2024

Pythonئۇيغۇرچە دەرسلىك قوللانمىسى

1. Python نىڭ كېلىپ چىقىشى :

python ئىنگلىزچە سۆز بولۇپ، « بوغما يىلان» دېگەن مەنىدە.

گوللاندىيەلىك Guido van Rossum بۇرۇن Abc ئوقۇتۇش تىلىنى لايىھەلەشكە قاتناشقان بولۇپ، ئۇنىڭ سۆزىگە قارىغاندا ، ABC ئىنتايىن گۈزەل ۋە كۈچلۈك تىل بولۇپ ، ئۇ كەسىپي بولمىغان پروگراممىرلار ئۈچۈن ئالاھىدە لايىھەلەنگەن. لىكىن ABC تىلى مۇۋەپپەقىيەت قازىنالمىغان. گۈيدو بۇنىڭ سەۋەبىنى ئوچۇق ـ ئاشكارە بولمىغانلىقتىن دەپ قارىغان.

شۇڭا 1991-يىلى گۈيدو python نىڭ بىرىنجى تەرجىمالىنى يېزىپ چىققان.

ئۇ C تىلىدا يولغا قويۇلغان بولۇپ ، C تىلىدا كۈتۈپخانا ھۆججىتىنى چاقىرىشقا بولىدۇ.

1.1 تەرجىمال :

كومپيۇتېر ماشىنا تىلىدىن باشقا ھەرقانداق تىلنى بىۋاستە چۈشىنەلمەيدۇ ، شۇڭا پروگراممېر يازغان پروگرامما ئىجرا قىلىنىشتىن بۇرۇن چوقۇم ماشىنا تىلىغا تەرجىمە قىلىنىشى كېرەك. باشقا تىللارنى ماشىنا تىلىغا تەرجىمە قىلىدىغان قورال **تەرجىمال** دەپ ئاتالغان

2. نېمىشقا Python نى ئىشلىتىمىز؟

ئوخشاش مەسىلە ئوخشىمىغان تىللاردا ھەل قىلىنسا ، كود مىقدارىدا چوڭ پەرق بار. ئادەتتە Python دا يېزىلغان كود Java نىڭ 1/5 قىسمىغا باراۋەر.

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

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

3. Python نىڭ ئالاھىدىلىكى:

- Python پۈتۈنلەي ئوبيېكتقا يۈزلەنگەن تىل،
- فونكسىيە ، مودۇل ، سان ، string ھەممىسى ئوبيېكت ، يەنى Python دىكى ھەممە نەرسە ئوبيېكت.
- ۋاربسلىق قىلىش، قايتا يۈكلەش، كۆپ خىل ۋاربسلىق قىلىش قاتارلىقلارنى پۈتۈنلەي قوللايدۇ.
- Python نىڭ كۈچلۈك ئۆلچەملىك كۈتۈپخانىسى بار ، Python تىلىنىڭ يادروسى پەقەت سان ، string ، تىزىملىك ، لۇغەت ۋە ھۆججەت قاتارلىق كۆپ ئۇچرايدىغان تىپ ۋە ئىقتىدارلارنى ئۆز ئىچىگە ئالىدۇ. Python ئۆلچەملىك كۈتۈپخانىسى سىستېما باشقۇرۇش ، تور ئالاقىسى ، تېكىست بىر تەرەپ قىلىش ، ساندان ئارايۈزى ۋە گرافىك. سىستېما ، XML بىر تەرەپ قىلىش قاتارلىقلارنى تەمىنلەيدۇ.
- Python مەھەللىسى نۇرغۇنلىغان ئۇچىنچى تەرەپ مودۇلى بىلەن تەمىنلەيدۇ ، ئۇنى ئۆلچەملىك كۈتۈپخانىغا ئوخشاش ئۇسۇلدا ئىشلىتىشكە بولىدۇ. ئۇلارنىڭ ئىقتىدارى ئىلمىي ھېسابلاش ،

سۈنئي ئىدراك ، ماشىنا ئۆگىنىش ، تور ئېچىش ، ساندان كۆرۈنمە يۈزى ، گرافىك سىستېمىسى قاتارلىقلارنى ئۆز ئىچىگە ئالىدۇ.

3.1 ئوبيېكتقا يۈزلەنگەن:

ئوبيېكتقا يۈزلىنىش بولسا ئۇ بىر خىل تەپەككۇر ئۇسۇلى ۋە پروگراممىا تۈزۈش تېخنىكىسى.
بىر مەسىلىنى ھەل قىلىشتىن بۇرۇن ، ئالدى بىلەن كىمنىڭ قىلىدىغانلىقىنى ، كىمنىڭ قانداق قىلىشقا مەسئۇل ئىكەنلىكىنى بىكىتىش لازىم. بۇ يەردىكى ئوبيېكت " كىم " نى كۆرسىتىدۇ.

4. Python نىڭ ئەۋزەللىكى:

- ئاددىي ۋە ئۆگىنىش ئاسان
- ھەقسىز ، ئوچۇق مەنبە
- ئوبيېكتقا يۈزلەنگەن
- مول كۇتۇپخانا

5. Python نىڭ كەمچىلىكى:

- ئىجرا قىلىش سۈرئىتى باشقا تىللارغا قارىغاندا نىسپەتەن ئاستا، ئەمما ھازىرقى كومپيوتۇرلارنىڭ سەپلىمىسىنىڭ ئىلغارلىقى سەۋەبلىك باشقا تىللار بىلەن بولغان پەرقنى بىۋاستە ھىس قىلغىلى بولمايدۇ.
- ئۈيغۇرچە ماتېرىيال بەك ئاز.

1. python نى قاچىلاش

/https://www.python.org/downloads

2. pycharm نى قاچىلاش

/https://www.jetbrains.com/pycharm

تۇنجى Hello Python پروگراممسى:

Python مەنبە پروگراممسى ئالاھىدە فورماتتىكى تېكست ھۆججىتى بولۇپ ، ھەر قانداق تېكست تەھرىرلەش يۇمشاق دېتالى ئارقىلىق Python پروگراممسنى يېزىشقا بولىدۇ.

Python پروگراممىلىرىنىڭ ھۆججەت كېڭەيتىلمىسى ئادەتتە *.py** بۇ شەكىلدە بولىدۇ.

```
print("hello python")
print("hello world")
```

print بولسا python دىكى بىز ئۇگەنگەن بىرىنجى فونكسىيە،

print ئىقتىدارى بولسا "***" نىڭ ئىچىدىكى مەزمۇننى ئىكران يۈزىگە بىسىپ چىقىرىش.

خاتالىقلارنى چۈشىنىش (BUG):

BUG دىگىنىمىز كومپيۇتېر پروگراممسى ياكى سىستېمىسىدىكى خاتالىق ياكى كاشىلا. بۇ خاتالىقلار پروگراممىنىڭ بۇزۇلۇشنى ، خاتا نەتىجە ياكى باشقا بىنورمال ھەرىكەتلەرنى كەلتۈرۈپ چىقىرىشى مۇمكىن.

- قول خاتالىقى:

مەسىلەن pirnt("Hello world")

NameError: name 'pirnt' is not defined

- بىرقانچە print كودىنى بىرقۇرغا تەڭ يېزىش:

SyntaxError: invalid syntax

- بوشلۇق خاتالىقى:

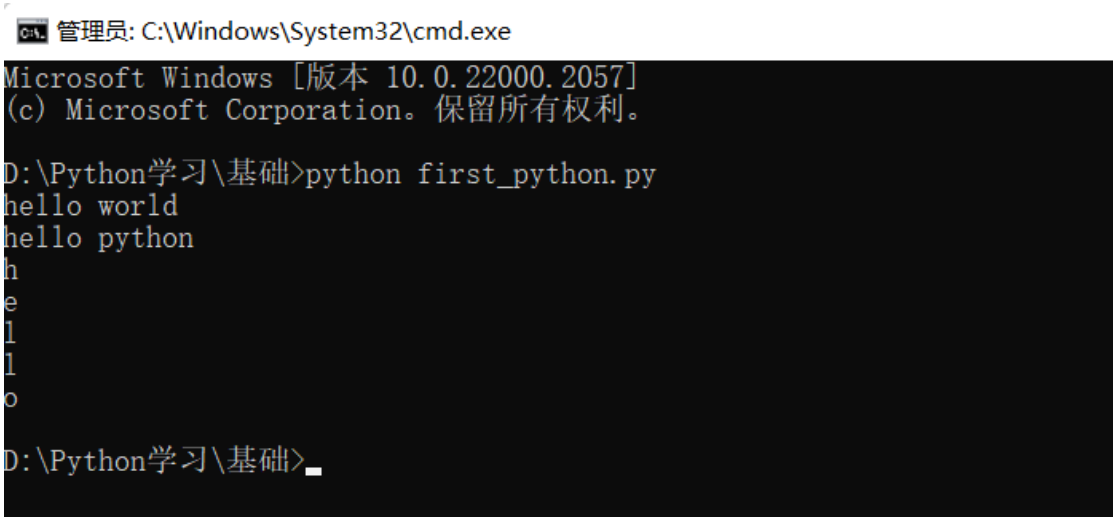
IndentationError: unexpected indent

Python ئىنتايىن قاتتىق فورماتلانغان پروگراممۇ تىلى، يەنى بوشلۇقنى قويىدىغان يەرگە قويماي قويمايدىغان يەرگە قويساق، پروگراممدا خاتالىق كۆرىلىدۇ.

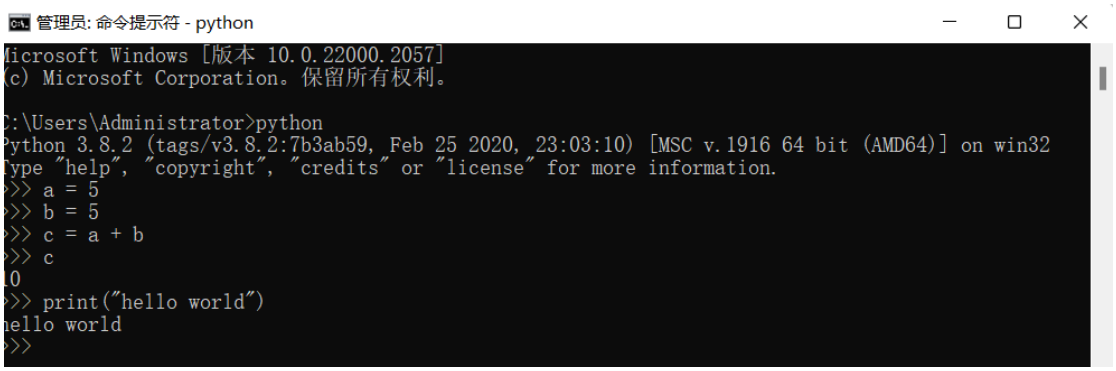
python پروگراممىسىنى ئىجرا قىلىشنىڭ ئۇچ خىل شەكلى:

1. dos كۆزنىكىدە ھازىر بولغان python ئىجرا قىلىش:

python **.py**



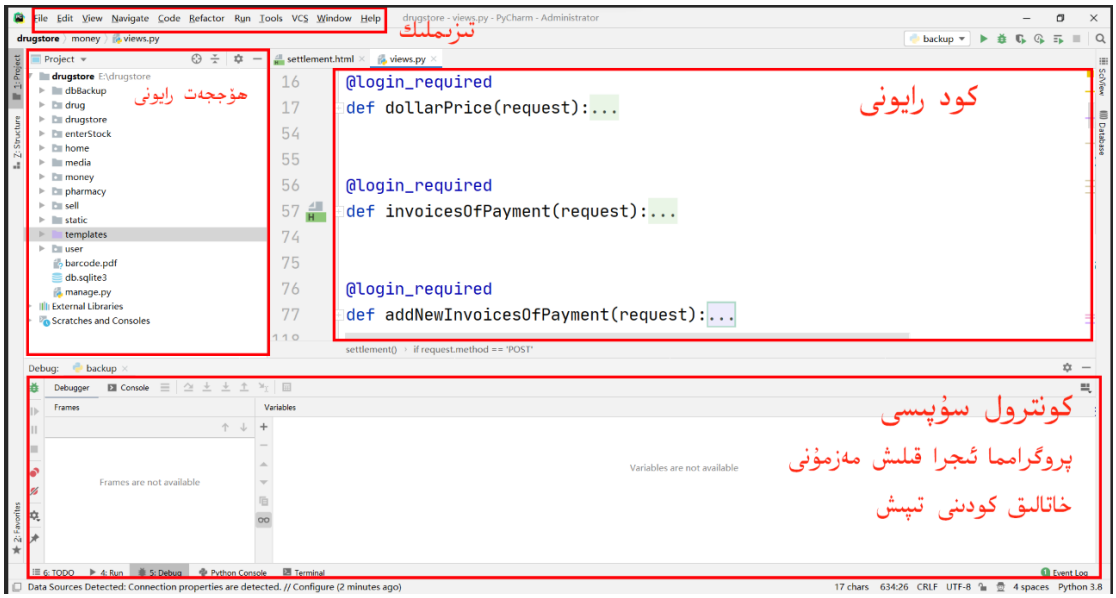
2. dos كۆزنىكىدە python پروگراممىسى يېزىش(كىچىك پروگراممۇ):



2. pycharm دا python پروگراممىسى يېزىش:

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Python学习 〉 基础 〉 first_python.py
first_python.py
1  print('hello world')
2  print('hello python')
3
4  names = 'hello'
5
6  for name in names:
7      print(name)
```

pycharm نىڭ چۈشەندۈرۈلىشى:



يەككە قۇرلۇق ئىزاھلاش:

دىن باشلاپ # نىڭ ئوڭ تەرىپىدىكى ھەممە نەرسە ئىجرا قىلىنمىغان پروگراممىا ئەمەس ، بەلكى ئىزاھلاش تېكىستى دەپ قارىلىدۇ ، ئۇ پەقەت ئىزاھلاش رولىنى ئويناىدۇ.

بۇ ئارقىلىق پروگراممىنىڭ ئوقۇشچانلىقىنى يۇقىرى كۆتۈرگىلى بولىدۇ.

first_python.py

```
1  # بىرىنجى پروگراممىا
2  print('hello world')
3  print('hello python')
4
```

كۆپ قۇرلۇق ئىزاھلاش:

ئەگەر ئىزاھلايدىغان تېكست بىر قۇرغا پاتمىسا، كۆپ قۇرلۇق ئىزاھلاش ئۇسۇلىنى قوللىنىشقا بولىدۇ

first_python.py

```
1  """
2  ئەگەر ئىزاھلايدىغان تېكست بىر قۇرغا پاتمىسا،
3  كۆپ قۇرلۇق ئىزاھلاش ئۇسۇلىنى قوللىنىشقا بولىدۇ.
4  """
5
6  print("كۆپ قۇرلۇق ئىزاھلاش")
7
```

ماتېماتىكىلىق ھېسابلاش بەلگىلىرى:

بەلگە	مەنىسى	مىسال
+	قوشۇش	10 + 20 = 30
-	ئېلىش	50 - 30 = 20
*	كۆپەيتىش	20 * 30 = 600
/	بۆلۈش	10 / 20 = 0.5
//	پۈتۈن بۆلۈش	9 // 2 = 4
%	قالدۇق سانىنى تېپىش	9 % 2 = 1
**	دەرىجە	2 ** 3 = 8

سلىشتۇرۇش بەلگىلىرى:

بەلگە	مەنىسى
==	بەلگىنىڭ ئوڭ سول تەرىپى تەڭ بولسا، True قايتىدۇ
!=	بەلگىنىڭ ئوڭ سول تەرىپى تەڭ بولمىسا، True قايتىدۇ
>	بەلگىنىڭ سول تەرىپى ئوڭ تەرىپىدىن چوڭ بولسا، True قايتىدۇ
<	بەلگىنىڭ سول تەرىپى ئوڭ تەرىپىدىن كىچىك بولسا، True قايتىدۇ

مەسىلەن:

```
print(7==7)  → True    print(7==9)  → False
print(7!=3)  → True    print(7!=7)  → False
print(9 > 7)  → True   print(7 > 9)  → False
print(7 >= 7) → True   print(7 >= 9) → False
```

لوگىكىلىق ھىسابلاش:

لوگىكا	مىسال	مەنىسى
and	x and y	x,y ھەممىسى True بولسا، ئاندىن True قايتىدۇ
or	x or y	پەقەت x,y نىڭ بىرسى True بولسا، True قايتىدۇ
not	not x	ئەگەر x True بولسا، False قايتىدۇ

مەسىلەن:

```
x = True        y = False
print(x and y)  → False
print(x or y)   → True
print(not x)    → False
```

قىممەت بېرىش بەلگىلىرى:

بەلگە	مىسال	مەنىسى
+=	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
**=	a **= b	a = a ** b

ماتىماتىكىلىق ھىسابلاش تەرتىپلىرى:

- ئاۋال كۆپەيتىش بۆلۈش، ئاندىن قوشۇش ئىلىش.
- سولدىن ئوڭغا قاراپ ھىسابلايدۇ.
- () تىرناق ئارقىلىق ھىسابلاش تەرتىپىنى ئۆزگەرتكىلى بولىدۇ.

مەسىلەن:

2 + 3 * 5 = 17 (2 + 3) * 5 = 25

2 * 3 + 5 = 11 2 * (3 + 5) = 16

تۆۋەندىكى جەدۋەلدە ھىسابلاش تەرتىپى يۇقىرىدىن تۆۋەنگە قاراپ تىزىلغان:

ئىجرا بولۇش تەرتىپى	بەلگە
1	**
2	* / % //
3	+ -
4	> < >= <=
5	== !==
6	%= /= //= -= += *= **=
7	and or not

8 › 9 and 7 == 3

ئۆزگۈرۈشچان قىممەت:

پروگراممىلار سانلىق مەلۇماتلارنى بىر تەرەپ قىلىشقا ئىشلىتىلىدۇ، ئۆزگۈرۈشچان قىممەت سانلىق مەلۇماتنى ساقلاشقا ئىشلىتىلىدۇ.

1. ئۆزگۈرۈشچان قىممەتكە قىممەت بېرىش:

python دا ئۆزگۈرۈشچان قىممەتنى ئىشلىتىشتىن بۇرۇن چوقۇم ئۇنىڭغا قىممەت بېرىش كېرەك. مەسىلەن

id = 498139

2. ئۆزگۈرۈشچان قىممەتنىڭ تىپى:

باشقا پروگراممىلا تىللىرىدا ئۆزگۈرۈشچان قىممەتنى ئىشلىتىشتىن بۇرۇن ئۇنىڭ تىپىنى بەلگىلەيدۇ. ئەمما **python** دا ئۆزگۈرۈشچان قىممەتنىڭ تىپىنى ئالدىن بەلگىلىمەيدۇ، پەقەت = بەلگىسى بىلەن بېرىلگەن قىممەت ئارقىلىق ئۆزگۈرۈشچان قىممەتنىڭ تىپى بەلگىلىنىدۇ.

python دا مەلۇمات تىپى سان ۋە سان بولمىغان دەپ ئىككىگە ئايرىلىدۇ.

- سان تىپى:

پۈتۈن سان**(int)**

پارچه سان(float)

بول(bool)

كۆپلۈك(complex) : ئاساسلىقى ئىلمىي ھېسابلاش ئۈچۈن ئىشلىتىلىدۇ.

- سان بولمىغان تىپ:

تېكىست (string)""

تىزىملىك(list)[]

گۇرۇپ(tuple)()

لوغەت(dictionary){}

3. ئۆزگۈرۈشچان مىقدارنىڭ ئىسىم قويۇش:

ئۆزگۈرۈشچان مىقدارنىڭ ئىسىم قويغاندا تۆۋەندىكى قۇرالارغا رىئايە قىلىش لازىم

- ئۆزگۈرۈشچان مىقدارنىڭ ئىسمى پەقەت ھەرپ، سان ۋە تۆۋەن سىزىقتىن تەركىپ تاپقان بولىشى لازىم، لىكىن سان بىلەن باشلانماسلىقى كېرەك.

مەسىلەن :

message_1 √

1_message x

- ئۆزگۈرۈشچان مىقدارنىڭ ئىسمىدا بوشلۇق بولماسلىقى لازىم. مەسىلەن:

greeting_message √

greeting message x

- ئۆزگۈرۈشچان مىقدارنىڭ ئىسمى python دىكى ھالقىلىق سۆز ياكى فونكسىيە ئىسمى بىلەن ئوخشاش بولماسلىقى لازىم.
- ئۆزگۈرۈشچان مىقدارنىڭ ئىسمى ئىمكان قەدەر قىسقا ھەم چۈشۈنۈشلۈك بولىشى لازىم.
- ئىمكان قەدەر كىچىك l بىلەن چوڭ O نى ئىشلەتمەسلىك، بولمىسا كىشىلەر ئۇنى سان 1 بىلەن سان 0 دەپ قالىدۇ.

ئوخشىمىغان تىپتىكى ئۆزگۈرۈشچان قىممەتلەر ئارىسىدىكى ھىسابلاشلار:

- سان تىپىدىكى ئۆزگۈرۈشچان قىممەتلەر ئارا بىۋاستە ھىسابلاش ئىلىپ بارغىلى بولىدۇ. مەسىلەن:

```
i = 10
f = 1.5
b = True
result = i + f + b
print(result)
```

- تېكىست تىپىدىكى ئۆزگۈرۈشچان قىممەتلەر ئارا + ئارقىلىق تېكىستلەرنى ئۆزئارا ئۇلىغىلى بولىدۇ. مەسىلەن:

```
first-name = “Omer”
last-name = “Faruk”
result = first-name + last-name
print(result)
print(first-name + last-name)
```

- تېكىست تىپىدىكى ئۆزگۈرۈشچان قىممەتلەر ئارا * ئارقىلىق ھەرپلەرنى ئۆزئارا تەكرار ئۇلىغىلى بولىدۇ. مەسىلەن:

```
i = 25
flag = “_“
result = i * flag
print(result) → _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

- سان تىپىدىكى ئۆزگۈرۈشچان قىممەت بىلەن تېكىست تىپىدىكى ئۆزگۈرۈشچان قىممەت ئارىسىدا باشقا ھىسابلاش ئىلىپ بارغىلى بولمايدۇ.

فونكسىيە input نىڭ ئىشلىتىلىشى:

فونكسىيە **input** كونۇپكا تاختىسىدىن كىرگۈزۈلگەن مەلۇماتلارنى قوبۇل قىلىش ئۈچۈن ئىشلىتىلىدۇ.

ئىشلەتكۈچى كىرگۈزگەن بارلىق مەلۇماتلارنى

تېكىست **(string)** شەكلىدە قوبۇل قىلىدۇ.

تىپ ئۆزگەرتىش فونكسىيىسى:

int(x) x نى پۈتۈن سانغا ئايلاندۇرىدۇ.

float(x) x نى پارچە سانغا ئايلاندۇرىدۇ.

ئۆزگۈرۈشچان قىممەتنى فورماتلىق چىقىرىش:

ئەگەر تېكىست ئۇچۇرلىرى ۋە باشقا سانلىق مەلۇماتلارنى بىللە چىقارماقچى بولساق ، فورماتلىغۇچنى ئىشلىتىشىمىز كېرەك. % بۇ بەلگە فورماتلىغۇچى بەلگە دەپ ئاتىلىدۇ ، بۇ تېكىست فورماتنى بىر تەرەپ قىلىشتا ئالاھىدە ئىشلىتىلىدۇ.

%s تېكىست تىپىدىكى ئۆزگۈرۈشچان قىممەت ئۈچۈن ئىشلىتىلىدۇ.

%d پۈتۈن سانلار ئۈچۈن ئىشلىتىلىدۇ، **%06d** چىقىرىلغان پۈتۈن سانىنىڭ خانە سانىنى بىلدۈرىدۇ.

%f پارچە سانلار ئۈچۈن ئىشلىتىلىدۇ، **%.2f** چىقىرىلغان پارچە سانىنىڭ خانە سانىنى بىلدۈرىدۇ.

مەسىلەن:

```
isim = "Ahmed"
yax = 25
boy = 1.75453453
print("ismi: %s, yixi: %d, boyi: %.2fm" % (isim, yax, boy))
```

→ **ismi: Ahmed, yixi: 25, boyi: 1.75m**

مەشىق

تاللا بازىرىدىن مۇۋە ستىش پروگراممىسى:

1. ئالدى بىلەن مال ساتقۇچى ستىلغان مۇننىڭ ئىسمىنى كىرگۈزىدۇ.

2. ئاندىن ستىلغان مۇننىڭ باھاسىنى كىرگۈزىدۇ.

3. ئاندىن ستىلغان مۇننىڭ ئېغىرلىقى كىرگۈزىدۇ.

4. ئاخىردا پروگراممىا مال ستىش تالۇنىنى تۆۋەندىكىدەك بىسىپ چىقىرىدۇ:

****

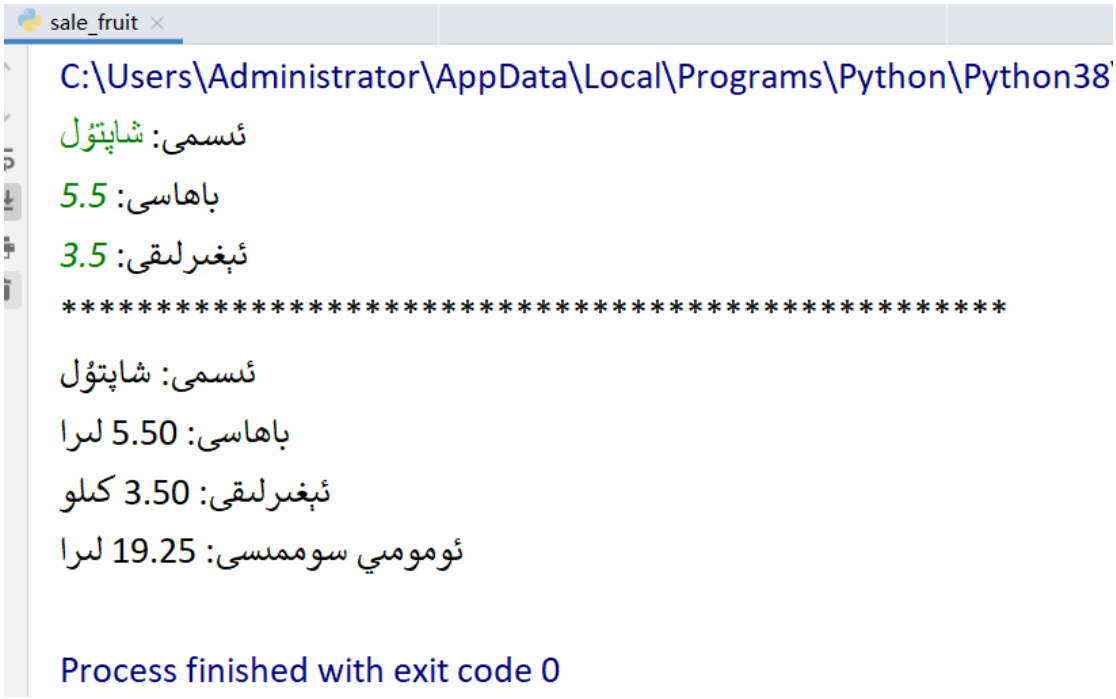
ئىسمى: شاپتۇل

باھاسى: 8.5 لىرا \ كىلو

ئېغىرلىقى: 3.5 كىلو

ئومومىي سوممىسى: 29.75 لىرا

```
name = input("ئىسمى: ")
price = float(input("باھاسى: "))
weight = float(input("ئېغىرلىقى: "))

total = price * weight
print("*" * 50)
print("ئىسمى: %s" % name)
print("باھاسى: %.2f  لىرا" % price)
print("ئېغىرلىقى: %.2f  كىلو" % weight)
print("سوممىسى  ئومومىي: %.2f  لىرا" % total)
```



ھالقىلىق سۆز:

python نىڭ ئىچكى قىسمىدا ئاللىبۇرۇن ئىشلىتىلگەن بەلگە بولۇپ، ئۇنىڭ ئالاھىدە ئىقتىدارى ۋە مەنىسى بار. شۇڭا پروگراممىر بۇ ھالقىلىق سۆزگە ئوخشاش كەلمىلەرنى قوللىنىشى چەكلىنىدۇ.

تۆۋەندىكى كود ئارقىلىق **python** دىكى بارلىق ھالقىلىق سۆزلەرنى كۆرگىلى بولىدۇ.

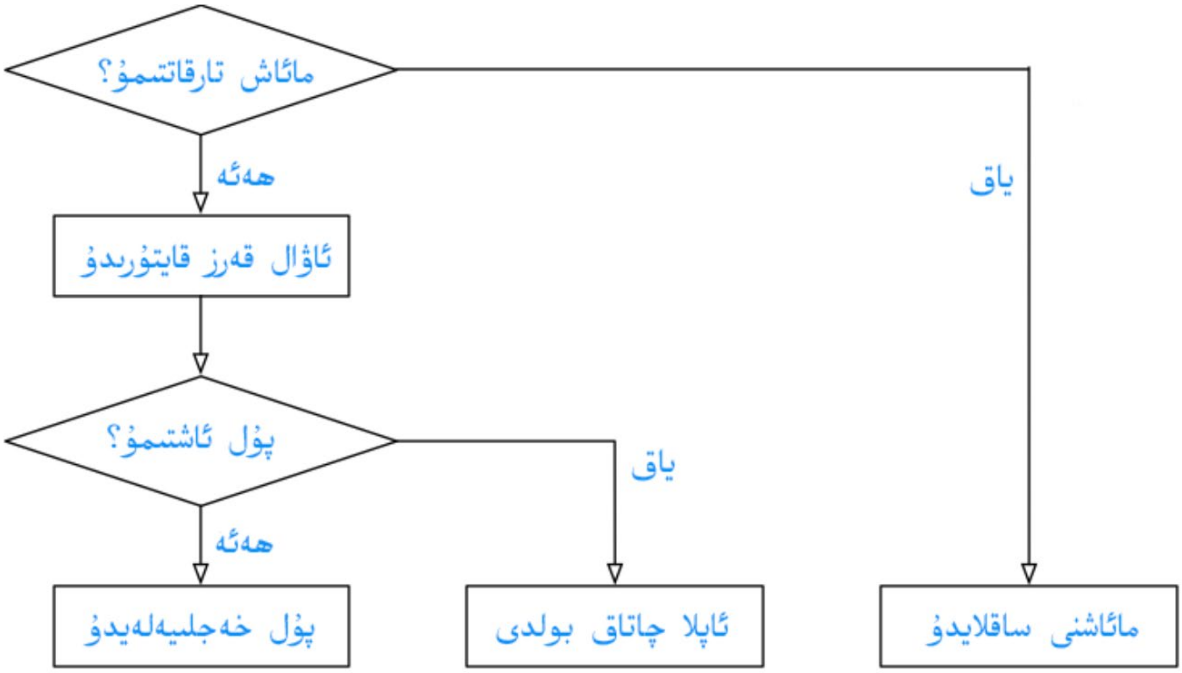
```
import keyword
print(keyword.kwlist)
```

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

if جۈملەسى:

ئەگەر شەرت ھازىرلانسا، مەلۇم بىر ئىشنى قىلالايدۇ،

ئەگەر شەرت ھازىرلانمىسا، باشقا بىر ئىشنى قىلىدۇ، ياكى ھېچ ئىش قىلمايدۇ.



if نىك تىپلىرى:

تىپى	ئىشلىتىلىشى
if...	if A: print(A)

if... else...	if A:
	print(A)
	else:
	print(B)

if... elif... elif...	if A:
	print(A)
	elif B:
	print(B)

if... elif... elif... else...	if A: print(A)

elif B:
    print(B)
else:
    print(C)	

if جۈملىسىنىڭ ئىچىدە يەنە if جۈملىسىنى ئىشلىتىش:

```
has_ticket = True
knife_length = 21
if has_ticket:
    print("قىلىڭ تەكشۈرۈش بىخەتەرلىك بارىكەن، بىلىتىڭىز")

    if knife_length > 20:
        print("سانتىمېتىركەن %s پىچىقىڭىز" % knife_length)
        print("بولمايدۇ چىقىشقا ئېلىپ پويىزغا")

    else:
        print("تاماملاندى تەكشۈرۈش بىخەتەرلىك ")
        print("بولسۇن كۆڭۈللۈك سەپەرلىرىڭىز !")
else:
    print("ئېلىڭ بىلەت ئاۋال")
```

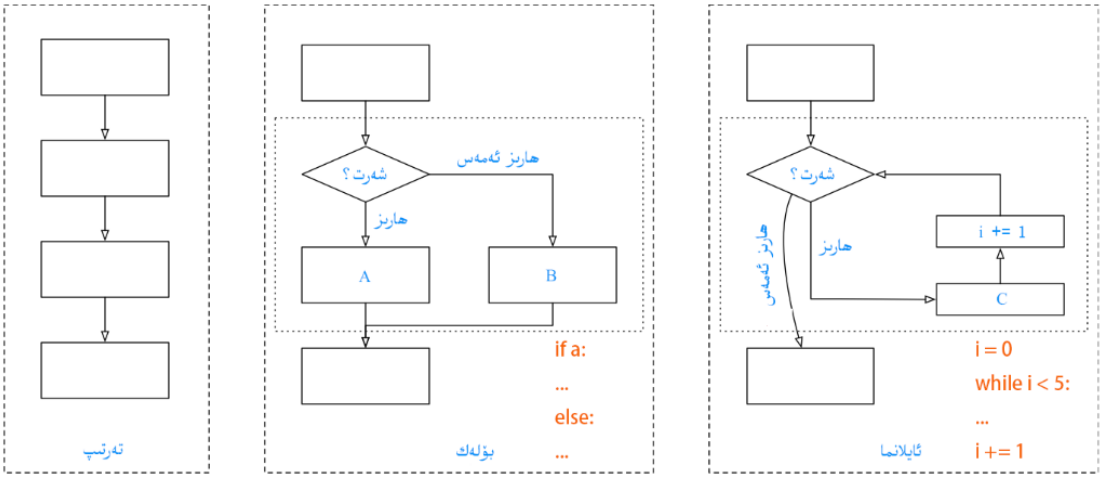
لوگىكىلىق ئىپادىلەرنىڭ if جۈملىسىدە ئىشلىتىلىشى:

```
if age >= 0 and age <= 120:
if python >= 60 or C++ >= 60:
if not is_employee:
```

ئىختىيارى سان(random)نىڭ ئىشلىتىلىشى:

```
import random
print(random.randint(1, 9))          √
print(random.randint(9, 1))          x
```

پروگراممىنىڭ 3 خىل ئىجرا بولۇش شەكلى:



دەۋرىيلىك

دەۋرىيلىك بولسا مەلۇم بىر بۆلەك كودنى قايتا-قايتا ئىجرا قىلىش.

while نىڭ رولى بولسا ئىجرا قىلىنىدىغان كودنى بەلگىلەنگەن قېتىمغا ئاساسەن قايتا-قايتا ئىجرا قىلىش.

