен (модификаторлар, оқиғаның түрі және оқиғаның детализациясы) тұратын оқиға үлгісі түрінде сипатталады.

Виджетті жасау және конфигурациялау

Виджетті жасау тиісті сынып конструкторының шақыруы арқылы жүзеге асырылады . Конструктор шақыру келесі синтаксис бар:

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

Мұнда widget - виджет класы, masterвиджет-иесі, option және value - конфигурациялық опция және оның мәні (мұндай жұптар бірнеше болуы мүмкін).

Әрбір виджет config () (немесе configure ())) әдістерімен орнатуға және сөздіктермен жұмыс істеу әдістеріне ұқсас әдістермен оқуға болатын қасиеттерге ие. Төменде қасиеттермен жұмыс істеу үшін мүмкін синтаксис:

```
widget.config(option=value, ...)
        widget["option"] = value
        value = widget["option"]
        widget.keys()
```

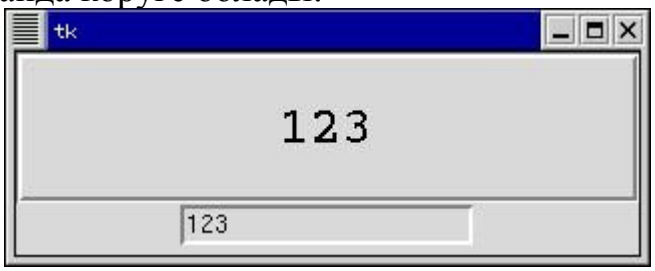
Сипат атауы Python тілінің негізгі сөзімен сәйкес келген жағдайда, атаудан кейін бір астын сызу қолданылады. Сонымен, class қасиетін class_ сияқты, to_ сияқты қою керек.

Виджет конфигурациясын кез келген уақытта өзгертуге болады. Бұл өзгеріс оқиғаны өңдеу циклына қайта оралғанда немесе update_idletasks () айқын қоңырауы кезінде экранда сызылады.

Келесі мысал ішінде екі виджеті бар терезені көрсетеді. Айнымалы жазба енгізу өрісімен тікелей байланысты. Бұл мысал конфигурациялау мүмкіндіктерін көрсету үшін көптеген қасиеттерді пайдаланады:

```
from Tkinter import *
        tk = Tk()
        tv = StringVar()
        Label(tk,
            textvariable=tv,
            relief="groove",
            borderwidth=3,
            font=("Courier", 20, "bold"),
            justify=LEFT,
            width=50,
            padx=10,
            pady=20,
            takefocus=False,
            ).pack()
        Entry(tk,
            textvariable=tv,
            takefocus=True,
            ).pack()
        tv.set("123")
        tk.mainloop()
```

Нәтижесінде экранда көруге болады:



Сурет 14 – Графикалық интерфейс

Виджеттер жасау кезінде теңшеледі. Сонымен қатар, виджеттер аттармен байланыспайды, оларды тек виджет-терезе ішінде орналастырады. Бұл мысалда textvariable (мәтіндік айнымалы), relief (рельеф), borderwidth (шекараның ені), justify (туралау), width (ені, таныстарда), padx және pady (виджет шегі мен мазмұн арасындағы пикселдегі қабат), takefocus (Tab пернесін басқан кезде фокусты қабылдау мүмкіндігі), font (қаріп, оның тапсырмасының бір тәсілі) қасиеттері қолданылған. Бұл қасиеттер көптеген виджеттер үшін өте типтік, бірақ кейде өлшем бірліктері өзгеше болуы мүмкін, мысалы, Canvas виджеті үшін ені танысу емес пикселде беріледі.

