Модуль 1. Python-ға кіріспе

Тақырып 1.1. Python құрылымы.

Жалпы ақпарат

Бұл тарау Python бағдарламасының толық нұсқаулығы емес. Оның жалғыз мақсаты - Python-мен таныс болмасаңыз, сізге жақсы бастау үшін жеткілікті ақпарат беру. Егер сіз басқа компьютер тілін білсеңіз және біз білеміз деп есептейміз, қалғанын жүріп жатқанда алу қиын емес.

Python - 1980-ші жылдардың соңында сценарий тілі ретінде жасалған объектіге бағытталған тіл (атауы британдық телехикаядан алынған, Monty Python's Flying Circus). Python инженерлік ортада басқа тілдер сияқты танымал болмаса да, оның бағдарламалау қауымдастығында айтарлықтай ізбасарлары бар.

Python дамып келе жатқан тіл ретінде қарастырылуы мүмкін, өйткені ол әлі де дамып, нақтылануда. Қазіргі күйінде бұл инженерлік қосымшаларды әзірлеуге арналған тамаша тіл.

Python бағдарламалары машиналық кодқа құрастырылмайды, бірақ интерпретатор арқылы іске қосылады. Интерпретацияланатын тілдің үлкен артықшылығы - бағдарламаларды тез тексеруге және жөндеуге болады, бұл пайдаланушыға бағдарламаның артындағы принциптерге көбірек және бағдарламалаудың өзіне азырақ назар аударуға мүмкіндік береді. Әрбір түзетуден кейін құрастырудың, байланыстырудың және орындаудың қажеті жоқ болғандықтан, Python бағдарламаларын балама Fortran немесе C бағдарламаларына қарағанда әлдеқайда қысқа мерзімде жасауға болады. Теріс жағы, интерпретацияланған бағдарламалар дербес қолданбаларды шығармайды. Осылайша, Python бағдарламасын тек Python интерпретаторы орнатылған компьютерлерде іске қосуға болады.

Python-ның оқу ортасында маңызды негізгі тілдерден басқа артықшылықтары бар:

Python – ашық кодты бағдарламалық жасақтама, яғни ол тегін;

Python барлық негізгі операциялық жүйелер үшін қолжетімді (Linux, Unix, Windows, MacOS және т.б.). Бір жүйеде жазылған программа барлық жүйелерде өзгеріссіз жұмыс істейді.

Python үйрену оңай және көптеген тілдерге қарағанда жеңіл оқылатын кодты құрайды.

Python және оның қосымша кеңейтімдерін орнату оңай.

Python қол жеткізу.

Python интерпретаторын мына жерден жүктеп алуға болады

http://www.python.org/getit

Ол әдетте бағдарламаларды редактордан тікелей іске қосуға мүмкіндік беретін ***Idle*** деп аталатын жақсы код редакторымен бірге келеді. Егер сіз Linux-ты пайдалансаңыз, Python компьютеріңізде орнатылған болуы әбден мүмкін. Жүктеп алу біздің бағдарламаларда қолданатын екі модулді қамтиды: массив операцияларына арналған әртүрлі құралдарды қамтитын ***numpy*** модулі және графикті құруда қолданылатын ***matplotlib*** графикалық модулі.

Python тілі көптеген басылымдарда жақсы құжатталған. Оқулықтар мен мысалдарды әртүрлі веб-сайттардан табуға болады. ***Numpy*** үшін сілтеме:

https://numpy.org/

Matplotlib үшін сілтеме:

http://matplotlib.sourceforge.net/contents.html

Егер сіз тәжірибелі Python бағдарламашысы болғыңыз келсе, онда мына кітапты қарауыңызға болады ***Hans P. Langtangen A Primer on Scientific Programming with Python (Springer- Verlag, 2016)***.

