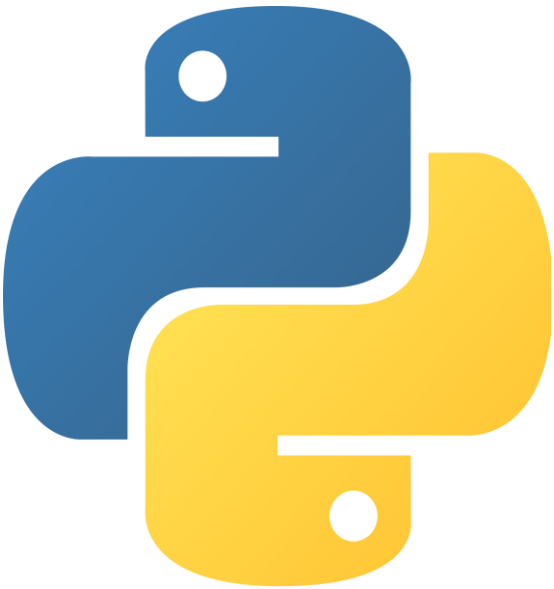

Python
Ուղեցույց

Հեղինակ՝ Վարդուհի Անդրեասյան

Սույն ուղեցույցը նախատեսված է սկսնակներին python ծրագրավորման լեզվի մասին հիմնային տեղեկություն հաղորդելու համար:

Բովանդակություն

Ընդհանուր տեղեկություն

Python-ն ընդհանուր նշանակության բարձր մակարդակի ծրագրավորման լեզու է: Դրա առավելություններից են ծրագրի արագ մշակումը և կոդի հեշտ ընթերցելիությունը: Կառուցվածքային հիմնական գծերն են դինամիկ տիպավորումը, հիշողության ավտոմատ կառավարումը, բացառությունների վերամշակման մեխանիզմը, բարձր մակարդակի տվյալների կառուցվածքը: Ազատ արտոնագրի շնորհիվ այն օգտագործվում է ցանկացած ծրագրում՝ առանց սահմանափակումների: Այն կիրառվում է բազմաթիվ ոլորտներում, այդ թվում՝ մեքենայական ուսուցման (machine learning), պատկերների մշակման ու ճանաչման(image processing) մեջ:

Python-ը նաև.

- **Ինտերակտիվ** է, այսինքն python-ի միջավայրում կարելի է անմիջականորեն գրել կոդը և աշխատեցնել այն,
- **ինտերպրետացվող** է. ծրագիրը մշակվում է կատարման ժամանակ եւ անհրաժեշտություն չկա կոմպիլացնել այն նախքան աշխատեցնելը,
- **օբյեկտ կողմնորոշված** է. սա ծրագրավորման այնպիսի տեխնիկա է, որը ներպարունակում է (encapsulates) կոդը օբյեկտներում,
- ունի մեծ թվով **գրադարաններ**՝ Pillow, Math, NumPy, PyGame, Django, OpenCV

Python-ի օպերատորները

Օպերատորներն ուսումնասիրելու համար պատկերացնենք՝ ունենք a և b փոփոխականներ: a-ի արժեքը 9 է, b-ինը՝ 3:

- Թվաբանական օպերատորներ

\+ - գումարում է երկու օպերանդների արժեքները (a+b = 12),

\- - աջ օպերանդը հանում է ձախից (a-b = 6),

* - բազմապատկում է երկու օպերանդների արժեքները (a*b = 27),

/ - ձախ օպերանդը բաժանում է աջի վրա (a/b = 3),

// - վերադարձնում է բաժանման արդյունքում ստացված թվի ամբողջ մասը (a//b = 3),

% - ձախ օպերանդը բաժանում է աջի վրա ու վերադարձնում մնացորդը (a%b = 0)

- Համեմատության օպերատորներ

== - ստուգում է օպերանդների հավասարությունը և վերադարձնում է «ճիշտ» (true), եթե դրանք հավասար են, հակառակ դեպքում՝ «սխալ» (false) (a == b՝ սխալ),

!= - ստուգում է օպերանդների հավասարությունը և վերադարձնում է «ճիշտ» դրանց հավասար չլինելու դեպքում (a!=b՝ ճիշտ),

> - ստուգում և վերադարձնում է «ճիշտ», եթե ձախ օպերանդը մեծ է աջից (a>b՝ ճիշտ),

< - ստուգում և վերադարձնում է «ճիշտ», եթե ձախ օպերանդը փոքր է աջից (a<b՝ սխալ),

>= - ստուգում և վերադարձնում է «ճիշտ», եթե ձախ օպերանդը աջից մեծ կամ հավասար է (a>=b՝ ճիշտ),