Келесі мысалда фонға, алдыңғы жоспарға (мәтінге), виджетті (шекараны жарықтандыру) белсенді күйде және фокус болмаған кезде бөлу мүмкіндіктері көрсетіледі:

```
from Tkinter import *
```

```
tk = Tk()
tv = StringVar()
Entry(tk,
    textvariable=tv,
    takefocus=True,
    borderwidth=10,
    ).pack()
mycolor1 = "#%02X%02X%02X" % (200, 200, 20)
Entry(tk,
    textvariable=tv,
    takefocus=True,
    borderwidth=10,
    foreground=mycolor1,            # fg, текст виджета
    background="#0000FF",           # bg, фон виджета
    highlightcolor='green',         # подсветка при фокусе
    highlightbackground='red',      # подсветка без фокуса
    ).pack()
tv.set("123")
tk.mainloop()
```

Қаласаңыз, барлық виджеттер үшін мәнерлі опцияларды бірден орнатуға болады: tk_setPalette () әдісі арқылы. Бұл әдіс бойынша жоғарыда пайдаланылған сипаттардан басқа, selectForeground және select Background (алдыңғы жоспар және бөлу фоны), selectColor (таңдалған күйдегі түс, мысалы, Checkbutton), insertBackground (кірістіру нүктесінің түсі) және т.б. пайдалануға болады.

Ескерту:

Енгізу өрісінен get () әдісі арқылы да мәнді алуға болады. Мысалы, Entry класының нысанын E деп атасаңыз, мәнді келесідей алуға болады: e.get(). Рас, бұл әдіс Text пішімделген мәтін үшін сынып даналарының get() әдісі сияқты икемділікке ие емес: тек барлық мәнді толығымен алуға болады.

2.6 Нысандар және class

Python объектілі-бағдарланған бағдарламалау парадигмасын қолдайды, яғни бағдарлама компоненттерін сыныптар ретінде анықтай аламыз.

Класс - нысанның үлгісі немесе ресми сипаттамасы, және нысан осы класстың данасын, оның нақты нұсқасын білдіреді. Мынадай ұқсастығы бар: бізде адам туралы бірнеше идея бар - екі қолдың, екі аяқтың, бастың, ас қорытудың, жүйке жүйесінің, мидың және т.б. болуы.

Код бойынша сынып белгілі бір тапсырманы орындайтын функциялар мен айнымалылар жиынтығын біріктіреді. Класс функциялары әдістер деп те аталады. Олар сыныптың мінез-құлқын анықтайды. Класс айнымалысы атрибуттар деп аталады, олар сыныптың күйін сақтайды[12].

Сынып class кілт сөзі арқылы анықталады: class сынып аты: сынып әдістемесі.

```
class Point:
    #нукте класы
```

Яғни экранға шығару үшін атрибут қажет болады

```
class Point:
p=Point()
x=5
y=5
print('(%g,%g)' %
(point.x,point.y))
```

Форматтау операторы

```
class Point:
        p=Point()
p.x=5
p.y=10
print('(%d,%d)' % (p.x,p.y))
```

Бір нүктенің қашықтығын анықтау

main.py saved

```
1   import math
2   class Point:
3
4   Point.x=10
5   Point.y=5
6
7   distance=math.sqrt(p.x**2
    +p.y**2)
8   print(distance)
```

Бөлек функция белгілі бір нүктені қабылдайтын екі мәнді бөліп шығаратын атрибута арқылы

main.py saved

```
1
2   class Point:
3       '''Nukte klasy'''
4   def print_point(p):
5   |
6   |
7   |  print('(%d,%d)'%(p.x,p.y))
```

```
p=Point()
p.x=1
p.y=5
print_point(p)
(1,5)
```

Келесі мысалдар тіктөртбұрышты құру класста. Екі класс құрамыз сонынан бастапқы нүктге мән сосын ені мен ұзындығын енгіземіз. Бірінші мақсат бастапқы нүктенің координатасы табу.
Орта нүктенің координатасы табу

```
class Point:
    '''Nukte klasy'''
def print_point(p):
  print('(%d,%d)'%(p.x,p.y))
class Restangle:
  '''zkjfdnvfdkjnvfkjd'''
def find_center(r):
  p=Point()
  p.x=(r.start.x+r.width)/2
  p.y=(r.start.y+r.height)/2
  return p
p=Point()
p.x=5
p.y=10
print_point(p)
r=Restangle()
r.width=50
r.height=30
r.start=p
```