Python негізі

Айнымалылар

Көптеген компьютерлік тілдерде айнымалының аты, тіркелген жад орнында сақталған, берілген типтің мәнін білдіреді. Мән өзгертілуі мүмкін, бірақ типі емес. Python-да олай емес, мұнда айнымалылар динамикалық типте болады. Python интерпретаторымен келесі интерактивті көрініс бұл мүмкіндікті көрсетеді (Пуск→Python 3.10→IDLE):

```
>>> b = 2          # b is integer type
>>> print(b)
2
>>> b = b*2.0   # Now b is float type
>>> print(b)
4.0
```

b = 2 меншіктелуі b атауы мен 2 бүтін мәнінің арасында байланысты жасайды. Келесі оператор b*2.0 өрнегін есептейді және нәтижені b-мен байланыстырады; 2 бүтін санымен бастапқы байланыс жойылады. Енді b өзгермелі нүкте мәнін білдіреді 4.0.

Тор белгісі (#) түсініктеменің басын білдіреді — # және жолдың соңы арасындағы барлық таңбаларды интерпретатор елемейді.

Жолдар(Strings)

Жол – бір немесе қос тырнақшаға алынған таңбалар тізбегі. Жолдар плюс (+) операторымен біріктіріледі, ал кесу (:) жолдың бір бөлігін шығару үшін қолданылады. Міне, мысал:

```
>>> string1 = Сандық әдісті Python '
>>> string2 = 'ортасында жасау. '
>>> print(string1 + '  ' + string2)        # Біріктіру
Сандық әдісті Python ортасында жасау.
>>> print(string1[0:11])                    # Бөліктеу
Сандық әдіс
```

Жолды split командасы арқылы оның құрамдас бөліктеріне бөлуге болады. Құрамдас бөліктер тізімдегі элементтер ретінде көрсетіледі. Мысалға,

```
>>> s = '3 9 81'
>>> print(s.split())       # Бөлгіш – бос орын
['3', '9', '81']
```

Жол өзгермейтін нысан оның жеке таңбаларын тағайындау операторымен өзгерту мүмкін емес және оның тұрақты ұзындығы бар. Өзгермейтіндікті бұзу әрекеті TypeError келесідей нәтиже береді:

```
>>> s = 'Sandyq adis'
>>> s[0] = 's'
TypeError                          Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_3840/986129307.py in <module>
      1 s = 'Sandyq adis'
----> 2 s[0] = 's'
TypeError: 'str' нысаны элемент тағайындауға қолдау көрсетпейді
```

Кортеждер (Tuples)

Кортеж – үтірмен бөлінген және жақшаға алынған ерікті нысандар тізбегі. Кортежде жалғыз нысан болса, соңында үтір қажет; мысалы, x = (2,). Кортеждер жолдар сияқты бірдей амалдармен сипатталады; олар да өзгермейтін нысан. Келесі мысалда rec кортежінде басқа кортеж (16,3,83) бар, қарастырып көрейік:

```
>>> rec = ('Kasenov', 'Syrym',(16,3,83)) # Mynau kortej
>>> Familia, Aty, tugankuni = rec # Kortejdi jaqshadan shygaru
>>> print(Aty)
```

```
Syrym
>>> tuganjyly = tugankuni [2]
>>> print(tuganjyly)
83
>>> name = rec[1] + ' ' + rec[0]
>>> print(name)
Syrym Kasenov
>>> print(rec[0:2])
('Kasenov', 'Syrym')
```

Тізімдер(Lists)

Тізім кортежге ұқсас, бірақ ол өзгермелі, сондықтан оның элементтері мен ұзындығын өзгертуге болады. Тізім оны квадрат жақшаға алу арқылы анықталады. Мұнда тізімдер бойынша орындалатын операциялардың үлгілері берілген:

```
>>> a = [1.0, 2.0, 3.0]          # Тізім құру
>>> a.append(4.0)                # Тізімге 4.0 элементін енгізу
>>> print(a)
[1.0, 2.0, 3.0, 4.0]
>>> a.insert(0,0.0)              # 0 позициясына 0.0 енгізіңіз
>>> print(a)
[0.0, 1.0, 2.0, 3.0, 4.0]
>>> print(len(a))                # Тізім ұзындығын анықтаңыз
5
>>> a[2:4] = [1.0, 1.0, 1.0]     # Таңдалған элементтерді өзгерту
>>> print(a)
[0.0, 1.0, 1.0, 1.0, 1.0, 4.0]
```

