

Programovanie v jazyku Python

Michal Kvasnica

Dnes

Dátový typ `list` (zoznam)

FOR a WHILE slučky

Dátový typ `list` (zoznam)

Reprezentuje jednorozmerové pole ľubovoľných dát

```
zoznam = [ 1, 2, 12, 4, 0 ]
```

Heterogénny dátový typ (môžeme kombinovať dáta):

```
zoznam = [ 1, "retazec", 2.0, [0, 5] ]
```

Zoznam je možné indexovať podobne ako reťazce:

```
zoznam[0]  
zoznam[1:3]  
zoznam[:2]  
zoznam[2:]  
zoznam[-1]  
zoznam[-3:]
```

Dátový typ `list` (zoznam)

Dĺžku zoznamu zistíme funkciou `len`:

```
dlzka = len(zoznam)
```

Prvky zoznamu môžeme meniť:

```
zoznam = [ 1, 2, 12, 4, 0 ]  
zoznam[0] = 5
```

Pri výmene sa môže zmeniť aj dátový typ:

```
zoznam = [ 1, "retazec", 2.0, [0, 5] ]  
zoznam[1] = 5.6
```

Dátový typ `list` (zoznam)

Metóda `append` vloží jeden prvok na koniec zoznamu:

```
>>> zoznam = []  
>>> zoznam.append(1); print(zoznam)  
[1]
```

```
>>> zoznam.append(12); print(zoznam)  
[1, 12]
```

```
>>> zoznam.append([4.0, 5]); print(zoznam)  
[1, 12, [4.0, 5]]
```

```
>>> zoznam.append("abc"); print(zoznam)  
[1, 12, [4.0, 5], 'abc']
```

Dátový typ `list` (zoznam)

Metóda `extend` pridá všetky prvky zoznamu na koniec:

```
>>> zoznam = []; print(zoznam)
[]
```

nie je zoznam!

```
>>> zoznam.extend(1)
TypeError: 'int' object is not iterable
```

ok, je zoznam

```
>>> zoznam.extend([1]); print(zoznam)
[1]
```

```
>>> zoznam.extend([2, 3]); print(zoznam)
[1, 2, 3]
```

```
>>> zoznam.extend("abc"); print(zoznam)
[1, 2, 3, 'a', 'b', 'c']
```

Dátový typ `list` (zoznam)

Zoznamy sa dajú spájať aj pomocou `'+'`:

```
>>> z1 = [1, 2, 3]; z2 = [4.0, "abc"]  
>>> vysledok = z1 + z2; print(vysledok)  
[1, 2, 3, 4.0, 'abc']
```

Pozor: oba spájané objekty musia byť zoznamy:

```
>>> [1, 2, 3] + 4  
TypeError: can only concatenate list (not "int") to list
```

nie je zoznam!

```
>>> [1, 2, 3] + [4]  
[1, 2, 3, 4]
```

ok, je zoznam

Dátový typ `list` (zoznam)

Zoznamy sa dajú násobiť celým číslom:

```
>>> 3*[1, 2, 3]  
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
>>> 0*[1, 2, 3]  
[]
```

```
>>> -1*[1, 2, 3]  
[]
```


Dátový typ `list` (zoznam)

Metóda `insert(i, x)` vloží prvok `x` pred pozíciu `i`

```
>>> zoznam = [1, 2, 3, 4]
>>> zoznam.insert(3, 5.0); print(zoznam)
[1, 2, 3, 5.0, 4]
```

vlož na začiatok

```
>>> zoznam.insert(0, 0); print(zoznam)
[0, 1, 2, 3, 5.0, 4]
```

vlož na koniec

```
>>> zoznam.insert(len(zoznam), -1); print(zoznam)
[0, 1, 2, 3, 5.0, 4, -1]
```

vlož na predposledné miesto

```
>>> zoznam.insert(-1, 9); print(zoznam)
[0, 1, 2, 3, 5.0, 4, 9, -1]
```

Dátový typ `list` (zoznam)

Metóda `pop()` odoberie a vráti jeden prvok z konca zoznamu:

```
>>> zoznam = [1, 2, 3, 4]
>>> prvok = zoznam.pop()
>>> print(prvok)
4
>>> print(zoznam)
[1, 2, 3]
```

zmenil sa aj zoznam

```
>>> dalsi = zoznam.pop()
>>> print(dalsi)
3
>>> print(zoznam)
[1, 2]
```

```
>>> zoznam = []
>>> prvok = zoznam.pop()
IndexError: pop from empty list
```

nedá sa popovať z prázdneho zoznamu

Dátový typ `list` (zoznam)