مەسلەن:

```
i =  0
while i < 5:
    print(“hello python”)
    i += 1
```

ئەگەر **i += 1** دېگەن بۇ كودنى يازمىساق، بۇ دەۋرىيلىك چەكسىز دەۋرىيلىككە ئايلىنىپ قالىدۇ دە پروگراممىدا خاتالىق كۆرۈلىدۇ. شۇڭا دەۋرىيلىك كودىنى ئىشلەتكەندە چوقۇم ئۇنىڭ ئاخىرلىشىش كودىنى قوشۇپ يېزىشنى ئۇنتۇپ قالماسلىقىمىز لازىم.

break ۋە continue

مەخسۇس دەۋرىيلىك ئىچىدە ئىشلىتىلىدىغان ھالقىلىق سۆز.

break مەلۇم بىر شەرت ھازىر بولغاندا، دەۋرىيلىك توختايدۇ ۋە ئارقىدىكى كودلار ئىجرا بولمايدۇ.

continue مەلۇم بىر شەرت ھازىر بولغاندا، مەزكۇر دەۋرىيلىك توختايدۇ ئارقىدىكى كودلار ئىجرا بولمايدۇ ۋە كېيىنكى دەۋرىيلىككە ئۆتىدۇ.

continue نى ئىشلىتىشتىن بۇرۇن دەۋرىيلىكنىڭ ئاخىرلىشىش كودىنى يېزىشىمىز كېرەك، بولمىسا چەكسىز دەۋرىيلىككە ئايلىنىپ قېلىشى مۇمكىن.

مىسال: **beak**

```
i = 0
while i < 10:
    if i == 3:
        break
    print(i)
    i += 1
print("over")
```

مىسال: **continue**

```
    i = 0
while i < 10:
    if i == 3:
        i += 1
        continue
```

```
    print(i)
    i += 1
```

دەۋرىيلىك while نىڭ ئىچىدە يەنە دەۋرىيلىك while نى ئىشلىتىش:

```
    row = 1
while row <= 5:
    col = 1

    while col <= row:
        print("*", end="")
        col += 1

    print("")
    row += 1
```

تېكىستتىكى ئۆزگەرتىش بەلگىلىرى:

\\ يانتۇ سىزىق

\' تاق تىرناق

\" جۈپ تىرناق

\n قۇر تاشلاش

\t توغرىسىغا 4 بوشلۇق يۆتكۈگۈچ

\r يېڭى قۇر

مىسال : cy-07.py

مەشىق

1. 0دىن 100 غىچە بارلىق سانلارنى ئۆزئارا قوشۇپ نەتىجىسىنى تاپىدىغان بىر پروگراما يېزىڭ.

```
result = 0
# 1. سانىنى ئايلانما ئۇنىڭغا يېزىپ بىر قېتىمدىن ئۆزگۈرۈشچان خاتىرىلەيمىز
i = 0
# 2. باشلىنىدۇ باشلاپ يەردىن بۇ ئايلانما
while i <= 100:
    print(i)

    # ئۆزئارا نى i بىلەن result ئايلانغاندا قېتىم ھەر قوشىمىز
    result += i
    # قېلىش بىرتەرەپ كۆدىنى ئاخىرلىشىش
    i += 1
print("بولىدۇ %s يىغىندىسى سانلارنىڭ بۇ" % result)
```

2. 0دىن 100 غىچە بارلىق جۈپ سانلارنى ئۆزئارا قوشۇپ نەتىجىسىنى تاپىدىغان بىر پروگراما يېزىڭ.

```
# 1. سانىنى ئايلانما ئۇنىڭغا يېزىپ بىر قېتىمدا تېخىن ئۆزگۈرۈشچان خاتىرىلەيمىز
result = 0
i = 0
while i <= 100:
    # قېتىم i نىڭ ھۆكۈم ئەمەس ياكى سانمۇ جۈپ نىڭ قېتىمىنى خاتىرىلەيمىز
    if i % 2 == 0:
        print(i)
        # قوشىمىز بولسا سان جۈپ ئەگەر
        result += i

    i += 1
print("بولىدۇ %s يىغىندىسى سانلارنىڭ بۇ" % result)
```

فونكسىيە:

مۇستەقىل ئىقتىدارلىرى بار بولغان كود بۆلەكلىرى فونكسىيە دېيىلىدۇ.

فونكسىيە رولى:

كود يېزىشنىڭ ئۇنۇمنى يۇقىرى كۆتۈرۈش، ۋە كودنى تەكرار ئىشلىتىش.

فونكسىيە يېزىش قائىدىسى:

- ئەڭ ئاۋال **def** نى يازىمىز، ئاندىن فونكسىيە ئىسمىنى يازىمىز،
- فونكسىيەگە ئىسىم يازغاندا، ئۇ فونكسىيەنىڭ ئىقتىدارىنى ئىپادىلەپ بېرەلەيدىغان، ئىخچام بولغان سۆزلەرنى ئىشلىتىش لازىم.
- فونكسىيە ئىسمى ھەرپ، تۆۋەن سىزىق، ۋە سان بىلەن يېزىلسا بولىدۇ، لىكىن سان بىلەن باشلانماسلىقى ۋە ھالقىلىق سۆزلەر بىلەن ئوخشاش بولۇپ قالماسلىقى لازىم.

```
    def test():
        print("hello world")
        print("hello python")
        print("hello Turkistan")
print("hello java")
test()
print("hello hello")
```

فونكسىيەنىڭ ئىشلىتىلىشى:

فونكسىيە ئىسمى يېزىپ ئىشلەتكەندىلا ئاندىن فونكسىيەدىكى كودلار ئىجرا قىلىنىدۇ، ئىجرا قىلىنىپ بولغاندىن كېيىن، فونكسىيەنى ئىشلەتكەن ئورۇنغا قايتىپ كېلىپ ئۇنىڭدىن كېينكى كودلارنى ئىجرا قىلىدۇ.

سۇئال:

فونكسىيەنى ئاۋال ئىشلىتىپ ئاندىن فونكسىيەنى يېزىشقا بولامدۇ؟

جاۋاب: بولمايدۇ، چۈنكى فونكسىيەنى ئىشلىتىشتىن بۇرۇن **python** غا بۇ فونكسىيەنىڭ مەۋجۇد ئىكەنلىكىنى بۈلدۈرۈش لازىم.

pycharm نىڭ كود تەڭشەش قۇرالى:

F8 Step Over يەككە قەدەملىك ئىجرا قىلىدۇ، **بۇ** فونكسىيەنى بىر قۇر كود دەپ قاراپ بىۋاستە ئىجرا قىلىدۇ.

F7 Step Into يەككە قەدەملىك ئىجرا قىلىدۇ، ئەگەر فونكسىيە بولۇپ قالسا، فونكسىيەنىڭ ئىچىگە كىرىپ ئىجرا قىلىدۇ.

فونكسىيەنىڭ ئىزاھاتى:

ئەگەر فونكسىيەگە ئىزاھات بەرمەكچى بولساق ، فونكسىيە ئېنىقلىمىسىنىڭ ئاستىغا ئۈدا ئۈچ جۈپ تىرناق ئاچىمىز.

```
def multiple_table():
    """جەدۋىلى كۆپەيتىش"""
```

فونكسىيەنىڭ پارامېتىر:

فونكسىيەنىڭ ئىنىقلىمىسىنى يازغاندا تىرناق ئىچىگە ھەرخىل ئۆزگۈرۈشچان قىممەتنى يازغىلى بولىدۇ، ۋە بۇ ئۆزگۈرۈشچان قىممەتنى فونكسىيەنىڭ ھەرقايسى ئورۇنلىرىدا ئىشلەتكىلى بولىدۇ.

رەسمىي پارامېتىر:

فونكسىيەنىڭ ئىنىقلىمىسىنى يازغاندا تىرناق ئىچىگە يازغان ئۆزگۈرۈشچان قىممەت رەسمىي پارامېتىر دېيىلىدۇ.

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

بۇ يەردىكى a, b رەسمىي پارامېتىر

ئەملىي پارامېتىر:

فونكسىيەنى ئىشلەتكەندە تىرناق ئىچىگە يېزىلغان مەلۇماتلار ئەملىي پارامېتىر دېيىلىدۇ. بۇ مەلۇماتلار رەسمىي پارامېتىردىكى ئۆزگۈرۈشچان قىممەتكە يەتكۈزىلىدۇ.

```
sum(10, 15)
```

فونكسىيەنىڭ قايتۇرۇش قىممىتى:

فونكسىيە خىزمىتىنى تاماملىغاندىن كېيىن ئىشلەتكۈچىگە قايتۇرغان ئاخىرقى نەتىجىسى فونكسىيەنىڭ قايتۇرۇش قىممىتى دېيىلىدۇ. ھالقىلىق سۆز return ئارقىلىق فونكسىيەنى ئىشلەتكەن ئورۇنغا ئاخىرقى نەتىجىنى قايتۇرىدۇ.

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

python دا فونكسىيەنلەر پۈتۈن سان، پارچە سان، تىكست، گورۇپ، تىزىملىك، لۇغەت، set، bool تىپىدىكى مەلۇماتنى قايتۇرالايدۇ.

فونكسىيەنلەر ئىچىدە فونكسىيە ئىشلىتىش:

```
def test1():
  print("*" * 50)

def test2():
   print("-" * 50)
      test1()
      print("+" * 50)

test2()
```

مودۇلدىكى فونكسىيەنى ئىشلىتىش:

كېڭەيتىلمە ئىسمى py بىلەن ئاخىرلاشقان ھەر بىر Python ھۆججىتى بىر مودۇل بولالايدۇ. مودۇلدا ئېنىقلانغان ئۆزگۈرۈشچان قىممەت ۋە فونكسىيەنلەر ھەممىسىنى مودۇلنىڭ سىرتىدا بېۋاستە ئىشلەتكىلى بولىدۇ.

ئۇنى import ھالقىلىق سۆزى ئارقىلىق ئىشلەتمەكچى بولغان ھۆججەتكە ئەكىرىپ ئىشلىتىشكە بولىدۇ.

مودۇل ئىسمى:

- فونكسىيە ئىسمى ھەرپ، تۆۋەن سىزىق، ۋە سان بىلەن يېزىلسا بولىدۇ،
- سان بىلەن باشلانماسلىقى لازىم
- ھالقىلىق سۆزلەر بىلەن ئوخشاش بولۇپ قالماسلىقى لازىم.

ئىلغار ئۆزگىرىشچان قىممەت:

ئۆزگۈرۈشچان قىممەتنىڭ تىپى:

- سان تىپى:

پۈتۈن سان(int)

پارچە سان(float)

بول(bool)

كۆپلۈك(complex) : ئاساسلىقى ئىلمىي ھېسابلاش ئۈچۈن ئىشلىتىلىدۇ.

- سان بولمىغان تىپ:

تېكست (string)

تىزىملىك(list)

گۇرۇپ(tuple)

لوغەت(dictionary)

تىزىملىك (list):

- ➢ بىر قاتار سانلارنى ساقلاش ئۈچۈن ئىشلىتىلىدۇ.
- ➢ [ ] بىلەن ئىنقىلما بېرىلىدۇ. سانلارنىڭ ئارىسى ، بىلەن ئايرىلىدۇ.
- ➢ تىزىملىكنىڭ تەرتىپى 0 دىن باشلىنىدۇ.

ئەسكەرتىش: تىزىملىكتىن قىممەت ئالغاندا ، كۆرسەتكۈچ تىزىملىكنىڭ دائىرىسىدىن ئېشىپ كەتسە ، پروگراممىدا خاتالىق كۆرۈلىدۇ.

```
name_list = ['Ahmed', 'Osman', 'Yuseyin']
print(name_list[0])    →    ?
```

تىزىملىك(list) كۆپ ئىشلىتىدىغان يۆنتەملەر:

name_list.append	name_list.count	name_list.insert
name_list.reverse	name_list.clear	name_list.extend
name_list.pop	name_list.sort	name_list.copy

name-list.remove name-list.index

تۈرى	ئۇسۇللار	چۈشەندۈرىشى
قوشۇش	list.insert(3, ‘hello’)	3 . ئورۇنغا hi نى قوشۇش
	list.append(5)	ئەڭ ئاخىرىغا 5 نى قوشۇش
	list.extend(list2)	list2 دىكى مەلۇماتنى list غا قوشۇش
ئۆزگەرتىش	list[1]=7	1 . ئورۇندىكى مەلۇماتنى 7 گە ئۆزگەرتىش
ئۆچۈرۈش	del list[2]	2 . ئورۇندىكى مەلۇماتنى ئۆچۈرۈش
	list.remove(7)	1 . قىممىتى ئۇچرىغان 7 نى ئۆچۈرۈش
	list.pop	ئەڭ ئاخىرىدىكنى ئۆچۈرۈش
	list.pop(2)	2 . ئورۇندىكى مەلۇماتنى ئۆچۈرۈش
	list.clear	list نى تازىلاش
ئىستاتىستىكا	len(list)	list نىڭ ئۇزۇنلۇقى
	list.count(5)	list دىكى 5نىڭ قىممىتى سانى
رەتلەش	list.sort()	تۆۋەندىن يۇقىرىغا رەتلەش
	list.sort(reverse=True)	يۇقىرىدىن تۆۋەنگە رەتلەش
	list.reverse()	تەتۈر رەتلەش

دىكى list نىڭ 10 تەرتىپىنى تاپىدۇ	list.index(10)	

ئەسكەرتىش:

1. del ھالقىلىق سۆز ماھىيەتتە ئۆزگۈرۈشچان قىممەتنى ئېچكى ساقلىغۇچ (RAM) دىن ئۆچۈرۈش ئۇچۇن ئىشلىتىلىدۇ. ئەگەر بۇنىڭ ئۆچۈرۈلسە، كېيىنكى كوددا بۇ ئۆزگۈرۈشچان قىممەتنى ئىشلەتكىلى بولمايدۇ.
2. فونكسىيە مۇستەقىل ئىقتىدارلارنى ئۆز ئىچىگە ئالغان ، بىۋاستە ئىشلىتىشكە بولىدۇ.
3. يۆنتەمۇ مۇستەقىل ئىقتىدارلارنى ئۆز ئىچىگە ئالغان بولۇم، ئوبيېكت ئارقىلىق ئىشلىتىشكە بولىدۇ.

تىزىملىكتىكى بارلىق مەلۇماتقا ئېرىشىش:

ھالقىلىق سۆز for ئارقىلىق تىزىملىكتىكى بارلىق مەلۇماتقا بىرىدىن-بىرىدىن ئېرىشكىلى بولىدۇ.

```
for name in name_list:
    print(name)
```

گورۇپ (tuple):

- بىر قاتار سانلارنى ساقلاش ئۇچۇن ئىشلىتىلىدۇ.
- ( ) بىلەن ئىنقىلىما بېرىلىدۇ. سانلارنىڭ ئارىسى ، بىلەن ئايرىلىدۇ.
- گورۇپنىڭ تەرتىپى 0 دىن باشلىنىدۇ.

ئەسكەرتىش:

تىزىملىك بىلەن گورۇپ ئوخشاپ قالىدۇ، ئوخشىمايدىغان يېرى:

گورۇپنىڭ ئېچىدىكى مەلۇماتنى ئۆزگەرتكىلى بولمايدۇ.

- گورۇپ قۇرۇش:

```
info_tuple = ("Omer", 18, 1.75, "Omer")
```

- قۇرۇق گورۇپ قۇرۇش:

```
info_tuple = ()
```

- بىر ئېلېمېنتلىق گورۇپ قۇرۇش:

```
info_tuple = (7,)
```

گورۇپنىڭ كۆپ ئىشلىتىدىغان مەشغۇلاتلىرى:

گورۇپنىڭ ئۇزۇنلىقى:

```
len(info_tuple)
```

سانلىق مەلۇماتنىڭ گورۇپدا كۆرۈلۈش قېتىم سانى:

```
info_tuple.count("Omer")
```

قىممەتكە ئېرىشىش:

```
info_tuple[0]
```

كۆرسەتكۈچكە ئېرىشىش:

```
info_tuple.index("Omer")
```

گورۇپتىكى بارلىق مەلۇماتقا ئېرىشىش:

هالقىلىق سۆز **for** ئارقىلىق گورۇپتىكى بارلىق مەلۇماتقا ئېرىشكىلى بولىدۇ.

```
for my_info in info_tuple:
    print(my_info)
```

گورۇپنىڭ ئىشلىتىش ئورۇنلىرى:

- فونكسىيەنىڭ پارامېتىرى ۋە قايتۇرۇش قىممىتى ئورنىدا گورۇپ تىپىنى ئىشلىتىشكە بولىدۇ

- فورمات تىزمىسى

```
info_tuple = ("Omer ", 21, 1.85)
print("%s يېشى %d بويى %.2f" % (info_tuple[0], info_tuple[1], info_tuple[2]))
(info_tuple[0], info_tuple[1], info_tuple[2]) = info_tuple
```

- قىممىتىنى ئۆزگەرتىشنى خالىمىغان ئورۇنلاردا

تىزىملىك ۋە گورۇپنى ئۆزئارا ئالماشتۇرۇش:

list فۇنكسىيەسىنى ئىشلىتىپ گورۇپنى تىزىملىككە ئايلاندۇرغىلى بولىدۇ.

```
info_tuple = ("Omer", 21, 1.85)
new_list = list(info_tuple)
print(new_list)
```

tuple فۇنكسىيەسىنى ئىشلىتىپ تىزىملىكنى گورۇپقا ئايلاندۇرغىلى بولىدۇ.

```
info_list = ["Omer", 21, 1.85]
new_tuple = tuple(info_list)
print(new_tuple)
```

سوئال: $a = 7, b = 5$ بۇ ئىككى ئۆزگىرىشچان قىممەتنىڭ ئۆز ئارا قىممىتىنى ئالماشتۇرۇڭ.