Егер a өзгеретін нысан болса, мысалы тізім болса, b = a тағайындау операторы жаңа b нысанын құрмайды, жай ғана a сілтемесін жасайды. Осылайша, b-ге енгізілген кез келген өзгерістер a-да көрсетіледі. a тізімінің тәуелсіз көшірмесін жасау үшін c = a[:] операторын пайдалану керек, осы жайлы келесі мысал:

```
>>> a = [1.0, 2.0, 3.0]
>>> b = a                # 'b'–'a' бүркеншік аты
>>> b[0] = 5.0           # 'b' өзгерту
>>> print(a)
[5.0, 2.0, 3.0]          # Өзгеріс 'a' түрінде көрсетіледі
>>> c = a[:]             # 'c'–'a' тәуелсіз көшірмесі
>>> c[0] = 1.0           # 'c' өзгерту
>>> print(a)
[5.0, 2.0, 3.0]          # 'a' өзгертуге әсер етпейді
```

Матрицаларды кіріктірілген тізімдер ретінде көрсетуге болады, әрбір жол тізімнің элементі болып табылады. Мұнда тізім түріндегі 3 × 3 матрицасы берілген:

```
>>> a = [[1, 2, 3], \
         [4, 5, 6], \
         [7, 8, 9]]
>>> print(a[1]) # Екінші жолды басып шығару (element 1)
[4, 5, 6]
>>> print(a[1][2]) # Екінші жолдың үшінші элементін басып шығару
6
```

Кері қиғаш сызық (\) Python бағдарламасының жалғасы болып табылады. Еске салайық, Python тізбектерінің нөлдік ығысуы бар, сондықтан a[0] бірінші жолды, a[1] екінші жолды және т.б. көрсетеді. Өте аз ерекшеліктермен біз сандық массивтер үшін тізімдерді пайдаланбаймыз. Бұл numpy модулімен қамтамасыз етілген массив нысандарын пайдалану өте ыңғайлы.

Арифметикалық операторлар
Python әдеттегі арифметикалық операторларды қолдайды:

+	Қосу
−	Алу
*	Көбейту
/	Бөлу
**	Дәрежеге шығару
%	Модульдік бөлу

Бұл операторлардың кейбірі жолдар мен тізбектер үшін де төмендегідей анықталған:

```
>>> s = 'Hello '
>>> t = 'to you'
>>> a = [1, 2, 3]
>>> print(3*s)          # Қайталау
Hello Hello Hello
>>> print(3*a)          # Қайталау
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> print(a + [4, 5])   # Элементтерді қосу
[1, 2, 3, 4, 5]
>>> print(s + t)        # Біріктіру
Hello to you
>>> print(3 + s)        # Бұл қосудың мағынасы жоқ
TypeError                    Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_3840/3230724383.py in <module>
----> 1 print(3 + s)        # Бұл қосудың мағынасы жоқ
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Python-да C пайдаланушыларына таныс a+ =b сияқты кеңейтілген тағайындау операторлары бар. Кеңейтілген операторлар мен баламалы арифметикалық өрнектер келесі кестеде көрсетілген.

$a+= b$	$a = a + b$
$a-= b$	$a = a - b$
$a *= b$	$a = a * b$
$a/= b$	$a = a/b$
$a **= b$	$a = a ** b$
$a\% = b$	$a = a\%b$

Салыстыру операторлары
Салыстыру (қатысты) операторлар True немесе False мәнін қайтарады. Бұл операторлар

<	Кем
>	Үлкен
<=	Кем немесе тең
>=	Үлкен немесе тең
==	Тең
! =	Тең емес

Әртүрлі типтегі сандар (бүтін, өзгермелі нүкте және т.б.) салыстыру жүргізілмес бұрын ортақ түрге түрлендіріледі. Әйтпесе, әртүрлі типтегі нысандар тең емес болып саналады. Міне, бірнеше мысал:

```
>>> a = 2                # Бүтін сан
>>> b = 1.99             # Өзгермелі нүкте
>>> c = '2'              # Жол
>>> print(a > b)
True
>>> print(a == c)
False
>>> print((a > b) and (a != c))
True
>>> print((a > b) or (a == b))
True
```

Шартты операторлар(Conditionals)

`if` конструкциясы

```
if шарт:
   блок
```

егер шарт ақиқат болса, блок орындалады (блок жазылғанда бір шегіністе болады). Шарт жалған мәнін қайтарса, блок өткізіп жіберіледі. `if` шартынан кейін бірнеше `elif` тұруы мүмкін («`else if`» дегеннің қысқасы) конструкциясы:

```
elif шарт:
   блок
```

бұл да сол сияқты жұмыс істейді. else тармағы басқашалау:

```
else:
   блок
```

`if-elif` сөйлемдерінің ешқайсысы ақиқат болмаса, орындалатын мәлімдемелер блогын анықтау үшін `else` пайдаланылуы мүмкін. `sign_of_a` функциясының шартты операторды қолданылуын көрсетеді.

```
def sign_of_a(a):
    if a < 0.0:
        sign = 'теріс сан'
    elif a > 0.0:
        sign = 'оң сан'
    else:
        sign = 'нөл'
    return sign
a = 1.5
print('a - ' + sign_of_a(a))
```

Бағдарламаны жүргізсек келесі нәтижені береді:

```
a - оң сан
```

Циклдар(Loops)

`while` конструкциясы

```
while шарт:
   блок
```

шарт ақиқат болса (шегіністелген) операторлар блогын орындайды. Блокты орындағаннан кейін шарт қайтадан бағаланады. Егер ол әлі ақиқат болса, блок қайтадан орындалады. Бұл процесс шарт «жалған» болғанша жалғасады. `else` тармағы

```
else:
  блок
```

шарт жалған болса, орындалатын операторлар блогын анықтау үшін пайдаланылуы мүмкін. Мұнда тізімді жасайтын мысал берілген $[1, 1/2, 1/3, \dots]$:

```
nMax = 5
n = 1
a = [] # Бос тізім жасаңыз
while n < nMax:
    a.append(1.0/n) # Элементті тізімге қосыңыз
    n = n + 1
print(a)
```

Бағдарлама нәтижесі

```
[1.0, 0.5, 0.3333333333333333, 0.25]
```

for операторына мақсат және мақсатқа жетуге цикл өтетін тізбек талап етеді. Бұл форманың конструкциясы

```
for мақсат in тізбек:
  блок
```

for циклі аяқталғаннан кейін орындалатын else сөйлемін қосуға болады. Алдыңғы бағдарламаны for операторымен былай жазуға болады

```
nMax = 5
a = []
for n in range(1,nMax):
    a.append(1.0/n)
print(a)
```

Мұндағы n – мақсат, ал диапазоны $[1, 2, \dots, nMax - 1]$ (`range` функциясын шақыру арқылы жасалған) – тізбек.

Кез келген цикл

```
break
```

Оперторы арқылы аяқталуы мүмкін. Егер циклмен байланысты `else` болса, ол орындалмайды. Тізімдегі атауды іздейтін келесі бағдарлама `break` және `else` операторын `for` циклімен бірге пайдалануды көрсетеді:

```
list = ['Jack', 'Jill', 'Tim', 'Dave']
name = eval(input('Type a name: ')) # Python енгізу сұрауы
for i in range(len(list)):
    if list[i] == name:
        print(name,'is number',i + 1,'on the list')
        break
else:
    print(name,'is not on the list')
```

Міне, екі іздеу нәтижелері:

```
Type a name: 'Tim'
```

```
Tim is number 3 on the list

Type a name: 'June'
June is not on the list
```

`continue`

операторы қайталанатын циклдің бір бөлігін өткізіп жіберуге мүмкіндік береді. Егер интерпретатор `continue` операторымен кездессе, ол одан кейінгі командаларды орындамай-ақ бірден циклдің басына оралады. Келесі мысал 7-ге бөлінетін 1 мен 99 арасындағы барлық сандардың тізімін құрастырады.

```
x =  []                    # Бос тізім жасаңыз
for i in range(1,100):
    if i%7  != 0:
        continue           # 7-ге бөлінбесе, циклдің қалған бөлігін өткізіп жіберіңіз
    x.append(i)            # Тізімге і қосыңыз
print(x)
```

Бағдарлама басып шығарады
```
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98]
```

Түрлендіру типі

Егер арифметикалық амалдарды орындау кезінде аралас типті сандар болса, амалдар орындалмас бұрын сандар автоматты түрде жалпы типке түрлендіріледі.

Типке түрлендірулеріне келесі функциялар арқылы да қол жеткізуге болады:

`int(a)`	a бүтін санға түрлендіреді
`float(a)`	a мәнін өзгермелі нүктеге түрлендіреді
`complex(a)`	$a + 0j$ комплекс санына түрлендіреді
`complex(a,b)`	$a + bj$ комплекс санына түрлендіреді

Бұл функциялар жолдарды сандарға түрлендіру үшін де жұмыс істейді, егер жолдағы литерал жарамды санды білдірсе. Өзгермелі нүкте санын бүтін санға түрлендіру дөңгелектеу арқылы емес, кесу арқылы жүзеге асырылады. Міне, бірнеше мысал:

```
>>> a = 5
>>> b = -3.6
>>> d = '4.0'
>>> print(a + b)
1.4
>>> print(int(b))
-3
>>> print(complex(a,b))
(5-3.6j)
>>> print(float(d))
4.0
>>> print(int(d)) # Бұл сәтсіз аяқталады: d - жол
ValueError                                Traceback (most recent call last)
<ipython-input-11-8855b2389f81> in <module>
----> 1 print(int(d)) # Бұл сәтсіз аяқталады: d - жол

ValueError: invalid literal for int() with base 10: '4.0'
```

Математикалық функциялар
Core Python тек келесі математикалық функцияларды қолдайды:

`abs(a)`	a санының абсолют мәні
`max(sequence)`	тізбектің ең үлкен элементі
`min(sequence)`	тізбектің ең кіші элементі
`round(a,n)`	a саннын үтірден n орынға дейін дөңгелектеу
`cmp(a,b)`	$\begin{cases} -1, \text{егер } a < b \\ 0, \text{егер } a = b \\ 1, \text{егер } a > b \end{cases}$ мәнін қайтарады.

Математикалық функциялардың көпшілігі `math` модулінде қолжетімді.

Енгізуді оқу(Reading Input)
Пайдаланушы енгізуін қабылдауға арналған ішкі орнатылған функция

```
input(prompt)
```

ол шақыруды көрсетеді, содан кейін жолға түрлендірілетін енгізу жолын оқиды. Жолды сандық мәнге түрлендіру үшін

```
eval(string)
```

функцияны пайдаланылады. Келесі бағдарлама осы функцияларды пайдалануды көрсетеді:

```
a = input('a енгізіңіз: ')
print(a, type(a)) # a және оның типін басып шығару
b = eval(a)
print(b,type(b)) #  b және оның типін басып шығару
```

`type(a)` функциясы a нысанының типін қайтарады; бұл бағдарламаны орындауға өте пайдалы құрал. Бағдарлама келесі нәтижелермен екі рет орындалды:

```
a енгізіңіз: 10.0
10.0 <class 'str'>
10.0 <class 'float'>

a енгізіңіз: 11**2
11**2 <class 'str'>
121 <class 'int'>
```

Санды енгізудің және оны a айнымалысына тағайындаудың ыңғайлы жолы

```
a=eval(input(prompt))
```

Нәтижені басып шығару(Printing Output)
Нәтижені басып шығару келесі функциясымен көрсетуге болады

```
print(нысан1,нысан2,...)
```

ол нысан1, нысан2 және т.б. жолдарға түрлендіреді және оларды бос орындармен бөлінген бір жолда басып шығарады. Жаңа жолдың '\n' таңбасы жаңа жолға міндеттеп басып шығару үшін пайдаланылуы мүмкін. Мысалы,

```
>>> a = 1234.56789
>>> b = [2, 4, 6, 8]
>>> print(a,b)
```

```
1234.56789 [2, 4, 6, 8]
>>> print('a =',a, '\nb =',b)
a = 1234.56789
b = [2, 4, 6, 8]
```

`print` функциясы әрқашан жолдың соңына жаңа жол таңбасын қосады. `end` кілт сөзінің аргументін пайдалану арқылы '\n' таңбаны алмастыра аламыз.

Мысалы,

```
print(нысан1,нысан2,...,end='')
```

\n бос орынмен ауыстырады.

Нәтиже format әдісімен форматталауы мүмкін. Түрлендіру операторының ең қарапайым түрі

```
'{:fmt1}{:fmt2}...'.format(arg1,arg2,...)
```

мұндағы fmt1, fmt2,. . . шамалары arg1, arg2, ... форматтары үшін спецификациялары болып табылады сәйкесінше. Әдетте пайдаланылатын форматтар спецификациялары

*w*d	Бүтін сан
w.df	Өзгермелі нүкте белгісі
w.de	Көрсеткіштік белгілеу

мұндағы *w* - өрістің ені және *d* - ондық үтірден кейінгі цифрлар саны.

Нәтиже көрсетілген өрісте оң жақ шет бойынша реттеледі және бос орындармен толтырылады (реттеуді өзгерту мен толтырудың шарттары бар). Мұнда бірнеше мысал келтірілген:

```
>>> a = 1234.56789
>>> n = 9876
>>> print('{:7.2f}'.format(a))
1234.57
>>> print('n = {:6d}'.format(n)) # Бос орындармен толтыру
n = 9876
>>> print('n = {:06d}'.format(n)) # Нөлдермен толтыру
n =009876
>>> print('{:12.4e} {:6d}'.format(a,n))
1.2346e+03 9876
```

Файлды ашу және жабу

Сақтау құрылғысындағы деректер файлына (мысалы, диск) қол жеткізу үшін алдымен файл нысанын келесі команда арқылы жасау керек

```
file_object = open(файл аты, әрекет)
```

мұндағы `файл аты` – ашылатын файлды көрсететін жол және `әрекет` келесі жолдардың бірі болып табылады:

'r'	Бар файлдан оқу.
'w'	Файлға жазу. Егер `файл атауы` жоқ болса, ол жасалады.
'a'	Файлдың соңына қосыңыз.
'r+'	Бар файлды оқу және одан жазу.
'w+'	'r+' сияқты, бірақ `файл аты` жоқ болса жасалады.
'a+'	'w+' сияқты, бірақ деректер файлдың соңына қосылады.

Файлға кіру қажет болмаған кезде оны жабу жақсы бағдарламалау тәжірибесі болып табылады. Мұны келесі әдіспен жасауға болады

```
file_object.close()
```

Файлдан деректерді оқу

Файлдан деректерді оқудың үш әдісі бар. Әдіс

```
file_object.read(n)
```

n таңбаны оқиды және оларды жол ретінде қайтарады. Егер *n* алынып тасталса, файлдағы барлық таңбалар оқылады.

Тек ағымдағы жолды оқу керек болса, онда келесіні пайдаланыңыз