<= - ստուգում և վերադարձնում է «ճիշտ», եթե ձախ օպերանդը աջից փոքր կամ հավասար է (a<=b՝ սխալ),

- Վերագրման օպերատորներ

= - վերագրման պարզագույն օպերատոր, աջ օպերանդը վերագրում է ձախին (a = b, a-ի արժեքը կդառնա b՝ 3),

+= - ձախ օպերանդի արժեքին գումարում է աջի արժեքը և գումարը վերագրում ձախ օպերանդին (a+=b նույնն է, եթե գրենք a = a + b),

-= - ձախ օպերանդի արժեքից հանում է աջի արժեքը և տարբերությունը վերագրում ձախ օպերանդին (a-=b նույնն է, եթե գրենք a = a - b),

Նույն սկզբունքով են աշխատում նաև *=, /=, //=, %= օպերատորները:

- Տրամաբանական օպերատորներ

and - վարադարձնում է «ճիշտ», եթե երկու օպերանդները ճիշտ են (a>0 and b>0՝ ճիշտ),

or - վարադարձնում է «ճիշտ», եթե երկու օպերանդներից առնվազն մեկը ճիշտ է (a>0 or b<0՝ ճիշտ),

not – ժխտում է օպերանդի արժեքը (not(a>0)՝ սխալ)

- Անդամակցության օպերատորներ

in - վերադարձնում է «ճիշտ», եթե աջ օպերանդի հաջորդականությունն իր մեջ պարունակում է ձախ օպերանդը:

not in - վերադարձնում է «ճիշտ», եթե աջ օպերանդի հաջորդականությունն իր մեջ չի պարունակում ձախ օպերանդը:

- Ինքնության օպերատորներ

is - վերադարձնում է «ճիշտ», եթե երկու օպերանդները հղվում են նույն օբյեկտի վրա,

is not - վերադարձնում է «ճիշտ», եթե երկու օպերանդները չեն հղվում նույն օբյեկտի վրա:

Տվյալների մուտքագրում և արտածում

Ծրագրի մեջ տվյալներ մուտքագրելու համար օգտագործում ենք **input()** ֆունկցիան:

Տվյալների արտածման համար օգտագործում ենք **print()** ֆունկցիան:
print()-ը լռելյայն իր բոլոր արգումենտները տպում է իրարից բացատներով առանձնացրած և իրենից հետո դնում է նոր տողի նշան: Այս հատկանիշները կարելի է փոխել sep և end արգումենտների միջոցով՝

1. print(1, 2, 3) 1 2 3
   print(4, 5, 6) 4 5 6
2. print(1, 2, 3, sep=', ', end='-- ') 1, 2, 3-- 456.
   print(4, 5, 6, sep=' ', end='. ')

Փոփոխականներ

- Փոփոխականին արժեք վերագրելու համար՝

փոփոխականի անուն = արժեք (=-ի փոխարեն կարող է լինել ցանկացած վերագրման օպերատոր),

- Միևնույն արժեքը միաժամանակ մի քանի փոփոխականների վերագրում՝

a = b = c = 7,

- Տարբեր փոփոխականների՝ միաժամանակ տարբեր արժեքների վերագրում՝

a, b, c, = 7, 4, “Արմաթ",

- Փոփոխականին մուտքագրված արժեքի վերագրում՝

a = input(),

Տվյալների տիպեր

- Թվեր (Numbers)
- Տող (String)

- Ցուցակ (List)
- Հավաքածու (Tuple)
- Բառարան (Dictionary)

Թվեր

- int ֊ ամբողջ թվեր,
- long ֊ երկար ամբողջ թվեր,
- float ֊ սահող ստորակետով թվեր

Տողեր

Գրվում են ապաթարցների կամ չակերտների մեջ:

Ենթատող վերցնելու համար օգտագործվում են [ ] եւ [:] օպերատորները ինդեքսներով՝ սկսած 0֊ից:

Ենթատողեր

s = ‘Բարև’

տող	**Բ**	**ա**	**ր**	**և**
Ինդեքսը ձախից	0	1	2	3
Ինդեքն աջից	-4	-3	-2	-1

s[0] ֊ տողի առաջին տարր՝ ‘Բ’

s[-1] ֊ տողի վերջին տարրը ՝ ‘և’

s[1:3] ֊ տողի 1֊ինից մինչև 3֊րդ տարրերը՝ չներառելով 3֊րդը՝ ‘ար’

s[:3] ֊ տողի սկզբից մինչև 3֊րդ տարրը՝ ‘Բար’

s[2:] ֊տողի 2֊րդ տարրից մինչև վերջ՝ ‘րև’

s[::2] ֊ տողի ամեն երկրորդ տարրը՝սկսած սկզբից ‘Բր’

s[::-2] ֊ տողի ամեն երկրորդ տարրը՝ սկսած վերջից ‘ևա’

s[1::2] ֊ տողի ամեն երկրորդ տարրը՝ սկսած 1 ինդեքս ունեցողից ‘աև’

s * 2 ֊ տողը 2 անգամ՝ ‘ԲարևԲարև’

s + ‘տեքստ’ - միացնում է 2 տողերը՝ ‘Բարևտեքստ’