لۇغەت ئېنىقلىمىسى:

لۇغەت بولسا ئاچقۇچلۇق قىممەت توپلىمى. لۇغەتمۇ كۆپ خىل مەمۇماتنى ساقلىيالايدۇ. ئادەتتە بىر جىسىمغا مۇناسىۋەتلىك ئۇچۇرنى ساقلايدۇ.

تىزىملىك بىلەن بولغان پەرقى:

تىزىملىك تەرتىپلىك، لۇغەت بولسا تەرتىپسىز.

لۇغەتكە { } بۇ تىرناق بىلەن ئىنقىلما بېرىلىدۇ.

```
person = {"name": "Omer", "age": 18, "height": 1.75}
```

ئاچقۇچ: name, age, height

قىممەت: Omer, 18, 1.75

ئەسكەرتىش:

- ئاچقۇچ چوقۇم بىرلا بولىشى كېرەك.
- ئاچقۇچ ۋە قىممەت : بىلەن ئايرىلىپ يېزىلىدۇ.
- قىممەتكە ھەرقانداق تىپنى يېزىشقا بولىدۇ، ئاچقۇچقا تىكست، سان، گورۇپ يېزىشقا بولىدۇ.

لۇغەتنىڭ كۆپ ئىشلىتىدىغان مەشغۇلاتلىرى:

يۆنتەملەر	چۈشەندۈرىشى
person["name"]	ئاچقۇچ ئارقىلىق قىممەتكە ئېرىشىش
person["age"] = 21	ئاچقۇچ ئارقىلىق قىممەت ئۆزگەرتىش
person["gender"] = "male"	يېڭى ئاچقۇچ ۋە قىممەت قىتىش
len(person)	ئاچقۇچلۇق قىممەت سانى
person.clear()	بارلىق ئاچقۇچلۇق قىممەتنى ئۆچۈرۈش
person.items()	ئاچقۇچلۇق قىممەت تىزىملىكى
person.setdefault("ab", 67)	ئاچقۇچ بولسا، ھېچ ئىش قىلمايدۇ، ئاچقۇچ بولمسا، يېڭى ئاچقۇچلۇق قىممەت قاتىدۇ
new-person=person.copy()	personنى new-personغا كۆچۈرۈش
person.keys()	لۇغەتتىكى ئاچقۇچلار توپلىمى
person.values()	لۇغەتتىكى قىممەتلەر توپلىمى
person.update(temp-dict)	person نى يېڭىلاش
new=dict.fromkeys(person)	ئاچقۇچى personغا ئوخشايدىغان قىممىتى

person.pop(″age″)	لۇغەتتىكى ئاچقۇچلۇق قىممەت ″age″:18 نى ئۆچۈرۈش
person.get(″name″)	ئاچقۇچ ئارقىلىق قىممەتكە ئېرىشىش
person.popitems	ئەڭ ئاخىردىكى ئاچقۇچلۇق قىممەتنى ئۆچۈرۈش

لۇغەتتىكى بارلىق مەلۇماتقا ئېرىشىش:

ھالقىلىق سۆز **for** ئارقىلىق لۇغەتتىكى بارلىق مەلۇماتقا ئېرىشكىلى بولىدۇ.

```
for k in person:
    print("%s - %s" % (k, person[k]))
```

تېكىست(string):

تېكىست بولسا ھەرپلەر توپلىمى، **python** دا تېكىستنى″″ قوش تىرناق ياكى `` تاق تىرناق بىلەن ئىنقىلىما بېرىمز. مەسىلەن

```
a = "helle python"
b = `helle python`
```

ئەسكەرتىش:

- ئەگەر تېكىست ئىچىدە ″″ قوش تىرناق بولۇپ قالسا تېكىستنى `` تاق تىرناق بىلەن ئىنقىلىما بېرىمز.
- ئەگەر تېكىست ئىچىدە `` تاق تىرناق بولۇپ قالسا تېكىستنى ″″ قوش تىرناق بىلەن ئىنقىلىما بېرىمز.
- ھالقىلىق سۆز **for** ئارقىلىق تېكىستتىكى ھەربىر ھەرپكە بىرىدىن ـ بىرىدىن ئېرىشكىلى بولىدۇ.
- ئاچقۇچ ئارقىلىق تېكىستتىكى مەلۇم بىر ئورۇندىكى ھەرپكە بىۋاستە ئېرىشكىلى بولىدۇ.

تېكىستنىڭ كۆپ ئىشلىتىدىغان مەشغۇلاتلىرى:

1. هۆكۈم تىپى:

ئۇسۇللار	چۈشەندۈرىشى
string.isspace()	Stringنىڭ ھەممىسى بوشلۇق بولسا، Trueنى قايتۇرىدۇ
string.isalnum()	string نىڭ ھەممىسى ھەرپ ياكى سان بولسا True نى قايتۇرىدۇ
string.isalpha()	string نىڭ ھەممىسى ھەرپ بولسا True نى قايتۇرىدۇ.
string.isdecimal()	string نىڭ ھەممىسى پۈتۈن سان بولسا True نى قايتۇرىدۇ پارچە سان بولسا False نى قايتۇرىدۇ.
string.isdigit()	string نىڭ ھەممىسى پۈتۈن سان بولسا True نى قايتۇرىدۇ پارچە سان بولسا False نى قايتۇرىدۇ. (1) \u00b2غا ئوخشاش سانلارغا ھۆكۈم قىلالايدۇ.
string.isnumeric()	string نىڭ ھەممىسى پۈتۈن سان بولسا True نى قايتۇرىدۇ پارچە سان بولسا False نى قايتۇرىدۇ. ئەرەپچە سان ياكى خىتتايچە سانغا ھۆكۈم قىلالايدۇ.
string.istitle()	stringدىكى ھەممە سۆزنىڭ باش ھەرپى چوڭ بولسا True نى قايتۇرىدۇ
string.islower()	string دىكى ھەممە سۆزنىڭ ھەرپى كىچىك بولسا True نى قايتۇرىدۇ
string.isupper()	string دىكى ھەممە سۆزنىڭ ھەرپى چوڭ بولسا True نى قايتۇرىدۇ

2. ئىزدەش ۋە ئالماشتۇرۇش:

ئۇسۇللار	چۈشەندۈرىشى
string.startswith(str)	string str بىلەن باشلانغان بولسا،

	True نى قايتۇرىدۇ.
string.endswith(str)	string نىڭ ئاياغلاشقان بولسا str بىلەن بولسا True نى قايتۇرىدۇ.
string.find(str, start=0, end=len(string))	string دىكى str نىڭ باشلىنىش ئورنىنى قايتۇرىدۇ. ئەگەر str بولمىسا 1- نى قايتۇرىدۇ. ئىختىيارى پارامېتىر بەلگىلەنگەن دائىرىدە بولسا، تەرتىپ نومۇرىنى قايتۇرىدۇ.
string.rfind(str, start=0, end=len(string))	find() غا ئوخشاش، پەقەت ئوڭدىن باشلاپ ئىزدەيدۇ.
string.index(str, start=0, end=len(string))	find() غا ئوخشاش، ئەگەر str نى تاپالمىسا خاتالىق مەلۇم قىلىدۇ.
string.rindex(str, start=0, end=len(string))	index() غا ئوخشاش، پەقەت ئوڭدىن باشلاپ ئىزدەيدۇ.
string.replace(old-str, new-str,num=string.count(old))	string دىكى old-str نى new-str غا ئالماشتۇرىدۇ. num بولسا ئالماشتۇرۇش قىتىم سانى.

3. چوڭ كىچىك يېزىشنى ئۆزگەرتىش:

ئۇسۇللار	چۈشەندۈرىشى
string.capitalize()	string دىكى بىرىنجى ھەرپنى چوڭ يېزىش
string.title()	string دىكى ھەربىر سۆزنىڭ باش ھەرپنى چوڭ يېزىش
string.lower()	string دىكى بارلىق چوڭ ھەرپنى كىچىك ھەرپ قىلىش
string.upper()	string دىكى بارلىق ھەرپنى چوڭ ھەرپ قىلىش
string.swapcase()	string دىكى چوڭ ھەرپنى كىچىك، كىچىك ھەرپنى چوڭ

	قىلىش

4. تېكست توغرىلاش:

ئۇسۇللار	چۈشەندۈرىشى
string.ljust(width)	string نى سول تەرەپكە توغۇرلاپ ئۇزۇنلۇقى width بولغان يېڭى تېكست چىقىرىدۇ، بوش قالغان قىسمىنى بوشلۇق بىلەن تولۇقلايدۇ.
string.rjust(width)	string نى ئوڭ تەرەپكە توغۇرلاپ ئۇزۇنلۇقى width بولغان يېڭى تېكست چىقىرىدۇ، بوش قالغان قىسمىنى بوشلۇق بىلەن تولۇقلايدۇ.
string.center(width)	string نى مەركەزگە توغۇرلاپ ئۇزۇنلۇقى width بولغان يېڭى تېكست چىقىرىدۇ، بوش قالغان قىسمىنى بوشلۇق بىلەن تولۇقلايدۇ.

5. بوشلۇقنى ئۆچۈرۈش :

ئۇسۇللار	چۈشەندۈرىشى
string.lstrip()	string نىڭ سول تەرىپىدىكى بوشلۇقنى ئۆچۈرىدۇ.
string.rstrip()	string نىڭ ئوڭ تەرىپىدىكى بوشلۇقنى ئۆچۈرىدۇ.
string.strip()	string نىڭ ئوڭ سول تەرىپىدىكى بوشلۇقنى ئۆچۈرىدۇ.

6. پارچىلاش ۋە ئۇلاش:

ئۇسۇللار	چۈشەندۈرىشى
string.partition(str)	string نى 3 ئىلمىنتلىق گورۇپقا پارچىلايدۇ، يەنى (str-front, str, str-behind)
string.rpartition(str)	partition غا ئوخشاش، پەقەت ئوڭ تەرەپتىن

	باشلايدۇ
string.split(str="", num)	بەلگىلەنگەن قائىدە بويىچە string نى پارچىلايدۇ، num بولسا پاچىلاش قىتىم سانى، ئاخىردا تىزىملىك شەكلىدۇ قايتۇرىدۇ.
string.splitlines()	('\r', '\n', '\r\n') غا ئاساسەن ھەربىر قۇرنى بىر ئىلمىنت قىلغان بىر تىزىملىك قايتۇرىدۇ.
string.join(seq)	بۇ ئۇسۇل ئادەتتە(تېكىست، تىزىملىك، گورۇپ، لۇغەت) تېكى تېكىست تىپىدىكى ئىلمىنتلارنى بەلگىلەنگەن string بويىچە ئۇلاپ يېڭى بىر تېكىست ھاسىل قىلىدۇ.

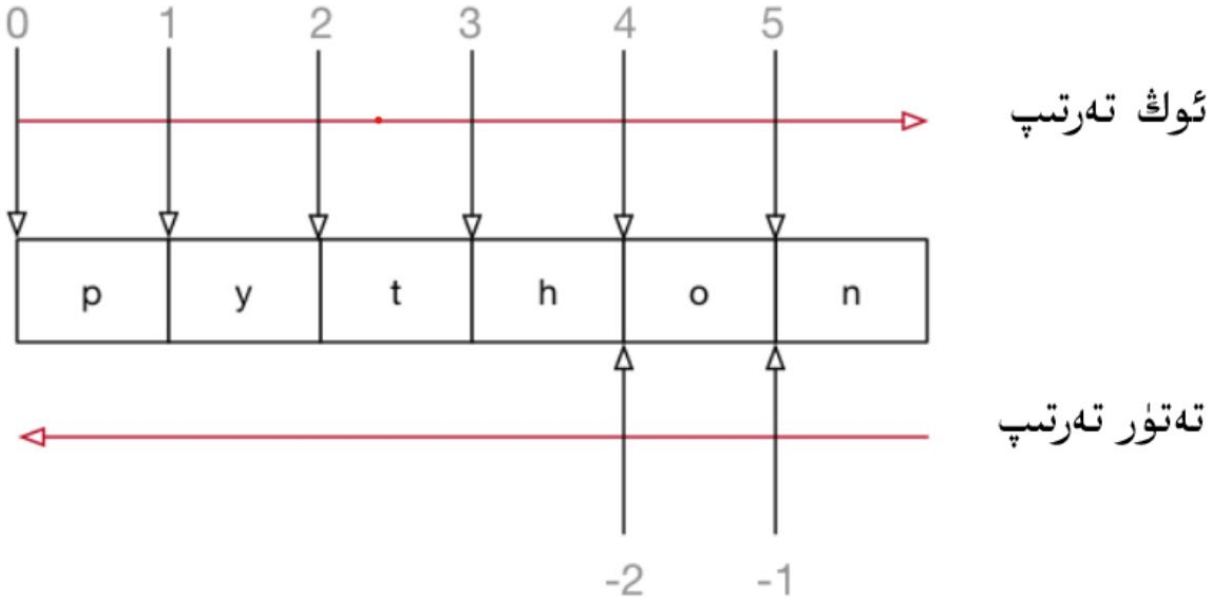
7. باشقا مەشغۇلاتلار

ئۇسۇللار	چۈشەندۈرىشى
len(string)	string ئۇزۇنلۇقى
string.count(str)	str نىڭ string دىكى قىتىم سانى
string.index(str)	str نىڭ string دىكى بىرىنجى قىتىمدىكى ئورنى
string[num]	string نىڭ num ئورۇندىكى ھەرپ

تېكىست(string) نى كىسىش:

چوڭ تېكىستنىڭ ئاچقۇچ قىممىتى ئارقىلىق دائىرە بەلگىلەپ چوڭ تېكىستىن كىچىك بىر تېكىست كىسىشكە بولىدۇ. مەسىلەن

```
string[start:end:size]
```




```
num_str = "0123456789"
print(num_str[2:6])
print(num_str[2:])
print(num_str[2:10])
print(num_str[0:6])
print(num_str[:6])
print(num_str[:])
print(num_str[::2])
print(num_str[1::2])
print(num_str[-1])
print(num_str[2:-1])
print(num_str[-2:])
print(num_str[0::-1])

print(num_str[-1::-1])
print(num_str[::-1])
```

ئەسكەرتىش :

- بەلگىلەنگەن دائىرەنىڭ باشلىنىش ئاچقۇچى تاقاق ئاخىرلىشىش ئاچقۇچى ئوچۇق، يەنى باشلىنىش ئاچقۇچىدىن باشلاپ ئاخىرلىشىش ئاچقۇچىنىڭ ئالدىنقى بىر ھەرپكىچە كىسىدۇ.
- ئەگ باشتىن باشلاپ كەسكەندە، باشلىنىش ئاچقۇچ قىممىتىنى يازمىساقمۇ بولىدۇ، ئەمما : قوش چىكىتنى چوقۇم يازىمىز.
- ئەگ ئاخىرغىچە كەسكەندە ، ئاخىرلىشىش ئاچقۇچىنىڭ قىممىتىنى يازمىساقمۇ بولىدۇ، ئەمما : قوش چىكىتنى چوقۇم يازىمىز.
- ئادەتتە قەدەم سانى 1، ئارقىمۇ ئارقا كەسكەندە قەدەم سانىنى يازمىساقمۇ بولىدۇ.

ئاممىۋى ئۇسۇل:

تىكىست، تىزىملىك، گورۇپ، لۇغەت ھەممىسى ئورتاق قوللىنالايدىغان ئۇسۇل ئاممىۋى ئۇسۇل دىيىلىدۇ.