Metóda `pop(i)` odoberie a vráti jeden prvok z danej pozície:

```
>>> zoznam = [1, 2, 3, 4]
>>> prvok = zoznam.pop(2)
>>> print(prvok)
3
>>> print(zoznam)
[1, 2, 4]
```

odober prvý prvok

```
>>> prvok = zoznam.pop(0)
>>> print(prvok)
1
>>> print(zoznam)
[2, 4]
```

odober posledný prvok

```
>>> prvok = zoznam.pop(-1)
>>> print(prvok)
4
>>> print(zoznam)
[2]
```

Dátový typ `list` (zoznam)

Metóda `index(x)` vráti index prvého výskytu prvku `x`:

```
>>> zoznam = [3, 1, 2, 1, 4]
>>> x = 1
>>> pozicia = zoznam.index(x)
>>> print(pozicia)
1
>>> print(zoznam[pozicia])
1
```

```
>>> zoznam = [3, 1, 2, 1, 4]
>>> pozicia = zoznam.index(0)
ValueError: 0 is not in list
```

prvok nie je v zozname

Dátový typ `list` (zoznam)

Metóda `count(x)` vráti počet výskytov prvku `x`:

```
>>> zoznam = [3, 1, 2, 1, 4]
>>> x = 1
>>> pocet = zoznam.count(x)
>>> print(pocet)
2
```

```
>>> pocet = zoznam.count(0)
>>> print(pocet)
0
```

```
>>> pocet = "abcdacd".count("a")
>>> print(pocet)
2
```

počet výskytov
znaku v reťazci


```
>>> pocet = "abcdacd".count("ab")
>>> print(pocet)
1
```

počet výskytov
podreťazca v reťazci

Dátový typ `list` (zoznam)

Metóda `sort()` usporiada pole od najmenšieho prvku po najväčší (priamo modifikuje zoznam):

```
>>> zoznam = [3, 1, 2, 1, 4]
>>> zoznam.sort()
>>> print(zoznam)
[1, 1, 2, 3, 4]
```



priamo mení zoznam

Metóda `sort(reverse=True)` usporiada pole od najväčšieho prvku po najmenší (priamo modifikuje zoznam):

```
>>> zoznam = [3, 1, 2, 1, 4]
>>> zoznam.sort(reverse=True)
>>> print(zoznam)
[4, 3, 2, 1, 1]
```

Dátový typ `list` (zoznam)

Funkcia `sorted()` vráti nový zoznam usporiadaný od najmenšieho prvku po najväčší

```
>>> zoznam = [3, 1, 2, 1, 4]
>>> novy = sorted(zoznam)
>>> print(novy)
[1, 1, 2, 3, 4]
>>> print(zoznam)
[3, 1, 2, 1, 4]
```

nemení pôvodný zoznam

```
>>> zoznam = [3, 1, 2, 1, 4]
>>> novy = sorted(zoznam, reverse=True)
>>> print(novy)
[4, 3, 2, 1, 1]
```

od najväčšieho
po najmenší

Dátový typ `list` (zoznam)

Metóda `reverse()` prehodí poradie prvkov:

```
>>> zoznam = [3, 0, 2, 1, 4]
>>> zoznam.reverse()
>>> print(zoznam)
[4, 1, 2, 0, 3]
```

```
>>> zoznam = [3, 'abc', 2, [1, 5.0], 4]
>>> zoznam.reverse()
>>> print(zoznam)
[4, [1, 5.0], 2, 'abc', 3]
```

prehadzuje sa iba poradie
hlavného zoznamu

Operácie so zoznamami čísel:

Funkcia `min(zoznam)` vráti najmenší prvok zoznamu:

```
>>> zoznam = [3, 0, 2, 1, 4]
>>> x = min(zoznam)
>>> print(x)
0
```

Funkcia `max(zoznam)` vráti najväčší prvok zoznamu:

```
>>> x = max(zoznam)
>>> print(x)
4
```

Funkcia `sum(zoznam)` vráti sumu prvkov zoznamu:

```
>>> s = sum(zoznam)
>>> print(x)
10
```

Operácie so zoznamami čísel:

`min()` a `max()` pre zoznamy reťazcov:

```
>>> zoznam = ['z', 'aa', 'abc', 'abeceda', 'xyz']
>>> q = min(zoznam)
>>> print(q)
aa
>>> q = max(zoznam)
>>> print(q)
z
```

lingvisticky najmenší prvok

lingvisticky najväčší prvok

Dnes

Dátový typ `list` (zoznam)