Dog классын жасаймыз :

```
class Dog():
age=0
name=""
weight=0
def bark(self):
print (self.name, " говорит гав")
myDog = Dog()
myDog.name="Spot"
myDog.weight=20
myDog.age=1
myDog.bark()
```

Программа нәтижесі:

```
Spot говорит гав
>>>
```

Python қателерімен танысу

Python-да ең жиі кездесетін қателіктерімен танысамыз.

Мәселе: SyntaxError - катесі
мысалы:
```
>>> a = input()
hello world
  File "<string>", line 1
    hello world
               ^
SyntaxError: unexpected EOF while parsing
```

Себебі: сіз Python 2 бағдарламасын қостыңыз.
Шешімі: Python 3 бағдарламасын қосыңыз.

Немесе:
```
name = raw_input()
print name
```
қате:
File "a.py", line 3

```
print name
            ^
SyntaxError: invalid syntax
```

Себебі: Python-2
шешуі: Python-2 мен Python-3 айырмашылықтарын оқып шығып Python-3 бағдарламаны қайта жазыныз
дұрысы:

```
name = input()
print(name)
```

Немесе: код мысалы:

```
1 a = 5
2 if a == 5
3     print('Ура!')
```

қате:

```
File "a.py", line 2
if a == 5
         ^
SyntaxError: invalid syntax
```

себебі: қос нүкте ұмытылып кетті
дұрысы:

```
a = 5
if a == 5:
print('Ура!')
```

Немесе: код мысалы:

```
1 a = 5
2 if a = 5:
3 print('Ура!')
```

қате:

```
File "a.py", line 2
if a = 5
     ^
SyntaxError: invalid syntax
```

себебі: сіз теңдік белгісің қоюға ұмытып кетіпсіз
мысал шешуі:

```
a = 5
if a == 5:
print('Ура!')
```

мәселе: TypeError:
код мысалы:

```
>>> a = input() + 5          8
```

```
Traceback (most recent call last):
        File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

себебі: номерді ж/е жолды қосуға болмайды
шешуі: int() функциясын қолдана отырып санды жолға айналдырыңыз. Input() функциясы әрқашан жолды қайтарады!
дұрысы:

```
>>> a = int(input()) + 5          // 8
>>> a                             //  13
```

Мәселе: NameError: name ‘a’ is not defined.
Код мысалы:

```
print (a)
```

Себебі: «a» айнымалысы жоқ.
Шешімі:

```
a = 10
print (a)
```

Мәселе: IndentationError: expected an indented block.
Код мысалы:

```
a = 10
if a > 0:
print(a)
```

Себебі: Шегініс қажет.
Шешуі:

```
a = 10
if a > 0:
        print(a)
```

Мәселе: TabError:
Код мысалы:

```
a = 10
if a > 0:
        print(a)
        print('Ура!')
```

Қате:

```
File "a.py", line 5
      print('Ура!')
        ^
TabError: inconsistent use of tabs and spaces in indentation
```

Себебі: шегіністердегі бос орындар мен қойындыларды араластыру.
Шешімі:.