ئۇسۇل	چۈشەندۈرىشى	ئەسكەرتىش
len(item)	ئىلمىنتنىڭ سانىنى ھىساپلايدۇ	
del(item) **del item**	ئۆزگۈرۈشچان قىممەتنى ئۆچۈرىدۇ	
max(item)	ئەڭ چوڭ قىممەتنى تاپىدۇ	ئەگەر لۇغەت بولسا، پەقەت ئاچقۇچقىلا قارىتىلغان
min(item)	ئەڭ كىچىك قىممەتنى تاپىدۇ	ئەگەر لۇغەت بولسا، پەقەت ئاچقۇچقىلا قارىتىلغان

ھىساپلاش ئىپادىلىرى:

ھىسابلاش بەلگىسى	ئىپادە	نەتىجە	چۈشەندۈرۈش	قوللايدىغىنى
+	[1,2] + [3,4]	[1,2,3,4]	بىرلەشتۈرۈش	تىكىست، تىزىملىك، گورۇپ

تىكست ، تىزىملىك ، گورۇپ	تەكرارلاش	[“hi”, “hi”]	[“hi”] *2	*
تىكست ، تىزىملىك ، گورۇپ ، لۇغەت	مەۋجۇتمۇ؟	True	3 in (1, 2, 3)	**in**
تىكست ، تىزىملىك ، گورۇپ ، لۇغەت	مەۋجۇت ئەمەس	True	4 not in (1, 2, 3)	**not in**
تىكست ، تىزىملىك ، گورۇپ	سېلىشتۇرۇش	True	(1,2,3) <(2,2,3)	> >= < <=

پۈتۈن دەۋرىيلىك for else :

دەۋرىيلىك **for** كودى باشتىن ئاخىرغىچە ئۈزۈلۈپ قالماي ئىجرا بولغاندىن كېيىن **else** كودى ئىجرا بولىدۇ.

```
students = [
        {"name": "Omer", "age": 20, "weight": 75, "height": 1.7},
        {"name": "Ahmed", "age": 19, "weight": 65, "height": 1.6},
        {"name": "Osman", "age": 19, "weight": 80, "height": 1.6}
    ]
find_name = "Omer"
for stu_dict in students:
    if stu_dict["name"] == find_name:
        print("بۇ ئىسىم تىزىملىكتە بار ئىكەن")
        break
```

```
else:
    print("بۇ ئىسىم تىزىملىكتە يوق ئىكەن")
```

ئۆزگىرىشچان قىممەتنىڭ ئاچقۇچى:

ئۆزگىرىشچان قىممەت ۋە سانلىق مەلۇماتنىڭ ھەممىسى ئېچكى ساقلىغۇچتا ساقلىنىدۇ. **python** دىكى فونكسىيەنىڭ پارامېتىرى ۋە قايتۇرۇش قىممىتىنىڭ ھەممىسى ئاچقۇچ ئارقىلىق يۆتكىلىدۇ.

- ئۆزگىرىشچان قىممەت ۋە سانلىق مەلۇمات ئېچكى ساقلىغۇچتا ئايرىم-ئايرىم ساقلىنىدۇ.
- سانلىق مەلۇمات ئېچكى ساقلىغۇچنىڭ مەلۇم بىر ئورنىدا ساقلىنىدۇ.
- ئۆزگىرىشچان قىممەتكە سانلىق مەلۇماتنىڭ ئېچكى ساقلىغۇچتىكى ئادرېسى ساقلىنىدۇ.
- ئۆزگىرىشچان قىممەتتىكى خاتىرىلەنگەن سانلىق مەلۇماتنىڭ ئادرېسى ئاچقۇچ دەپ ئاتىلىدۇ.

a = 1

a = 2

b = a

فونكسىيە **id()** ئارقىلىق ئۆزگىرىشچان قىممەتتىكى سانلىق مەلۇماتنىڭ ئادرېسىنى تەكشۈرگىلى بولىدۇ.

مەسىلەن:

```
print(id(a))
```

```
def test(num):
    print("a-->num", id(num))
    result = "hello"
```

```
    print("result=hello", id(result))
    return result
a = 10
print("a=10", id(a))
r = test(a)
print("result-->r", id(r))

    ************************************************************************
    a = {"name": "Omer"}
print(id(a))
a["age"] = 18
print(id(a))
a.clear()
print(id(a))
print(a)
a = {}
print(id(a))
```

ئۆزگىرىشچان ۋە ئۆزگەرمەس تىپلار:

ئۆزگەرمەس تىپلار : ئىچكى ساقلىغۇچتكى سانلىق مەلۇماتلارنى ئۆزگەرتىشكە بولمايدۇ.

سان تىپى : **int, bool, float, complex**

تىكست : **string**

گورۇپ : **tuple**

ئۆزگىرىشچان تىپلار : ئىچكى ساقلىغۇچتكى سانلىق مەلۇماتلارنى ئۆزگەرتىشكە بولىدۇ.

تىزىملىك : **list**

لۇغەت dict:

ئەسكەرتىش:

لۇغەتنىڭ ئاچقۇچىغا پەقەت ۋە پەقەت ئۆزگەرمەس تىپلارنى ئىشلىتىش كېرەك، ئۆزگىرىشچان تىپلار ئىشلىتىلسە خاتالىق كېلىپ چىقىدۇ.

فونكسىيە hash()

بۇ ئۆزگەرمەس تىپلارنى پارامېتىر قىلىپ قوبۇل قىلىدىغان ۋە قايتۇرۇش قىممىتىگە بىر پۈتۈن سانىنى قايتۇرىدىغان فونكسىيەدۇر.

```
print(hash(1))
print(hash(1))
print(hash("hello"))
print(hash("hello"))
```

يەرلىك ئۆزگىرىشچان قىممەت:

- يەرلىك ئۆزگىرىشچان قىممەت فونكسىيەنىڭ ئىچىدە ئىنىقلىما بىرىلىدۇ ۋە فونكسىيەنىڭ ئىچىدە ئىشلىتىلىدۇ.
- فونكسىيە ئىجرا قىلىنىپ بولغاندىن كېيىن فونكسىيەنىڭ ئىچىدىكى بۇ يەرلىك ئۆزگىرىشچان قىممەت سىستىما تەرىپىدىن قايتۇرىۋېلىنىدۇ.
- ئوخشىمايدىغان فونكسىيە ئوخشاش ئىسىملىك يەرلىك ئۆزگىرىشچان قىممەتكە ئىنىقلىما بېرەلەيدۇ، ھەمدە ئۆزئارا بىر بىرىگە تەسىر قىلمايدۇ.
- يەرلىك ئۆزگىرىشچان قىممەتنىڭ ئۆمرى فونكسىيە ئىجرا قىلىنىشقا باشلانغاندىن تارتىپ ئاخىرلاشقۇچىلىك بولىدۇ. ۋە بۇ جەريانىدا فونكسىيەنىڭ ئىچىدىكى ۋاقىتلىق مەلۇماتلارنى ساقلاش ئۇچۇن ئىشلىتىلىدۇ.

```
def demo1():
num = 10
print("demo1==>num", num)
```

```
def demo2():
    num = 99
    print("demo2 ==> num", num)
    pass

demo1()
demo2()
```

ئوموميي ئۆزگىرىشچان قىممەت:

ئوموميي ئۆزگىرىشچان قىممەت فونكسىيەنىڭ سىرتىدا ئىنىقلىما بىرىلىدۇ ۋە پۈتۈن فونكسىيەلەر بۇ ئۆزگىرىشچان قىممەت ئىشلىتەلەيدۇ.

```
    num = 10
def demo1():
    print("demo1 ==> %d" % num)
def demo2():
    print("demo2 ==> %d" % num)
demo1()
demo2()
```

فونكسىيە پارامېتىرلىرى ۋە قايتۇرۇش قىممىتىنىڭ رولى:

پارامېتىر ۋە قايتۇرۇش قىممىتىنىڭ بار يوقلىقىغا ئاساسەن تۆۋەندىكى 4 تۈرگە بۆلۈنىدۇ:

- پارامېتىر ۋە قايتۇرۇش قىممىتى يوق:

1. پەقەت بىرلا ئىشنى قىلىدۇ. مەسىلەن: تىزىملىكنى كۆرسىتىش دېگەندەك

➢ پارامېتر يوق، قايتۇرۇش قممىتى بار:

ئۇچۇر توپلاش، مەسىلەن: تېمپېراتۇرا ئۆلچەش

➢ پارامېتر بار، قايتۇرۇش قممىتى بار:

➢ پارامېتر بار، قايتۇرۇش قممىتى يوق:

بىردىن كۆپ قممەت قايتۇرۇش:

```
def measure():
    print("باشلاندى")
    temp = 39
    wetness = 50
    print("ئاخىرلاشتى")
    return temp, wetness
```

فونكسىيەنىڭ پارامېترى:

➢ ئەگەر فونكسىيەنىڭ ئىچىدە = تەڭلىك بەلگىسى ئارقىلىق فونكسىيەنىڭ رەسمىي پارامېترغا قممەت بەرسەك ئەمەلىي پارامېترغا تەسىر قىلمايدۇ. مەسىلەن :

```
def demo(num, num_list):
    print("the code in the function")
    num = 100
    num_list = [1, 2, 3]
    print(num)
    print(num_list)
    print("finished")
gl_num = 99
gl_list = [4, 5, 6]
```

```
demo(gl_num, gl_list)
print(gl_num)
print(gl_list)
```

➢ ئەگەر فونكسيەنىڭ ئىچىدە ئۇسۇل ئارقىلىق فونكسيەنىڭ رەسمىي پارامېتىرنىڭ قىممەتىنى ئۆزگەرتسەك ئەھۋالي پارامېتىرنىڭ قىممىتى ھەم ئۆزگىرەيدۇ. مەسىلەن :

```
    def demo(num_list):
        num_list.append(9)
        print(num_list)
gl_list = [1, 2, 3]
demo(gl_list)
print('ok')
print(gl_list)
```

ئەسكەرتىش :

ئەگەر تىزىملىك(**list**) **+=** نى ئىشلەتكەن بولسا، ئەمەلىيەتتە **extend** نى ئىشلەتكەن بولىدۇ، شۇڭا ئەھۋالي پارامېتىرنىڭ قىممىتى ھەم ئۆزگىرەيدۇ. مەسىلەن:

```
    def demo(num, num_list):
    num += num
    num_list += num_list
    print(num)
    print(num_list)
```

```
gl_num = 9
gl_list = [1, 2, 3]
demo(gl_num, gl_list)
print(gl_num)
print(gl_list)
```

فونكسىيەنىڭ كۆڭۈلدىكى پارامېتىرى:

فونكسىيەگە ئىنىقلىما بەرگەندە، مەلۇم بىر پارامېترغا كۆڭۈلدىكى بىر قىممەتنى بەرگىلى بولىدۇ، بۇ پارامېتر كۆڭۈلدىكى پارامېتر دېيىلىدۇ. ئەگەر فونكسىيەنى ئىشلەتكەندە، بۇ پارامېترغا قىممەت بەرمىسەكمۇ فونكسىيە بىز ئاۋال بەرگەن كۆڭۈلدىكى قىممەتنى ئىشلىتىدۇ.

ئەسكەرتىش:

1. كۆڭۈلدىكى پارامېترنى ئىشلەتكەندە، چوقۇم پارامېترلارنىڭ ئەڭ ئاخىرىغا يېزىلىشى كېرەك.

```
def print_info(name, gender=True):
    gender_text = "male"
    if not gender:
        gender_text = "female"
    print("%s is %s" % (name, gender_text))

print_info("Ahmed")
print_info("Osman")
print_info("Ayse", False)
```

2. بېرىدىن كۆپ كۆڭۈلدىكى پارامېترى بار فونكسىيەنى ئىشلەتكەندە، چوقۇم پارامېترلارنىڭ ئىسمىنى يېزىشىمىز كېرەك.

```
    def print_info(name, title="", gender=True):
        gender_text = "male"
        if not gender:
            gender_text = "female"
        print("[%s]%s is %s" % (title, name, gender_text))
print_info("Ahmed")
print_info("Osman")
print_info("Ayse", gender=False)
```

كۆپ خىللاشقان پارامېتىر:

بەزىدە پارامېترلارنىڭ سانى ئىنىق بولمايدۇ، بۇ ۋاقىتتا كۆپ خىللاشقان پارامېترنى ئىشلىتىشكە بولىدۇ. ئەگەر پارامېترنىڭ ئالدىغا بىر * يۇلتۇز يازساق گۇرۇپ(**tuple**)نى قوبۇل قىلالايدۇ. ئەگەر پارامېترنىڭ ئالدىغا ئىككى ** يۇلتۇز يازساق لوغەت(**dictionary**)نى قوبۇل قىلالايدۇ.

```
    def demo(num, *nums, **person):
        print(num)
        print(nums)
        print(person)

demo(1, 2, 3, 4, 5, name="Ahmed", age=18)
```

فونكسىيەنىڭ ئۆزىنى ئۆزى ئىشلىتىشى:

فونكسىيەنىڭ ئىچىدە باشقا فونكسىيەنى ئىشلىتەلەيدۇ، ھەمدە ئۆزىنى ئۆزى ئىشلىتەلەيدۇ.

ئالاھىدىلىكى:

1. ھەرقىتىمدا ئوخشاش كودنى ئىجرا قىلىدۇ، پەقەت ھەرقىتىمدا ئوخشىمىغان قىممەتنىكى پارامېترنى ئىشلىتىدۇ، بۇ ئارقىلىق ئوخشىمىغان نەتىجە چىقىدۇ.

2. مەلۇم بىر شەرت ھازىر بولغاندا فونكسىيە ئىجرا بولمايدۇ(بۇ بەك مۇھىم، بولمىسا ئۆلۈك ئايلانمىغا ئايلىنىپ قالىدۇ).

```
def sum_number(num):
print(num)
if num == 1:
        return
sum_number(num - 1)

sum_number(3)
```

ئوبجەكتقا يۆزلەنگەن ئاساسىي ئۇقۇملار:

ئالدى بىلەن بىز جەريانغا يۆزلەنگەن كودلاش ئۇسۇلى بىلەن تونۇشۇپ چىقايلى.

جەريانغا يۆزلەنگەن ـ (قانداق قىلىدۇ ؟)

4. مەلۇم تەلەپنىڭ بارلىق باسقۇچلارنى باشتىن ـ ئاخىر تەرتىپ بىلەن ئىجرا قىلىدۇ.

5. ئېھتىياجغا ئاساسەن ، بىر قىسىم مۇستەقىل ئىقتىدارغا ئىگە كودلارنى ئارقا ـ ئارقىدىن فونكسىيەگە جۇغلايدۇ.

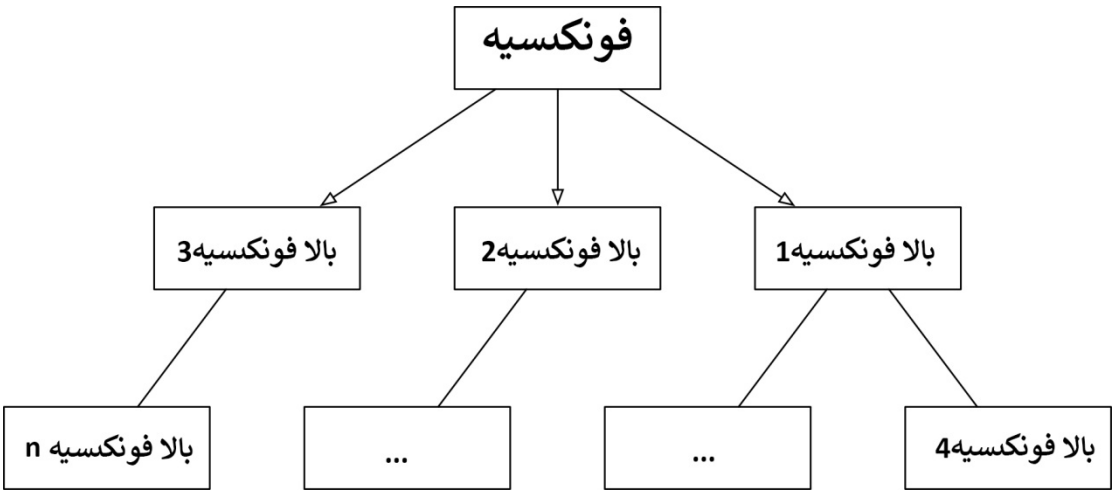
6. ئاخىردا تەرتىپلىك ھالدا ئوخشىمىغان فونكسىيەلەرنى چاقىرىدۇ.

ئالاھىدىلىكى:

1. مەسئۇلىيەتنى بۆلۈشكە ئەمەس ، قەدەم ۋە جەريانغا ئەھمىيەت بېرىدۇ.

2. ئەگەر تەلەپ مۇرەككەپ بولسا ، كود ئىنتايىن مۇرەككەپلىشىپ كېتىدۇ.

3. مۇرەككەپ تۈرلەرنى تەرەققىي قىلدۇرۇشنىڭ مۇقىم قائىدىسى يوق ، شۇڭا تەرەققىيات ئىنتايىن مۇشكۈل!



ئوبجەكتقا يۈزلەنگەن يۈزلەنگەن ـ (كىم قىلدۇ ؟)

فۇنكسىيەگە سىلىشتۇرغاندا ، ئوبجەكتقا يۈزلىنىش تېخىمۇ چوڭ بولغان جەملەش بولۇپ، مەسئۇلىيەتكە ئاساسەن بىر ئوبجەكتقا كۆپ خىل فونكسىيەنى جەملىگىلى بولىدۇ.

1. مەلۇم تەلەپنى ئورۇنداشتىن بۇرۇن ، ئالدى بىلەن مەسئۇلىيەتنى بەلگىلەيدۇ ـ نېمە قىلىش (فونكسىيە).