```
file_object.readline(n)
```

ол жолдан *n* таңбаны оқиды. Таңбалар \n жаңа жол таңбасымен аяқталатын жол болып келеді. *n*-ді түсіріп алу бүкіл жолды оқуға әкеледі.

Файлдағы барлық жолдарды оқуға болады

```
file_object.readlines()
```

Бұл жолдардың тізімін қайтарады, әрбір жол жаңа жол таңбасымен аяқталатын файлдан алынған жол болып табылады.

Барлық жолдарды бір-бірден шығарудың ыңғайлы әдісі - циклды пайдалану

```
for line in file_object:
    бірнәрсе жазылған line
```

Мысал ретінде жұмыс каталогында `sunspots.txt` атты файл бар деп есептейік. Бұл файлда күн дақтарының қарқындылығының күнделікті деректері бар, әр жолдың келесідей форматта (жыл/ай/күн/қарқындылық):

```
1896 05 26 40.94
1896 05 27 40.58
1896 05 28 40.20
      т.б.
```

Біздің міндетіміз – файлды оқу және тек қарқындылықты қамтитын *x* тізімін жасау. Файлдағы әрбір жол жол болғандықтан, алдымен `split` командасы арқылы жолды бөліктерге бөлеміз. Бұл ['1896','05','26','40.94'] сияқты жолдар тізімін жасайды. Содан кейін біз қарқындылықты (тізімнің [3] элементін) шығарып, оны бағалаймыз және нәтижені *x*-ке енгіземіз. Мынау алгоритмі:

```
x = []
data = open('sunspots.txt','r')
for line in data:
    x.append(eval(line.split()[3]))
data.close()
```

Деректерді файлға жазу

```
file_object.write(string)
```

әдісі жолды файлға жазады, ал

file_object`.write(`*list_of_strings*`)`

жолдар тізімін файлға жазу үшін қолданылады. Ешбір әдіс жолдың соңына жаңа жол таңбасын қоспайды.

Мысал ретінде, $k = 101$-ден 110-ға дейін k және k^2 форматталған кестесін *testfile* файлына жазайық. Файлға жазуды жүзеге асыратын бағдарлама:

```
f = open('testfile.txt','w')
for k in range(101,111):
    f.write('{:4d} {:6d}'.format(k,k**2))
    f.write('\n')
f.close()
```

'testfile.txt мазмұны мыналар

```
 101  10201
 102  10404
 103  10609
 104  10816
 105  11025
 106  11236
 107  11449
 108  11664
 109  11881
 110  12100
```

`print` функциясын нәтижені файл нысанына қайта бағыттау арқылы файлға жазу үшін де пайдалануға болады:

print(object1,object2,...,file = file_object)

Қайта бағыттаудан басқа, бұл әдеттегі `print` функциясы сияқты жұмыс істейді.

Қатені басқару

Бағдарламаны орындау кезінде қате орын алса, ерекше жағдай туындайды және бағдарлама тоқтайды. Ерекшеліктерді try және except операторларымен ұстауға болады:

```
try:
    бірдеңе жазылады
except error:
    басқа нәрсе жазылады
```

мұндағы *error* –Python ішінде орналасқан ерекшеліктің атауы. Ерекшелік қатесі орындалмаса, `try` блогы орындалады; қарсы жағдайда орындау `except` блокқа өтеді. Барлық ерекшеліктерді `except` операторынан қатені қалдыру арқылы ұстауға болады.

Келесі оператор ZeroDivisionError ерекшелігін тудырады:

```
>>> c = 12.0/0.0
ZeroDivisionError                 Traceback (most recent call last)
<ipython-input-30-d424c91496c1> in <module>
----> 1 c = 12.0/0.0

ZeroDivisionError: float division by zero
```

Бұл қатені ұстауға болады

```
try:
    c = 12.0/0.0
```

```
except ZeroDivisionError:
    print('Division by zero')
```