Ներկառուցված մեթոդներ տողերի համար

find() - փնտրում է ենթատողը՝ սկսած տողի սկզբից և վերադարձնում առաջին գտածի ինդեքսը, չգտնելու դեպքում վերադարձնում է -1 (s.find(‘ենթատող’)),

rfind() - փնտրում է ենթատողը՝ տողի վերջից և վերադարձնում առաջին գտածի ինդեքսը (s.rfind(‘ենթատող’)),

replace() - նշված ենթատողը փոխարինում է նորով (s.replace(‘հին’,’նոր’)),

count() - հաշվում է, թե քանի անգամ է նշված ենթատողը հանդիպում տողի մեջ և վերադարձնում քանակը (s.count(‘ենթատող’)),

capitalize() - տողի առաջին տառը դարձնում է մեծատառ (s.capitalize()).

len() - վերադարձնում է տողի նիշերի քանակը (len(s)),

join() - տողի բոլոր նիշերը միացնում է իրար (s.join()),

split() - վերադարձնում է տողի տարրերից կազմված ցուցակ (s.split()),

Ցուցակներ

- Python-ի ամենաճկուն տվյալների տիպն է:
- Գրվում է քառակուսի փակագծերի մեջ՝ ստորակետով բաժանվող արժեքներով:
- Տարրերը կարող են լինել տարբեր տիպերի:
- Տարրերին դիմելու կամ ցուցակից ենթահաջորդականություն վերցնելու համար օգտվել [ ], [:] օպերատորներից ինչպես տողերի դեպքում:
- Ցուցակին տարր ավելացնում ենք append() մեթոդով:
- Կտրման գործողության միջոցով հնարավոր է ցուցակի ենթահաջորդականությանը արժեքներ վերագրել, փոխել ցուցակի չափը կամ ջնջել ցուցակի պարունակությունը:
- Ցուցակից տարր ջնջելու համար օգտագործում ենք del (ըստ ինդեքսի) և remove()(ըստ արժեքի) մեթոդները:

Պայմանական հայտարարություն

- **if հայտարարություն**

Տրամաբանական արտահայտություն → դրույթ(ներ)

```
if տր.արտահայտություն:
        դրույթ(ներ)
```

օրինակ,

```
if a > 7:
        print('Փոխհատուցումը մեծ է յոթից:')
```

- **If … else հայտարարություն**

```
if տր.արտահայտություն:
        դրույթ(ներ)
else:
        դրույթ(ներ)
```

- **elif հայտարարություն**

```
if տր.արտահայտություն:
        դրույթ(ներ)
elif տր.արտահայտություն
        դրույթ(ներ)
else:
        դրույթ(ներ)
```

Կարելի է կիրառել ներդրված if կառուցվածք:

range() ֆունկցիա

range() ֆունկցիան օգտագործում ենք ամբողջ թվերի հաջորդականություններ ստանալու համար:

- **range(թիվ)** - «թիվ»-ը թվերի հաջորդականության քանակն է՝ սկսած 0-ից,
- **range(սկիզբ, վերջ)** - գեներացնում է թվերի հաջորդականություն՝ սկսած «սկիզբ»-ից մինչև «վերջ»՝ չներառելով «վերջ»-ը
- **range(սկիզբ, վերջ, քայլ)** - «քայլ»-ը թվային հաջորդականության քայլն է՝ հարևան թվերի տարբերությունը:

for ցիկլ

```
for փոփոխական in հաջորդականություն:
    ցիկլի մարմին

for i in  range(4):
    print (i)
(կտպվի 0 1 2 3)
```

«հաջորդականության» տեղում կարող է լինել տող, ցուցակ, հաջորդականություն՝ կազմված range() ֆունկցիայով և այլն:

Նախ, հաջորդականության առաջին տարրը վերագրվում է փոփոխականին և կատարվում են ցիկլի մարմնում գրված գործողությունները, այնուհետև փոփոխականին վերագրվում է հաջորդականության մյուս տարրը, կատարվում է գործողությունը. այսպես, մինչև հաջորդականության ավարտը:

while ցիկլ

```
while պայման:
        ցիկլի մարմին

while i > 0:
        print (i)
        i -= 1
```