2. مەسئۇلىيەتكە ئاساسەن ئوخشىمىغان ئوبجەكتنى بەلگىلەيدۇ، ۋە ئوبجەكتقا ئوخشاش بولمىغان فونكسىيەلەرنى جەملەيدۇ.

3. ئاخىردا تەرتىپلىك هالدا ئوخشىمىغان ئوبجەكتلەر ئوخشىمىغان فونكسىيەلەرنى چاقىرىپ ئىشلىتىدۇ.

ئالاهىدىلىكى :

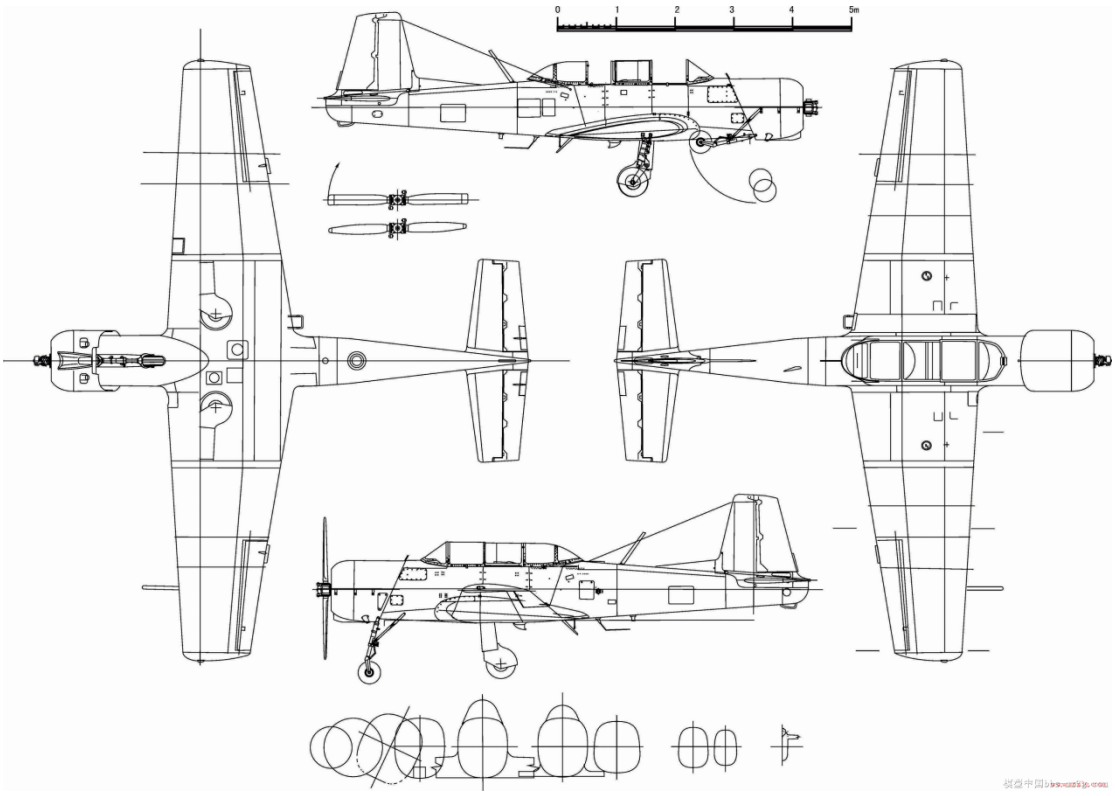
1. ئوبجەكت ۋە مەسئۇلىيەتكە دىققەت قىلىدۇ، ئوخشىمىغان ئوبىيېكتلار ئوخشىمىغان فونكسىيەلەرنى ئۈستىگە ئالىدۇ.

2. ئۇ مۇرەككەپ ئېهتىياج ئۆزگىرىشنى بىر تەرەپ قىلىشقا تېخىمۇ ماس كېلىدۇ، هەمدە مەخسۇس مۇرەككەپ پروگراممالار ئۈچۈن مۇقىم ئۈسۇللار بىلەن تەمىنلەيدۇ.

كلاس ۋە ئوبجەكت:

كلاس: ئوخشاش ئالاھىدىلىك ياكى ھەرىكەتكە ئىگە بىر گۇرۇپپا نەرسىلەرنىڭ ئومۇمىي ئاتىلىشى. ئۇ ئابستراكت بولۇپ ، بىۋاستە ئىشلىتىشكە بولمايدۇ.

كلاس بولسا ئايروپىلان ياساشتىكى چىرتىيوجغا باراۋەر ، ئۇ بىر قېلىپ بولۇپ ، جىسىملارنى قۇرۇشقا مەسئۇل.



ئوبجەكت بولسا بىر كلاس تەرىپىدىن بارلىققا كەلگەن كونكرېت مەۋجۇتلۇق بولۇپ ، بىۋاستە ئىشلىتىشكە بولىدۇ.

قايسى كلاستىن بارلىققا كەلگەن ئوبجەكتنىڭ شۇ كلاستا ئېنىقلانغان خاسلىقى ۋە ئۇسۇللىرى بار.

ئوبجەكت بولسا چىرتىيوجىدىن ياسالغان ئايروپىلانغا باراۋەر

پروگراممىدا ئالدى بىلەن كلاس ، ئاندىن ئوبجەكت بولۇشى كېرەك.

كلاس ۋە ئوبجەكتنىڭ مۇناسىۋىتى:

كلاس بولسا قېلىپ، ئوبجەكت بولسا مۇشۇ قېلىپ ئارقىلىق ياساپ چىقىلغان. ئاۋال كلاس بولىدۇ، ئاندىن ئوبجەكت بولىدۇ.

كلاس پەقەت بىرتال بولىدۇ، ئەمما ئوبجەكت كۆپ بولالايدۇ.

كلاستا قانداق خاسلىق ۋە ئۇسۇللار ئېنىقلىما بېرىلسە ، ئوبجەكتتە شۇنداق خاسلىق ۋە ئۇسۇللار بولىدۇ.

كلاسنىڭ لاھىيەلىنىشى:

ئوبجەكتقا يۈزلەنگەن لاھىيەنى ئىشلىتىشتىن بۇرۇن ، ئالدى بىلەن تەلەپنى تەھلىل قىلىپ ، پروگراممىدا قانداق كلاسنىڭ بارلىقىنى بىكىتىش كېرەك!

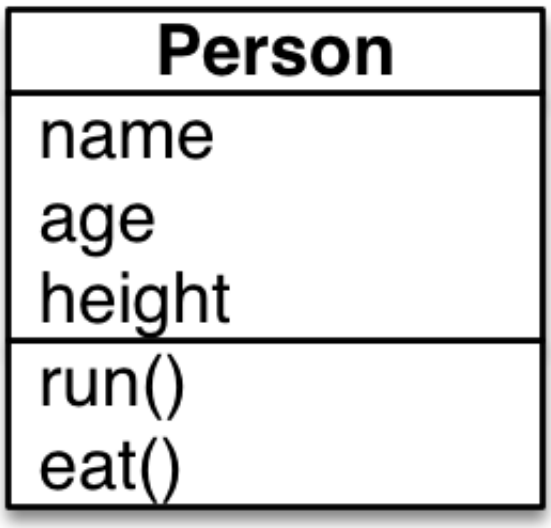
كلاسنى لاھىيەلەشتە تۆۋەندىكى 3 نوقتىغا دىققەت قىلىش كېرەك:

1. كلاسنىڭ ئىسمى
2. خاسلىقى: قانداق ئالاھىدىلىكى بار
3. ئۇسۇل: قانداق ھەرىكەتلىرى بار

مەشىق

ئۆمەر 18 ياش ئىگىزلىكى 1.75 ، ھەرىكۇنى يۈگرەيدۇ. تاماق يەيدۇ

ئوسمان 21 ياش ئىگىزلىكى 1.79، ھەرىكۇنى يۈگرەيدۇ. تاماق يەيدۇ



ئوبجەكتقا يۈزلەنگەن ئاساسي گراممماتىكا:

1. ئىچكى فونكسىيە **dir**:

ئىچكى فۇنكسىيەسى **dir** گە سانلىق مەلۇمات بىرىش ئارقىلىق ئوبجەكتكە تەۋە بارلىق خاسلىق ۋە ئۇسۇللارنى كۆرگىلى بولىدۇ. مەسىلەن:

```
info_list = ["Omer", 21, 1.85]
print(dir(info_list))
```

2. ئىچكى فونكسىيە **--new--**: ئوبجەكت قۇرۇلغاندا ئاپتوماتىك چاقىرىلىدۇ.

3. ئىچكى فونكسىيە **--init--**: ئوبجەكتكە قىممەت بەرگەندە ئاپتوماتىك چاقىرىلىدۇ.

4. ئىچكى فونكسىيە **--del--**: ئوبجەكت ئىچكى ساقلىغۇچتىن ئۆچۈرۈلۈشتىن بۇرۇن ئاپتوماتىك چاقىرىلىدۇ.

5. ئىچكى فونكسىيە **--str--**: **print(**ئوبجەكت**)** قىلغاندا چاقىرىلىدۇ.

```
class Cat:
    def __init__(self, new_name):
        self.name = new_name
        print("%s Come" % self.name)
    def __del__(self):
```

```
        print("%s I am going " % self.name)

    def __str__(self):
        return "I am a cat[%s]" % self.name

tom = Cat("Tom")
print(tom)
```

شەخسى خاسلىق:

شەخسي خاسلىق بىر ئوبيېكتنىڭ ئاشكارىلاشنى خالمايدىغان خۇسۇسيىتى.

شەخسي ئۇسۇللار ئوبيېكتنىڭ ئاشكارىلاشنى خالمايدىغان ئۇسۇللىرى.

```
class Women:
    def __init__(self, name):
        self.name = name
        self.__age = 18
    def __secret(self):
                print("%s Age is %d" % (self.name,
self.__age))
xiaofang = Women("Ahmed")

print(xiaofang._Women__age)
xiaofang._Women__secret()
```

شەخسي خاسلىق ۋە شەخسي ئۇسۇللار:

شەخسي خاسلىق بىر ئوبيېكتنىڭ ئاشكارىلاشنى خالمايدىغان خۇسۇسيىتى.

شەخسي ئۇسۇللار ئوبيېكتنىڭ ئاشكارىلاشنى خالمايدىغان ئۇسۇللىرى.

شەخسي خاسلىق ۋە شەخسي ئۇسۇللارنى پەقەت كلاسنىڭ ئىچىدىلا ئىشلەتكىلى بولىدۇ، كلاسنىڭ سرتىدا ئىشلىتىش چەكلىنىدۇ.

```
class Women:
    def __init__(self, name):
        self.name = name
        self.__age = 18
    def __secret(self):
          print("%s Age is %d" % (self.name, self.__age))



xiaofang = Women("Ayxe")
print(xiaofang._Women__age)
xiaofang._Women__secret()
```

شەخسي خاسلىق ۋە شەخسي ئۇسۇللارنى كلاسنىڭ سرتىدا ئىشلەتمەكچى بولساق، ۋاستىلىق ئىشلەتسەك بولىدۇ. مەسىلەن:

```
class Women:
    def __init__(self, name):
        self.name = name
        self.__age = 18

    def __secret(self):
        print("%s Age is %d" % (self.name, self.__age))
    def display_age(self):
```

```
        print("%s Age is %d" % (self.name, self.__age))

Ayxe = Women("Ayxe")

Ayxe.display_age()
```

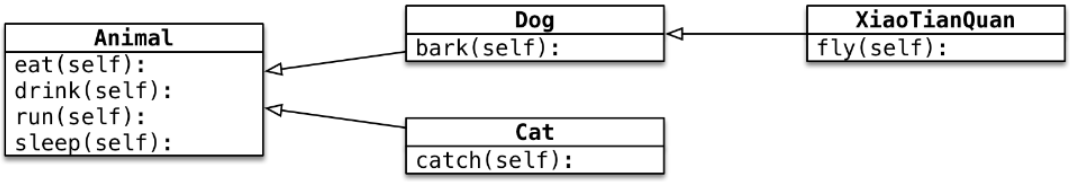
يەككە ۋارىسلىق قىلىش:

بالا كلاس ئاتا كلاسنىڭ بارلىق ئۇسۇل ۋە خاسلىقلىرىغا ۋارىسلىق قىلىدۇ.

ھەمدە بالا كلاس ئۆزىنىڭ ئىھتىياجىغا ئاساسەن خاسلىق ۋە ئۇسۇللارنى يازالايدۇ.

ۋارىسلىق قىلىش ئىشلىتىلمىگەن ھالەت:

ۋارىسلىق قىلىش ئىشلىتىلگەن ھالەت:



ۋارىسلىق قىلىشنىڭ گراماماتىكىسى:

class بالا كلاس(ئاتا كلاس):

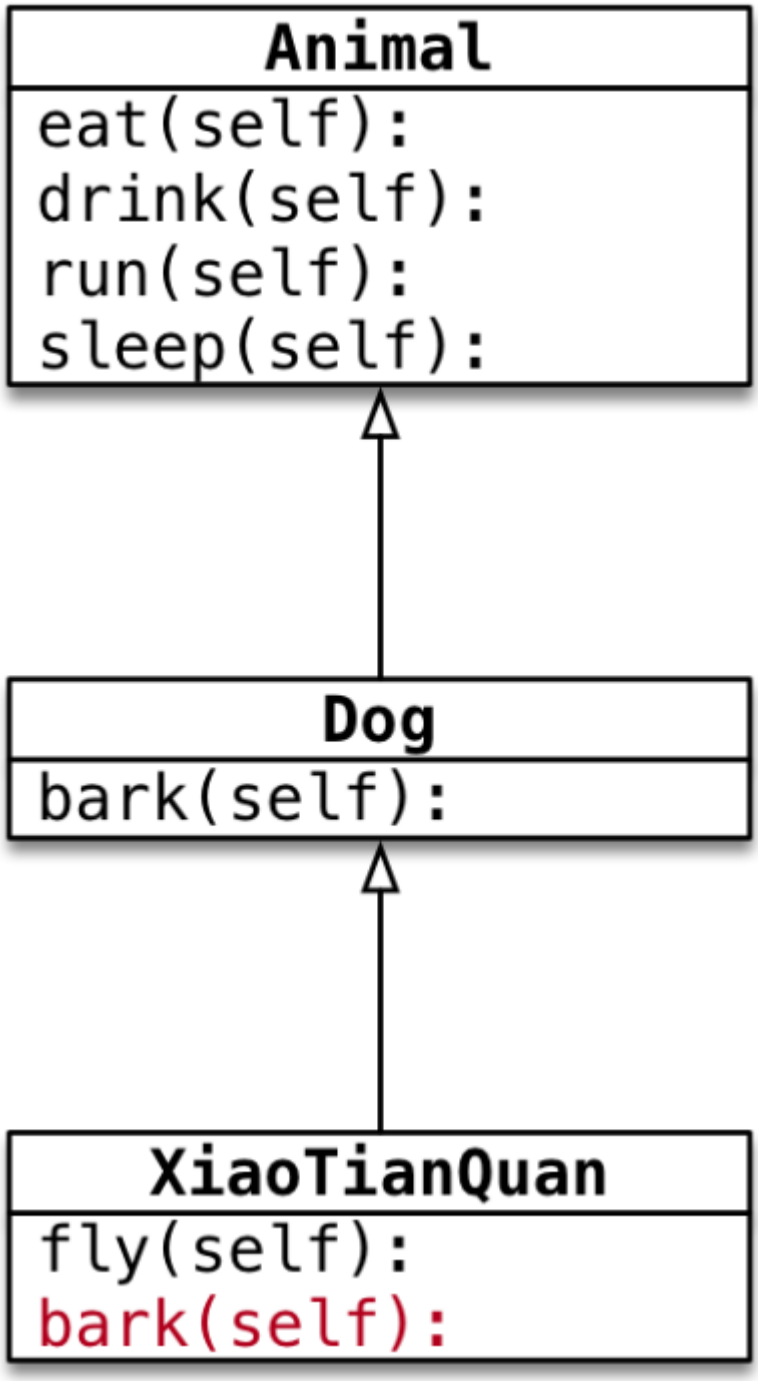
مەسلەن:

```
class Animal:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def eat(self):
        print('eat')
    def drink(self):
        print('drink')
    def run(self):
        print('run')
    def sleep(self):
        print('sleep')
class Dog(Animal):

    @staticmethod
    def bark():
        print('bark')


dog = Dog('Ati', 2)
animal = Animal('tom', 1)


dog.eat()
dog.drink()
```

```
dog.run()

dog.sleep()

dog.bark()

print(dog.name, dog.age)
```

ئۇسۇلنىڭ قايتا يېزىلىشى:

ئاتا كلاسنىڭ ئۇسۇلى بالا كلاسنىڭ ئېھتىياجىغا ماس كەلمىگەندە، بۇ ئۇسۇلنى قايتا يېزىشقا بولىدۇ.



ئاتا كلاسنىڭ ئۇسۇلىنى قايتا يېزىشتا ئىككى خىل ئەھۋال مەۋجۇت:

1. ئاتا كلاسنىڭ ئۇسۇلىنى پۈتۈنلەي قايتا يېزىش:

كونكرېت قايتا يېزىش شەكلى: بالا كلاستا ئاتا كلاستىكى ئۇسۇلغا ئوخشاش ئىسىمدىكى ئۇسۇلدىن بىرنى ئىنقلىما بىرىمىز ۋە يېزىپ پۈتتۈرىمىز. ئاندىن چاقىرغان ۋاقتىمىزدا بالا كلاستىكى يېڭى يېزىلغان ئۇسۇل چاقىرىلىدۇ. ئاتا كلاستىكى ئۇسۇل چاقىرىلمايدۇ.