```
a = 10
if a > 0:
        print(a)
        print('Ура!')
```

Мәселе: UnboundLocalError:
код мысалы:

```
def f():
        a += 1
        print(a)
a = 10
f()
```

қате:

```
Traceback (most recent call last):
  File "a.py", line 7, in <module>
      f()
  File "a.py", line 3, in f
      a += 1
UnboundLocalError: local variable 'a' referenced before assignment
```

қате себебі: сіз әлі жасалмаған жергілікті айнымалыға қол жеткізу әрекетін жасамақшы болдыныз
шешуі:

```
def f():
        global a
        a += 1
        print(a)
a = 10
f()
```

Мәселе: бағдарлама орындалды, бірақ файлға ештеңе жазылмаған немесе толық бәрі жазылмаған.
Код мысалы:

```
>>> f = open('output.txt', 'w', encoding='utf-8')
>>> f.write('bla')                  // 3
```

Себебі: файл жабылмаған, мәліметтердің бір бөлігі буферде қалуы мүмкін.
Шешімі:

```
>>> f = open('output.txt', 'w', encoding='utf-8')
>>> f.write('bla')   //  3
>>> f.close()
```

Тест тапсырмалары

1 Келесі код не шығарады?

t = (( ))

type (t)

A. <type ‘tuple’>

B. <type ‘int'>

C. <class ‘tuple’>

D. Type error

E. <class ‘Int’>

02. IndexError қандай код шығарады

A. t = {1: 1, 2: 2} print (t[3])

B. t = {1: 1, 2: 2}t [3] = 0

C. t = [1,2, 3] print (t [3])

D. t = (1, 2, 3)t [3] = 3

E. t(3) +=1

03. Келесі код не шығарады?

t = (1, 2)

t = t * 3

print(t)

A. TypeError

B. [3, 6]

C. (1, 2, 1, 2, 1, 2)

D. (3, 6)

E. [(1, 2), (1, 2), (1, 2)]

04. Келесі код не шығарады?

set ([1, 2, 3, 3, 2, 1]) == {1, 2, 3, 2, 1}
A. False
B. True
C. TypeError
D. SyntaxError
E. RuntimeError

05. Қандай код қате бермейді:
A. t ={1, 2, 3}t.remove(4)
B. t ={1, 2, 3}t.discard(4)
C. t = {1, 2, 3}t.difference(4)
D. t = {1, 2, 3}t.difference_update(4)
E. t = {1, 2, 3}t.sub(4)
06. Келесі код не шығарады?
t = {1: 1, 2: 2, 3: 3}
t. values()
A. TypeError
B. dIct_values([1, 2, 3])
C. [1, 2, 3]
D. (1, 2, 3)
E. {1 , 2, 3}

08. Келесі код не шығарады?
t = (0: 1, 1: 2, 2: 3}
any(t), all(t)
A. True, True
B. True, False
C. False, True
D. False, False
E. SyntaxError

09.КЕЛЕСІ КОД НЕ ШЫҒАРАДЫ?
d = {i: i**2 for i in range(3)}
print(d[2])
A. 2
B. KeyError
C. SyntaxError
D. 4
E. TypeError

10. Келесі кодты орындағаннан кейін:
g = (i**2 for i in range(10))
A. type(g)- <class 'list’>
B. type(g)- <class ‘tuple’>
C. type(g)- <class ‘set’>
D. type(g)- <class 'generator’>
E. type(g)- <class 'NoneType'>

11. Python тілі -:
A. динамикалық типтеу
B. статистикалық үлгілеу
C. үйректің түрленуі
D. функционалдық бағдарламалау парадигмасын қолдау
E. объектілі-бағытталған бағдарламалау парадигмасы қолдау

12. Объект құрғаннан кейін, әдіс автоматты түрде шақырылады:
A. __Init__
B. __str__
C. __new__
D. __del__
E. __repr__

13. MyClass класы берілсін:
class MyClass:
name = "no name"
def__init__(self, name) :
self. name = name
def print_name (self) :
print (self . name)
Келесі кодты орындағаннан кейін
my_obj = MyClass ("Alice")
MyClass. name = "Bob"
my_obj . print_name ()

A. no name
B. Alice
C. Bob
D. TypeError
E. RuntimeError

14. Операторды қайта жүктеу < класы үшін magic әдісін іске асыруға болады:
A._eq_(self, other)
B._ne_(self, other)
C._gt_(self, other)
D._It_(self, other)
E._ge_(self, other)

15. Операторды қайта жүктеу // класы үшін magic әдісін іске асыруға болады:
A. _add_(self, other)
B. _sub_(self, other)
C. _muI_(self, other)
D. _div_(self, other)
E. _floordiv_(self, other)

Қайталау бақылау сұрақтары

1. Python тілінде идентификатор дегеніміз не?
2. Python тілінде енгізу операторы
3. Phyton тіліндегі шартсыз қайталау операторлары
4. Phyton тіліндегі таңдау операторлары
5. Phyton тіліндегі шартты оператор
6. Phyton-да қай әдіс көпбұрыш сызады?
7. Python-да графикалық интерфейстерді қамтамасыз етуге арналған кітапхананы атаңыз
8. Тұрақтылар дегеніміз не?
9. Phyton тілінде файлды жабу үшін қандай әдіс қолданылады?
10. Phyton тілінде символдық шаманың ұзындығын анықтау операторы
11. Phyton-да файлдан қажетті орынға өті үшін қандай функциясы қолданылады?
12. Виртуалды орта дегеніміз не?
13. Phyton-да файл ашылғанда тек бос жол көрсететін қандай функциясы қолданылады?
14. Framework дегеніміз не?
15. PYTHON тілі қай жылы пайда болды?
16. PYTHON тілін кім ойлар шығарған?
17. PYTHON тілінде қайталау операторларының түрлері
18. Интерпретатор дегеніміз не?
19. NumPy – кітапханасының қызметі қандай?
20. Айнымалы болып табылады
21. Компилятор– бұл
22. MatPlotLib – кітапханасының қызметі қандай?
23. Python бағдарламалау тілінің IDLE интеграцияланған ортасы неге арналған?
24. Python тілінің шартты операторы қалай жазылады?
25. PyQt – кітапханасының қызметі қандай?
26. Цикл дегеніміз не?
27. Range функциясы не үшін қолданылады?
28. PyGame – кітапханасының қызметі қандай?
29. Django-да веб-серверді шақыру үшін:
30. Continue операторының қызметі -
31. Python-да қандай орнатылған деректер типтері бар?
32. Chaco – кітапханасының қызметі қандай?
33. Django-да қарапайым қосымша құру үшін, қанша модел қосу керек?
34. break операторының қызметі-35. Тізім қалай құрылады?

36. Тізімнің соңына элемент қосатын функция
37. Массив болып табылады
38. Бөлгіштің жеке және қалдық бөлігін табу функциясы
39. Сөздік қалай құрылады?
40. Оператор:
41. Phyton тілінде файл атрибуттары қанша?
42. Тізімнің ұзындығын анықтау функциясы
43. Массив дегеніміз не
44. Сандарды дәрежеге шығару функциясы
45. Теру (кортеж) қалай құрылады?
46. Комментарий:
47. Тізімдегі элементтер санын есептейтін функция
48. Сандарды дөңгелектеу функциясы
49. Питон тілінде бір өлшемді массив элементтерін енгізу

50. Тізімдерді (list) құру жолдарындағы бастапқы екі элементті көрсету
51. Тізімдегі элементтерді сұрыптайтын функция
52. Файл қалай құрылады?
53. Тізімдегі ең кіші элементтерді табатын функция
54. Тізімнің соңына тізім қосатын функция
55. Қандай оператор switch операторынан шығуда қолданылады?
56. Тізімдегі элементтердің орнын ауыстыратын функция
57. Phyton тілінде файлды оқу үшін қандай әдіс қолданылады?
58. Тізімдегі ең үлкен элементтерді табатын функция
59. Unicode-ты қайтару функциясы
60. Тізімдерді (list) құру жолдарындағы екіншіден кейінгі барлық элементті көрсету
61. Питон тілінде бір өлшемді массив элементтерін шығару