Կրկնում է ցիկլի մարմնում գրված գործողություն(ներ)ը, քանի դեռ տրված պայմանը ճիշտ է:

Ցիկլի կառավարման օպերատորներ

- **Break**

  Ընդհատում է ցիկլի աշխատանքը:

  Կարող է օգտագործվել և՛ while, և՛ for ցիկլերում:

- **Continue**

  Բաց է թողնում ցիկլի մարմնի մնացած մասի կատարումը եւ անմիջապես անցնում է ցիկլի հաջորդ քայլին:

  Կարող է օգտագործվել և՛ while, և՛ for ցիկլերում:

Ֆունկցիաներ

- Ապահովում են կոդի բազմակի օգտագործման հնարավորություն:
- Ֆունկցիայի բլոկը սկսում է def բանալի բառով, այնուհետև գրվում է անունը, փակագծեր՝ ( ) , որոնց մեջ գրվում են ֆունկցիայի ներմուծման արգումենտները՝ առկայության դեպքում:
- return օպերատորի միջոցով դուրս ենք գալիս ֆունկցիայից: Եթե անհրաժեշտ է, որ ֆունկցիան վերադարձնի ինչ-որ արտահայտություն, ապա return օպերատորից հետո գրում ենք այդ արտահայտությունը: Առանց արգումենտի return-ը համարժեք է «return None»-ին:
- Ֆունկցիան կանչելիս արգումենտները գրում ենք այն հաջորդականությամբ, որով դրանք սահմանել ենք:

Գլոբալ և լոկալ փոփոխականները ֆունկցիայում

- **Գլոբալ** են այն փոփոխականները, որոնք սահմանվել են ֆունկցիայից դուրս: Դրանք ունեն տեսանելիության գլոբալ տիրույթ:
- **Լոկալ** փոփոխականները սահմանվում են ֆունկցիայի մարմնի մեջ և ունեն տեսանելիության լոկալ տիրույթ:

Այսինքն, լոկալ փոփոխականները հասանելի են միայն այն ֆունկցիայի մեջ, որում հայտարարվել են, իսկ գլոբալ փոփոխականները՝ կոդի ցանկացած հատվածում:

Գրադարաններ

Փայթոնում գրադարանները ֆունկցիաների և մեթոդների հավաքածուներ են, որոնք հնարավորություն են տալիս մի շարք գործողություններ կատարել, առանց դրանց համար ինքնուրույն կոդ գրելու:

Գրադարանն օգտագործելու համար նախ անհրաժեշտ է այն ներմուծել փայթնի նիշքի մեջ: Ներմուծման, ֆունկցիաների կիրառման ձևերին ծանոթանալու համար ուսումնասիրենք math գրադարանը: Այս գրադարանը ներառում է մի շարք օժանդակ ֆունկցիաներ՝ կապված թվերով կատարվող գործողությունների հետ:

Գրադարանը կարելի է ներմուծել հետևյալ 3 եղանակներով . Ուշադրություն դարձնել ֆունկցիայի գրելաձևին՝ կախված գրադարանի ներմուծման ձևից:

1.

```
import math
a = math.ceil(4.2)
```

2.

```
from math import ceil
a = ceil(4.2)
```

3.

```
from math import *
a = ceil(4.2)
```

Առաջին տարբերակում ներմուծում ենք ամբողջ գրադարանը, երկրորդ տարբերակում՝ գրադարանի կոնկրետ ֆունկցիա, մի քանի ֆունկցիա ներմուծելու համար կարող ենք դրանք իրարից անջատել ստորակետներով (from math import ceil, floor, pi), իսկ երրորդ տարբերակում՝ գրադարանի բոլոր ֆունկցիաները:

Math գրադարանի ֆունկցիաներ.

- floor(x) ֊ x֊ը կլորացնում է դեպի ներքև, վերադարձնում է x-ից փոքր կամ x֊ին հավասար ամենամեծ ամբողջ թիվը,
- ceil(x) ֊ x֊ը կլորացնում է դեպի վերև, վերադարձնում է x-ից մեծ կամ x֊ին հավասար ամենափոքր ամբողջ թիվը,
- sqrt(x) ֊ վերադարձնում է x֊ի քառակուսի արմատը,
- log(x)
- e
- sin(x)
- asin(x)
- pi և այլն:

Օգտագործված աղբյուրներ՝

- http://ggg.i-gorc.am/#69
- https://snakify.org

Ավելին սովորելու համար՝

http://ggg.i-gorc.am/#69 (հայերեն)

https://snakify.org (անգլերեն)

https://www.datacamp.com (անգլերեն)

http://pythontutor.ru (ռուսերեն)

https://stepik.org