2. ئاتا كلاستىكى ئۇسۇلنى كېڭەيتىش:

بالا كلاستا ئاتا كلاستىكى ئۇسۇلنى قايتا يازغاندا، ئۇنىڭدىكى مەلۇم كودلار ئاتا كلاسنىڭ ئۇسۇلىدا بار بولسا، **super()** ئارقىلىق ئاتا كلاسنىڭ ئۇسۇلنى چاقىرىپ، ئاندىن كېيىن ئۆزىمىزنىڭ ئېھتىياجىغا ئاساسەن باشقا كودلارنى يازىمىز.

مەسىلەن:

```
class Animal:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def eat(self):
        print('eat')
    def drink(self):
        print('drink')
    def run(self):
        print('run')

    def sleep(self):
        print('sleep')
```

```
class Dog(Animal):


    def bark(self):
        print('bark')
class Ati(Dog):
    def fly(self):
        print('fly')
    def eat(self):
        super().eat()
        print('eat so much')
ati = Ati('Ati', 2)
ati.eat()
ati.bark()
print(ati.name)
print(ati.age)
```

ئەسكەرتىش:

بالا كلاس بىر ئۇسۇلنى چاقىرغان ۋاقىتتا **python** ئالدى بىلەن بۇ ئۇسۇلنى بالا كلاستىن ئىزدەيدۇ، ئەگەر تاپالمىسا ئاتا كلاستىن ئىزدەيدۇ، ئەگەر تاپالمىسا بوۋا كلاستىن ئىزدەيدۇ. ئەگەر ئۇنىڭدىنمۇ تاپالمىسا خاتالىق چىقىدۇ.

ئاتا كلاسنىڭ شەخسىي خاسلىقى ۋە شەخسىي ئۇسۇللىرى:

بالا كلاس ئۆزىنىڭ ئۇسۇلىدا ئاتا كلاسنىڭ شەخسىي خاسلىقى ۋە شەخسىي ئۇسۇللىرىغا بىۋاستە ئىرىشەلمەيدۇ. ئامما ئاتا كلاسنىڭ ئاممىۋى ئۇسۇلى ئارقىلىق ۋاستىلىق ئىرىشەلەيدۇ.

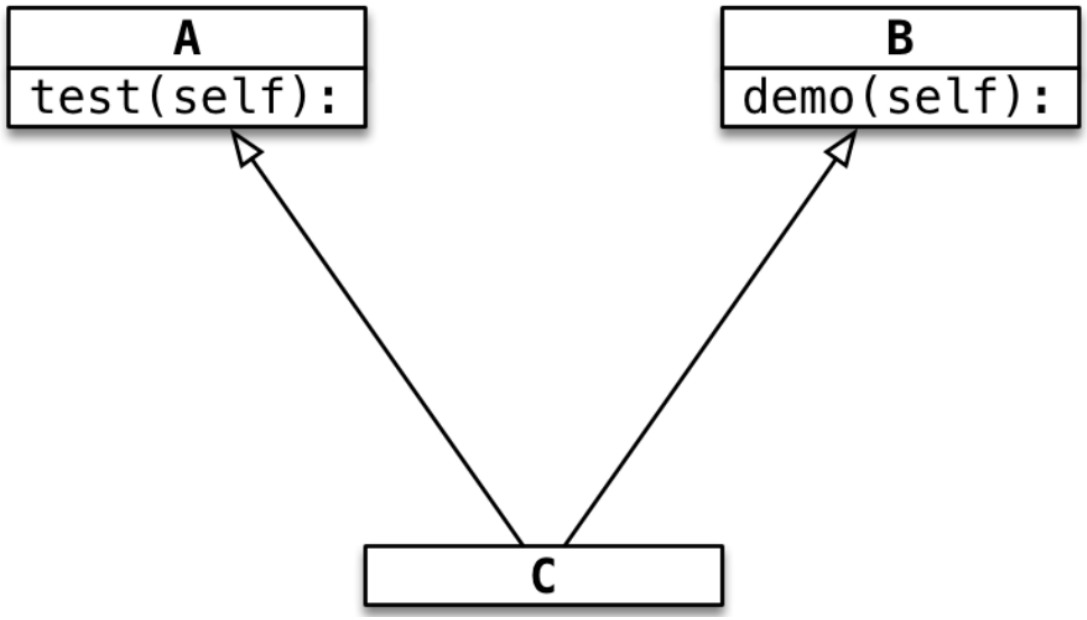
```
class Animal:
    def __init__(self, name):
```

```
        self.name = name
        self.__age = 2

    def eat(self):
        print('eat')

    def drink(self):
        print('drink')

    def run(self):
        print('run')

    def sleep(self):
        print('sleep')

    def display_age(self):
        print(self.__age)

class Dog(Animal):

    def bark(self):
        print('bark')

    # def display(self):
```

```
    #     print(self.__age)

class Ati(Dog):

    def fly(self):
        print('fly')

    def eat(self):

        super().eat()

        print('eat so much')

ati = Ati('Ati')
momo = Ati('Momo')
ati.eat()
momo.eat()

ati.eat()
print(ati.name)
# print(ati.__age)
# ati.display_age()
ati.display_age()
# ati.display()
```

كۆپ خىل ۋارىسلىق قىلىش:

بالا كلاس بىردىن كۆپ كلاسقا ۋارىسلىق قىلالايدۇ. ھەمدە بارلىق ئاتا كلاسنىڭ خاسلىقى ۋە ئۇسۇلىغا ھەم ۋارىسلىق قىلالايدۇ.




```
class A:

    def test(self):

        print("Test")



class B:

    def demo(self):

        print("Demo")
```

```
class C(A, B):


    def abc(self):
        print('Abc')


c = C()
c.test()
c.demo()
c.abc()
```

ئەمەلىي خاسلىق ۋە ئەمەلىي ئۇسۇل:

ئەمەلىي خاسلىق : ئوبجەكتنىڭ خاسلىقى

ئەمەلىي ئۇسۇل : ئوبجەكتنىڭ ئۇسۇلى

ھەر بىر ئوبجەكتنىڭ ئوخشىمىغان خاسلىقىنى ساقلاش ئۈچۈن ئۆزىگە خاس ئىچكى ساقلىغۇچ بوشلۇقى بار.

ئىچكى ساقلىغۇچتا كۆپ خىل ئوبجەكتنىڭ ئۇسۇلى پەقەت بىرلا. ئۇسۇلنى ئىشلەتكەندە، ئوبجەكت ئارقىلىق ئۇسۇلنى چاقىرىپ ئىشلىتىمىز.

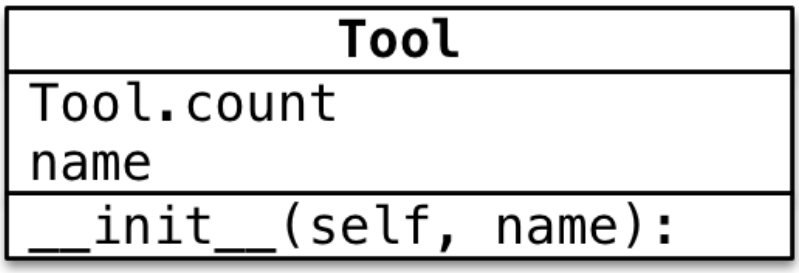
كلاسنىڭ خاسلىقى ۋە كلاسنىڭ ئۇسۇلى:

1. كلاسنىڭ خاسلىقى كلاستا ئېنىقلىما بىرىلگەن خاسلىق.

ئادەتتە بۇ كلاسقا مۇناسىۋەتلىك ئالاھىدىلىكلەرنى خاتىرىلەشكە ئىشلىتىلىدۇ

كلاسنىڭ خاسلىقى كونكرېت ئوبجەكتنىڭ ئالاھىدىلىكىنى خاتىرىلەشكە ئىشلىتىلمەيدۇ.

مەسىلەن:




```
class Tool(object):
    count = 0
    def __init__(self, name):
        self.name = name
        Tool.count += 1
```

كلاسنىڭ خاسلىقىغا ئېرىشىشنىڭ ئىككى خىل ئۇسۇلى بار. مەسىلەن:

1. Tool.count

2. **tool = Tool()**

   tool.count

ئىككىنجى خىل ئۇسۇل تەۋسىيە قىلىنمايدۇ.

ئەسكەرتىش:

ئەگەر **object.class = value** تەقسىملەش بايانىتىنى ئىشلەتسەك ، ئۇ پەقەت ئوبجەكتقا خاسلىق قوشىدۇ ھەمدە كلاسنىڭ خاسلىقىنىڭ قىممىتىگە تەسىر كۆرسەتمەيدۇ.

```
class Tool(object):
    count = 0
    name = ''
    def __init__(self, name):
        self.name = name
        Tool.count += 1
```

```
tool1 = Tool("palta")
tool2 = Tool("bolka")
tool3 = Tool("qilak")

tool3.count = 99
print("Total tools is %d" % tool3.count)  # 99
print("===> %d" % Tool.count)  # 3
Tool.name = 'Ali'
print(Tool.name)  #  Ali
print(tool2.name)  # bolka
```

2. كلاسنىڭ ئۇسۇلى كلاستا ئېنىقلىما بىرىلگەن ئۇسۇل. بۇ ئۇسۇلدا كلاسنىڭ خاسلىقىنى بۇاستە ئىشلەتكىلى بولىدۇ، ۋە باشقا ئۇسۇللانى چاقىرغىلى بولىدۇ.

ئىشلىتىش شەكلى تۆۋەندىكىچە:

- كلاسنىڭ ئۇسۇلىنى زىننەتلىگۈچى **@classmethod** بىلەن بەلگە قىلىپ ، تەرجىمانغا بۇنىڭ بىر كلاسنىڭ ئۇسۇلى ئىكەنلىكىنى بۇلدۇرۇش كېرەك.
- كلاسنىڭ ئۇسۇلىنىڭ بىرىنچى پارامېترى **cls** بولۇشى كېرەك.
- كلاسنىڭ ئۇسۇلىنىڭ ئىچىدە، **cls** ئارقىلىق ئوبجەكتنىڭ خاسلىقىغا ئىرىشكىلى بولىدۇ.
- **cls** ئارقىلىق كلاستىكى باشقا ئۇسۇللارنىمۇ چاقىرغىلى بولىدۇ.

مەسىلەن:

```
class Tool:
    count = 0
    @classmethod
    def show_tool_count(cls):
```

```
        print("Total tool is %d" % cls.count)
    def __init__(self, name):
        self.name = name
        Tool.count += 1

tool1 = Tool("palta")
tool2 = Tool("bolka")
Tool.show_tool_count()
```

تۇراقلىق ئۇسۇل:

ئەگەر كلاستىكى بىر ئۇسۇلدا، ئوبجەكتنىڭ خاسلىقى ياكى ئوبجەكتنىڭ ئۇسۇلىنى ئىشلىتىشنىڭ ھاجىتى يوق بولسا، ھەمدە

كلاسنىڭ خاسلىقى ياكى كلاسنىڭ ئۇسۇلىنى ئىشلىتىشنىڭ ھاجىتى يوق بولسا،

بۇ ۋاقىتتا ، بۇ ئۇسۇلنى تۇراقلىق ئۇسۇل قىلىپ يېزىشقا بولىدۇ.

ئىشلىتىش شەكلى تۆۋەندىكىچە:

- تۇراقلىق ئۇسۇللارغا بېزەكچى **staticmethod@** بەلگىسى قويۇلۇشى كېرەك ، تەرجىمانغا بۇنىڭ تۇراقلىق ئۇسۇل ئىكەنلىكىنى بۇلدۇرۇش كېرەك.
- كلاسنىڭ ئىسمى ئارقىلىق تۇراقلىق ئۇسۇلغا ئىرىشكىلى بولىدۇ.

مەسىلەن:

```
class Dog(object):
    @staticmethod
```

```
    def run():
        print("run..")
Dog.run()
```

يەككىلىك (Singleton):

لايىھىلەش ئەندىزىسى: ئىلگىرىكى خىزمەتلەرنىڭ خۇلاسىسى ۋە ئىنچىكەلىكى ، ئادەتتە ، كەڭ تارقالغان لايىھىلەش ئەندىزىسى مەلۇم مەسىلىنىڭ پىشقان ھەل قىلىش چارىسى.

لايىھىلەش ئەندىزىسىنى ئىشلىتىشتىكى مەقسەت كودنى قايتا ئىشلىتىش ، كودنى باشقىلارنىڭ چۈشىنىشىگە قولايلىق يارىتىش ۋە كودنىڭ ئىشەنچلىكلىكىگە كاپالەتلىك قىلىش.

يەككىلىك لايىھىلەش ئەندىزىسى:

مەقسەت ـ كلاس قۇرغان ئوبيېكتنىڭ سىستېمىدا پەقەت بىرلا مىسالى بولسۇن.

ھەر قېتىم ئوبيېكت قۇرغاندا ئىچكى ساقلىغۇچتىكى ئادرېسى ئوخشاش بولىدۇ.

يەككىلىك لايىھىلەش ئەندىزىسىنىڭ قوللىنىش ئورۇنلىرى:

- مۇزىكا قويغۇچ ئوبيېكتى
- ئەخلەت ساندۇقى ئوبيېكتى
- پرىنتېر ئوبيېكتى

__new__ ئۇسۇلى:

- كلاس نامىنى ئىشلىتىپ ئوبيېكت قۇرغاندا ، **Python** تەرجىمانى ئالدى بىلەن --**new**-- ئۇسۇلىنى چاقىرىپ ، ئوبيېكتكە بوشلۇق تەقسىملەيدۇ.
- --**new**-- بولسا **object** تەمىنلىگەن ئىچكى تۇراقلىق ئۇسۇل. ئۇنىڭ ئاساسلىق ئىككى رولى بار:
  1. ئوبيېكتكە ئىچكى ساقلىغۇچتا بوشلۇق تەقسىملەش
  2. ئوبيېكتنىڭ ئادرىسىنى قايتۇرۇش
- **Python** تەرجىمانى ئوبيېكتنىڭ ئادرىسىغا ئېرىشكەندىن كېيىن ، ئۇنى بىرىنجى پارامېتىر قىلىپ --**init**-- ئۇسۇلىغا بېرىش.

--**New**-- ئۇسۇلىنى قايتا يېزىش كودى ناھايىتى مۇقىم!

--New-- ئۇسۇلىنى قايتا يازغاندا چوقۇم مۇشۇ شەكىلدە يېزىش لازىم:

```
return super().__new__(cls)
```

بولمىسا ، Python تەرجىمانى تەقسىملەنگەن ئوبجەكتنىڭ ئادىرسغا ئېرىشەلمەيدۇ، ھەمدە ئوبجەكتنىڭ --init-- ئۇسۇلىنى چاقىرمايدۇ.

```
class MusicPlayer(object):
    instance = None
    init_flag = False    # True

    def __new__(cls, *args, **kwargs):
        if cls.instance is None:
            cls.instance = super().__new__(cls)
        return cls.instance

    def __init__(self):
        if MusicPlayer.init_flag:
            return
        print('init music player')
        MusicPlayer.init_flag = True


player1 = MusicPlayer()
player2 = MusicPlayer()
player3 = MusicPlayer()
print(player1)
print(player2)
print(player3)
```

بنورماللىق(abnormal):

پروگراما ئىجرا بولۇۋاتقاندا ، **Python** تەرجىمانى خاتالىققا يولۇقسا ، ئۇ پروگراممىنىڭ ئىجرا قىلىنىشىنى توختىتىدۇ ۋە بىر قىسىم خاتالىق ئۇچۇرىنى ئەسكەرتىدۇ.

پروگراما ئىجرا قىلىشنى توختىتىدۇ ۋە خاتالىق ئۇچۇرى بېرىدۇ. بىز بۇنى بنورماللىقنى ئىتتىش دەيمىز.

پروگراما تۇزگەنده ، بارلىق ئالاهىده ئەهۋاللارنى ئەتراپلىق بىر تەرەپ قىلىش تەس ، بنورماللىقنى تۇتۇش ئارقىلىق ، بنورمال ئەهۋاللارنى مەركەزلىك بىر تەرەپ قىلىپ ، پروگراممىنىڭ مۇقىملىقى ۋە پۇختا بولۇشىغا كاپالەتلىك قىلغىلى بولىدۇ.

بنورماللىقنى تۇتۇش:

پروگراما تۇزگەنده بەزى كودلارنىڭ ئىجرا قىلىنىشىنىڭ توغرا ياكى ئەمەسلىكىنى جەزملەشتۈرەلمىگەنده، **try** ئارقىلىق بنورماللىقنى تۇتقىلى بولىدۇ.