62. Тізімдерді (list) құру жолдарындағы кері жолмен көрсету
63. Екі өлшемді массив дегеніміз не?
64. Phyton тілінде Джанго дегеніміз не?
65. Квадрат матрицалар дегеніміз не?
66. Екі өлшемді массивті енгізу операторы
67. Төмендегі функция файлға сөз тіркесін жазады
68. Жолдарды файлға жазу үшін арналған функция болып табылады
69. Екі өлшемді массивті шығару операторы
70. Матрицаны траниспонирлеу дегеніміз не?
71. Phyton тілінде файл көрсеткішінің орнын анықтау үшін қандай әдіс қолданылады?
72. Массивтің бағандарын біріктіретін функция
73. Жолдың ұзындығы анықтайтын функцияны атаңыз?
74. Жолды жалғастыру амалы (Конкатенация)
75. Жолды қадам бойынша кесіп алу амалы
76. ASCII коды бойынша санды символға аудару амалы
77. ASCII коды бойынша символды санға аудару амалы
78. Жолды кіші әріптерге түрлендіру амалы
79. Жолды бас әріптерге түрлендіру
80. Switch операциясынан шығу үшін қай нұсқау қолданылатынын анықта
81. Phyton-да қай әдіс шеңбер сызады?
82. Phyton-да қай әдіс мәтінді шығарады?
83. Phyton-да функцияны қалай шақырады?
84. Phyton-да файлды ашу үшін қандай функциясы қолданылады?
85. Phyton-да қандай қызметші сөз нақты айнымалыны сипаттайды?

86. Phyton-да қай әдіс сызық сызады?
87. Phyton-да қай әдіс тіктөртбұрыш сызады?
88. Phyton-да қай әдіс эллипс сызады?
89. Phyton-да қай әдіс доға сызады?
90. Phyton-да файлды жабу үшін қандай функциясы қолданылады?
91. Phyton тілінде файлды ашу үшін қандай әдіс қолданылады?
92. Айнымалының сипатталуы :
93. Phyton-да файлды оқу және жазу үшін қандай функциясы қолданылады?

Қолданылған әдебиеттер тізімі

1 Сузи Р.А. Язык программирования Python. Учебное пособие. - М.: Интернет Университет информационных технологий, 2017. – 327 с.

2 Марк Лутц. Программирование на Python. Учебник. Тома 1 и 2, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2015. - 992 с.

3 Саммерфилд М. Программирование на Python 3. Подробное Руководство. - Пер. с англ. Киселев А. – М.: Символ-Плюс, 2016. – 608 с.

4 Дронов В.А. Django Практика создания Web-сайтов на Python. Учебник.2016. – 886 с

5 Хахаев И. А. Практикум по алгоритмизации и программированию на Python. Учебное пособие. - М. : Альт Линукс, 2013. - 126 с.

6 Н. Прохоренок. Python3 PyQT разработка приложений. Учебник.- СПб.: Питер, 2016 . – 568 с

7 Эл Свейгарт. Автоматизация рутинных задач с помощью Python. Учебное пособие.– СПб. : ДиасофтЮП, 2011. – 384 с

8 Уэсли Дж. Чан. Python создание приложений. Библиотека профессионала.-СПб: Символ-Плюс, 2015. – 652 с

9 Бизли Д. Python. Подробный справочник. - СПб: Символ-Плюс, 2014. – 794 с

10 Гифт Н. Python в системном администрировании UNIX и Linux. Учебник. – СПб. : Символ-Плюс, 2017. – 685 с

11. Лейнингем И. Освой самостоятельно Python за 24 часа. Учебное пособие. - М.: Издательский дом «Вильямс», 2011. – 273 с

12 Доусон М. Программируем на Python. Учебник.- СПб.: Питер, 2014. - 796 с.

13 http://www.pythonware.com/library/tkinter/introduction/

14 http://pythoncard.sourceforge.net/.

15 http://phaseit.net/claird/comp.lang.python/python_GUI.html