مەسىلەن:

```
try:
    # توغرا خاتالىقنى بىلگىلى بولمايدىغان كود
    num = int(input("پۈتۈن سان كىرگۈزۈڭ"))
except:
    # خاتالىقنى بىرتەرەپ قىلىدىغان كود
    print("توغرا پۈتۈن سان كىرگۈزۈڭ")

print("-" * 50)
```

بنورماللىق تېپىنى تۇتۇش:

پروگراما ئىجرا قىلىنغاندا ، ئوخشىمىغان تۈردىكى بنورمال ئەهۋاللارغا يولۇقۇشىمىز مۇمكىن ، ھەمده ئوخشىمىغان تۈردىكى بنورمال ئەهۋاللارنى ئوخشىمىغان شەكىلده بىر تەرەپ قىلىشىمىز مۇمكىن.

مەسىلەن:

```
try:
    num = int(input("پۈتۈن سان كىرگۈزۈڭ "))
    result = 8 / num
    print(result)
except ZeroDivisionError:
    print("نۆل خاتالىقى")
except ValueError:
    print("توغرا پۈتۈن سان كىرگۈزۈڭ")
```

نامەلۇم خاتالىقلارنى تۇتۇش:

پروگراما تۈزگەندە بارلىق خاتالىقلارنى مۆلچەرلەش يەنىلا تەس. ئەگەر پروگراممدا ھەر قانداق خاتالىق بولۇشىدىن قەتئىينەزەر يەنىلا نورمال ئىجرا بولۇشنى ئۈمىد قىلساق، **except** نى قوشۇپ يىزىشىمىز كېرەك.

مەسىلەن:

```
try:
    num = int(input("پۈتۈن سان كىرگۈزۈڭ "))
    result = 8 / num
    print(result)
except ZeroDivisionError:
    print("نۆل خاتالىقى")
except ValueError:
    print("توغرا پۈتۈن سان كىرگۈزۈڭ")
except:
    print('نامەلۇم  خاتالىقلارنى بىر تەرەپ  قىلىدۇ')
```

بىنورماللىقنى تۇتۇشنىڭ تولۇق گرامماتىكىسى:

```
try:
    num = int(input("پۈتۈن سان كىرگۈزۈڭ "))     result = 8 /
num
    print(result)
except ValueError:
    print("توغرا پۈتۈن سان كىرگۈزۈڭ")
except Exception as result:
    print("نامەلۇم خاتا'%s" % result)
else:
    print("مۇۋەپپەقىيەتلىك سىناق قىلىندى ")
finally:
    print("چوقۇم ئىجرا قىلىنىدىغان كود")
print("-" * 50)
```

بىنورماللىقنىڭ يۆتكىلىشى:

فۇنكسىيە / ئۇسۇلنى ئىجرا قىلىش جەريانىدا بىنورمال ئەھۋال كۆرۈلسە ، بۇنى ئىقتىدار فۇنكسىيە / ئۇسۇلنىڭ چاقىرغۇچىغا تاپشۇرۇلىدۇ.

مەسىلەن:

```
def demo1():
    return int(input("پۈتۈن سان كىرگۈزۈڭ "))
def demo2():
    return demo1()
try:
    print(demo2())
```

```
except Exception as result:
    print("نامەلۇم خاتا'%s" % result)
```

بىنورماللىقنى ئىتتىش:

پروگراممىا تۈزگەندە **Python** تەرجىمانى كود ئىجرا قىلىش جەريانىدا بىنورماللىقنى ئاتقاندىن سىرت، بىزمۇ ئاكتىپلىق بىلەن پروگراممىنىڭ ئۆزگىچە ئېھتىياجىغا ئاساسەن بىنورماللىقنى ئاتالايمىز

ئىتتىش ئۇسۇلى:

Python تەمىنلىگەن كلاس **Exception** بىلەن بىر ئوبجەكت قۇرىمىز، ۋە بۇ ئوبجەكتنى **raise** ئارقىلىق ئاتىمىز.

مەسىلەن:

```
def input_password():
    pwd = input("input password: ")
    # 2. password >= 8
    if len(pwd) >= 8:
        return pwd
    # 3. password < 8 abnormal
    print("Abnormal")
    # 1 create object
    ex = Exception("less than 8 charactor")
    # 2> raise
    raise ex
try:
    print(input_password())
except Exception as result:
    print(result)
```

```
# input_password()
```

ھۆججەت(file):

كومپيۇتوردىكى ھۆججەت بولسا ساقلىغۇچتىكى بىر بۆلەك مەلۇمات.

ساقلىغۇچ: **HDD, SSD, U disk, Optical Disk**

ھۆججەت كومپيۇتوردا ئىككىلىك سان ھالىتىدە ساقلىنىدۇ.

ھۆججەتنىڭ ئاساسلىق مەشغۇلاتى:

كومپيۇتوردا ھۆججەتنىڭ مەشغۇلاتى ئىنتايىن مۇقۇم:

- ھۆججەتنى ئېچىش،
- يېزىش، ئوقۇش،
- تاقاش،

Python دا ھۆججەت مەشغۇلاتى ئۈچۈن ئىشلىتىدىغان 1 فونكسىيە 3 ئۇسۇل:

open فونكسىيە ھۆججەتنى ئېچىشقا مەسئۇل، ھەمدە ھۆججەت ئوبجەكتىنى قايتۇرىدۇ

read ئۇسۇلى ھۆججەتكى مەزمۇننى ئوقۇپ **RAM** غا ساقلايدۇ.

write ئۇسۇلى مەزمۇننى ھۆججەتكە يازىدۇ.

close ئۇسۇلى ھۆججەتنى تاقايدۇ.

مەسىلەن:

```
1. file = open("D:\Okul\python_teaching\code\me.txt")
2. text = file.read()
print(text)
3. file.close()
```

```
1. file = open("D:\Okul\python_teaching\code\me.txt", 'a')
2. file.write("123 hello")
3. file.close()
```

ئەسكەرتىش:

- ئەگەر ھۆججەتنى تاقاشنى ئۇنتۇپ قالساق ، سىستېما مەنبەلىرى خورايدۇ ۋە كېيىنكى ھۆججەتنى ئوقۇش تەسىرگە ئۇچرايدۇ.
- **read** ئۇسۇلى ئىجرا قىلىنغاندىن كېيىن ، ھۆججەت كۆرسەتكۈچى ھۆججەتنىڭ ئاخىرىغا يۆتكىلىدۇ.

ھۆججەت كۆرسەتكۈچى:

ھۆججەتنى قايسى ئورۇندىن باشلاپ ئوقۇيدىغانلىقىنى بەلگىلەيدىغان ئىشارەت.

ھۆججەت تۇنجى قېتىم ئېچىلغاندا ، ھۆججەت كۆرسەتكۈچى ئادەتتە ھۆججەتنىڭ بېشىنى كۆرسىتىدۇ.

read ئۇسۇلى ئىجرا بولغاندا ، ھۆججەت كۆرسەتكۈچى ھۆججەتنىڭ ئاخىرىغا يۆتكىلىدۇ

سۇئال:

ئەگەر **read** ئۇسۇلى بىر قېتىم ئىجرا قىلىنىپ ، بارلىق مەزمۇنلار ئوقۇلسا ، **read** ئۇسۇلىنى قايتا چاقىرىش ئارقىلىق مەزمۇنغا ئېرىشكىلى بولامدۇ؟

```
file = open("D:\Okul\python_teaching\code\me.txt")
text = file.read()
print(text)
print(len(text))
print("-" * 50)
text = file.read()
print(text)
print(len(text))
file.close()
```

جاۋاب: بولمايدۇ. بىرىنچى قېتىم ئوقۇپ بولغاندىن كېيىن ، ھۆججەت كۆرسەتكۈچى ھۆججەتنىڭ ئاخىرىغا يۆتكىلىدۇ

، قايتا چاقىرغاندا ھېچقانداق مەزمۇن ئوقۇلمايدۇ.

ھۆججەتنى ئېچىش ھالىتى:

ئېچىش شەكلى	چۈشەندۈرلىشى
r	پەقەتلا ئوقۇش ھالىتىدە ئېچىش. ھۆججەت كۆرسەتكۈچى ھۆججەتنىڭ بېشىدا قويۇلىدۇ ، بۇ defual ھالەت. ئەگەر ھۆججەت مەۋجۇت بولمىسا ، خاتالىق چىقىدۇ.
w	پەقەتلا يېزىش ھالىتىدە ئېچىش. ئەگەر ھۆججەت مەۋجۇت بولسا ئۇ قايتا يېزىلىدۇ. ئەگەر ھۆججەت مەۋجۇت بولمىسا ، يېڭى ھۆججەت قۇرۇدۇ.
a	قوشۇش ھالىتىدە ئېچىش. ئەگەر ھۆججەت مەۋجۇت بولسا ، ھۆججەت كۆرسەتكۈچى ھۆججەتنىڭ ئاخىرىغا قويۇلىدۇ. ئەگەر ھۆججەت مەۋجۇت بولمىسا ، يېڭى ھۆججەت قۇرۇپ يازىدۇ.
r+	ئوقۇش ۋە يېزىش ھالىتىدە ئېچىش. ھۆججەت كۆرسەتكۈچى ھۆججەتنىڭ بېشىدا قويۇلىدۇ. ئەگەر ھۆججەت مەۋجۇت بولمىسا ، خاتالىق چىقىدۇ.
w+	ئوقۇش ۋە يېزىش ھالىتىدە ئېچىش. ئەگەر ھۆججەت مەۋجۇت بولسا ئۇ قايتا يېزىلىدۇ. ئەگەر ھۆججەت مەۋجۇت بولمىسا ، يېڭى ھۆججەت قۇرۇدۇ.
a+	ئوقۇش ۋە يېزىش ھالىتىدە ئېچىش. ئەگەر ھۆججەت مەۋجۇت بولسا ، ھۆججەت كۆرسەتكۈچى ھۆججەتنىڭ ئاخىرىغا قويۇلىدۇ. ئەگەر ھۆججەت مەۋجۇت بولمىسا ، يېزىش ئۇچۇن يېڭى ھۆججەت قۇرۇدۇ

ئەسكەرتىش:

ھۆججەت كۆرسەتكۈچىنى دائىم يۆتكەش ھۆججەتنىڭ ئوقۇش ۋە يېزىش ئۇنۇمىگە تەسىر كۆرسىتىدۇ. شۇڭا پروگراممىا يېزىش جەريانىدا ھۆججەتلەر كۆپىنچە ۋاقىتتا ئوقۇش ياكى يېزىش ھالىتىدىلا ئىچىلىدۇ.

ھۆججەت مەزمۇنىنى قۇر بويىچە ئوقۇش:

read ئۇسۇل ھۆججەتتىكى بارلىق مەزمۇنلارنى بىرلا قىتىمدا RAM غا ساقلايدۇ، ئەگەر ھۆججەت بەك چوڭ بولسا ، ئىچكى ساقلىغۇچتىكى بوشلۇق ئىگەللاش ئىنتايىن ئېغىر بولىدۇ.

Readline ئۇسۇل بىر قىتىمدا بىر قۇر ئوقۇيدۇ، بۇ ئۇسۇل ئىجرا قىلىنغاندىن كېيىن ، ھۆججەت كۆرسەتكۈچى كېيىنكى قۇرغا يۆتكىلىپ ، قايتا ئوقۇشقا تەييارلىنىدۇ.

```
file = open("D:\Okul\python_teaching\code\me.txt", 'r+')
while True:
    text = file.readline()
if not text:
        break
    print(text, end='')

file.close()
```

ھۆججەتلەرنى كۆچۈرۈش:

1. مەۋجۇت ھۆججەتنى ئېچىش ،

2. ۋە بۇ ھۆججەتنى تولۇق ئوقۇش،

3. ۋە باشقا ھۆججەتكە يېزىش،

مەسلەن:

```
file_read = open("D:\Okul\python_teaching\code\me.txt")
file_write = open("D:\Okul\python_teaching\code\me1.txt", 'w')
# 2.
text = file_read.read()
file_write.write(text)
# 3.
```

```
file_read.close()

file_write.close()
```

چوڭ ھۆججەتلەرنى كۆچۈرۈش:

```
file_read = open("D:\Okul\python_teaching\code\me.txt")

file_write = open("D:\Okul\python_teaching\code\me1.txt", 'w')

# 2.

while True:

        text = file_read.readline()

    if not text:

            break

    file_write.write(text)

    file_read.close()

file_write.close()
```

ھۆججەت مەشغۇلاتى:

ئۇسۇل	چۈشەندۈرىشى	مىسال
rename	ھۆججەتنىڭ نامىنى ئۆزگەرتىش	**os.rename**
remove	ھۆججەتلەرنى ئۆچۈرۈش	**os.remove**

مۇندەرىجە مەشغۇلاتى:

ئۇسۇل	چۈشەندۈرىشى	مىسال
listdir	مۇندەرىجە تىزىملىكى	**os.listdir(**مۇندەرىجە ئىسمى**)**
mkdir	مۇندەرىجە قۇرۇش	**os.mkdir(**مۇندەرىجە ئىسمى**)**
rmdir	مۇندەرىجىنى ئۆچۈرۈش	**os.rmdir(**مۇندەرىجە ئىسمى**)**
getcwd	نۆۋەتتىكى مۇندەرىجىگە ئېرىشىش	**os.getcwd()**

chdir	خىزمەت مۇندەرىجىسىنى ئۆزگەرتىش	os.chdir(مۇندەرىجە ئىسمى)
path.isdir	ھۆججەت ياكى ئەمەسلىكىنى ئېنىقلاش	os.path.isdir(ھۆججەت يولى)

eval فونكسىيە:

```
input_str = input("ماتېماتىكىلىق سوئالنى كىرگۈزۈڭ: ")
print(eval(input_str))
```

:assert

assert مەلۇم بىر شەرتنىڭ راست ياكى ئەمەسلىكىنى تەكشۈرۈشكە ياردەم بېرىش ئۈچۈن ئىشلىتىلىدۇ. ئەگەر شەرت خاتا بولسا ، **AssertionError** خاتاسىنى ئاتىدۇ. مەسىلەن:

```
x = 10
assert x > 5   # right
assert x < 5   # raise AssertionError
```

:async ,await

بۇ ئىككى ئاچقۇچلۇق سۆز ماس قەدەمسىز پروگراممىا تۈزۈشتە ئىشلىتىلىدۇ ، ئادەتتە **asyncio** مودۇلى بىلەن بىللە. **async** ماس قەدەمسىز ئىقتىدارنى ئېنىقلاشقا ئىشلىتىلىدۇ ، شۇنىڭ بىلەن بىر ۋاقىتتا بۇ ئىقتىدارنىڭ ئىجرا قىلىنىشىنى توختىتىدۇ ۋە ماس قەدەمسىز مەشغۇلاتنىڭ تاماملىنىشىنى ساقلايدۇ. مەسىلەن:

```
import asyncio
async def example():
    print("Start")
    await asyncio.sleep(2)  # async working
    print("End")
# run async function
asyncio.run(example())
```

:lambda

lambda نامسىز فونكسىيە يېزىشقا ئىشلىتىلىدىغان ئاچقۇچلۇق سۆز. ئۇ **def** جۈملىسىنى ئىشلەتمەي ئاددي

فونكسىيەلەرنى بەلگىلىيەلەيدۇ. مەسلەن:

```
# نامسز فونكسيه ئىككى سانىنڭ يىغىندىسىنى تېپىش
add = lambda x, y: x + y
print(add(3, 5))  #  8
```

nonlocal:

nonlocal گىرەلەشكەن فۇنكسىيەسىدە تاشقى **(global** ئەمەس) ئۆزگەرگۈچى مىقدارنى ئۆزگەرتىش ئۈچۈن ئىشلىتىلىدۇ. ئەگەر **nonlocal** ئىشلەتمىسە، گىرەلەشكەن فۇنكسىيە ئۇنىڭ سىرتىدىكى فۇنكسىيەنىڭ يەرلىك ئۆزگەرگۈچى قىممىتىنى بىۋاستە ئۆزگەرتەلمەيدۇ. مەسلەن:

```
def outer():
    x = 5
    def inner():
        nonlocal x
        x = 10  #
    inner()
    print(x)
    outer()  # 10
```

yield:

ئۇ فۇنكسىيە ئىچىدە گېنېراتور ياساشقا ئىشلىتىلىدۇ. ئۇ بىر ئىقتىدارنىڭ قىممەتنى قايتۇرۇش ۋە نۆۋەتتىكى ئىجرا هالىتىنى ئەسته ساقلاشقا يول قويىدۇ ، قايتا چاقىرغاندا بۇ هالەتتىن ئىجرا قىلىشنى داۋاملاشتۇرىدۇ. مەسلەن:

```
def my-generator():
    yield 1
    yield 2
    yield 3
```

```
for value in my_generator():
    print(value)  #  1, 2, 3
```

__name__ :

--name-- خاسلىقىنىڭ ئالاھىدىلىكى:

- سىناق مودۇلىنىڭ كودى پەقەت سىناق ئەھۋالدا ئىجرا بولىدۇ ، ئىمپورت قىلغاندا ئىجرا قىلىنمايدۇ!
- --name-- python نىڭ ئىچكى خاسلىقى
- ئەگەر ئۇ باشقا ھۆججەتلەر تەرىپىدىن ئىمپورت قىلىنغان بولسا ، --name-- دا بۆلەك ئىسمى ساقلىنىدۇ.
- ئەگەر مەزكۇر ئىجرا قىلىنۋاتقان پروگرامادا --name-- نىڭ قىممىتى --main-- بولىدۇ.