FOR a WHILE slučky

FOR slučky v jazyku Python

The diagram illustrates the syntax of a Python for loop with three red callout boxes pointing to specific parts of the code:

- povinná dvojbodka** (mandatory colon) points to the colon at the end of the `for` line.
- blok opakujúcich sa príkazov** (block of repeating commands) points to the indented lines of code.
- tu kód pokračuje** (the code continues here) points to the end of the code block.

```
for prvok in zoznam:  
    # blok opakujucich sa prikazov  
    # premenna "prvok" sa dynamicky meni  
    print('Vzdy pokračuj tymto riadkom')
```

Poznámky:

- príkaz `for` postupne prechádza cez prvky zoznamu
- blok opakujúcich sa príkazov je potrebné odsadiť
- v každom priebehu for-slučky sa mení hodnota iteračnej premennej
- pozor: premenná `prvok` je hodnotou prvku, nie jeho indexom!

FOR slučky v jazyku Python

```
for prvok in [0, 1, 2, 3]:  
    print(prvok)
```

```
0  
1  
2  
3
```

FOR slučky v jazyku Python

```
for prvok in [0, [1, 2], "abc"]:  
    print(prvok)
```

```
0  
[1, 2]  
abc
```

FOR slučky v jazyku Python

pole 3 znakov

```
for prvok in "abc":  
    print(prvok)
```

```
a  
b  
c
```

pole s 1 prvkom

```
for prvok in ["abc"]:  
    print(prvok)
```

```
abc
```

FOR slučky v jazyku Python

```
ludia = ['Jan', 'Marta', 'Zuzana']  
for index in [0, 1, 2]:  
    print("Ahoj {}".format(ludia[index]))
```

```
Ahoj Jan  
Ahoj Marta  
Ahoj Zuzana
```

```
ludia = ['Jan', 'Marta', 'Zuzana']  
for clovek in ludia:  
    print("Ahoj {}".format(clovek))
```


FOR slučky v jazyku Python

Funkcia `range(x)` vráti zoznam od 0 po $x-1$:

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

```
0  
1  
2  
3
```

FOR slučky v jazyku Python

Funkcia `range(a, b)` vráti zoznam od `a` po `b-1`:

```
for i in range(1, 5):  
    print(i)
```

```
1  
2  
3  
4
```

FOR slučky v jazyku Python

Funkcia `range(a, b, c)` vráti zoznam od `a` po `b-1` s krokom `c`

```
for i in range(2, 7, 2):  
    print(i)
```

```
2  
4  
6
```

Príklad

Vytvorte program, ktorý:

- načíta z klávesnice celé číslo
- na obrazovku vypíše do riadku príslušný počet hviezdíčiek

```
opakovani = int(input("Pocet opakovani: "))  
for i in range(opakovani):  
    print("*", end="")  
print() # odriadkovanie
```

Čisté python riešenie:

```
opakovani = int(input("Pocet opakovani: "))  
print("*"*opakovani)
```

Príklad

Vytvorte program, ktorý:

- načíta z klávesnice reťazec
- zistí počet výskytov znaku medzera

```
retazec = input("Vloz retazec: ")
medzery = 0
for i in range(len(retazec)):
    if retazec[i]==" ":
        medzery = medzery + 1
print("Pocet medzier: %d" % (medzery))
```

Lepšie python riešenie:

```
retazec = input("Vloz retazec: ")
medzery = 0
for znak in retazec:
    if znak==" ":
        medzery = medzery + 1
print("Pocet medzier: %d" % (medzery))
```

Príklad

Vytvorte program, ktorý:

- načíta z klávesnice reťazec
- zistí počet výskytov znaku medzera

```
retazec = input("Vloz retazec: ")
medzery = 0
for i in range(len(retazec)):
    if retazec[i]==" ":
        medzery = medzery + 1
print("Pocet medzier: %d" % (medzery))
```

Čisté python riešenie:

```
retazec = input("Vloz retazec: ")
medzery = retazec.count(" ")
print("Pocet medzier: %d" % (medzery))
```

Príklad

Vytvorte program, ktorý:

- načíta z klávesnice počet čísel
- z klávesnice načíta požadovaný počet čísel do poľa
- na obrazovku vypíše pole a priemer jeho prvkov

```
pocet = int(input("Pocet cisel: "))
pole = []
for i in range(pocet):
    prvok = float(input("%d. cislo: " % (i+1)))
    pole.append(prvok)
print("Pole: %s" % (pole))
print("Priemer: %f" % (sum(pole)/len(pole)))
```

FOR slučky v jazyku Python

Funkcia `enumerate(zoznam)` vráti dvojice `index, hodnota`

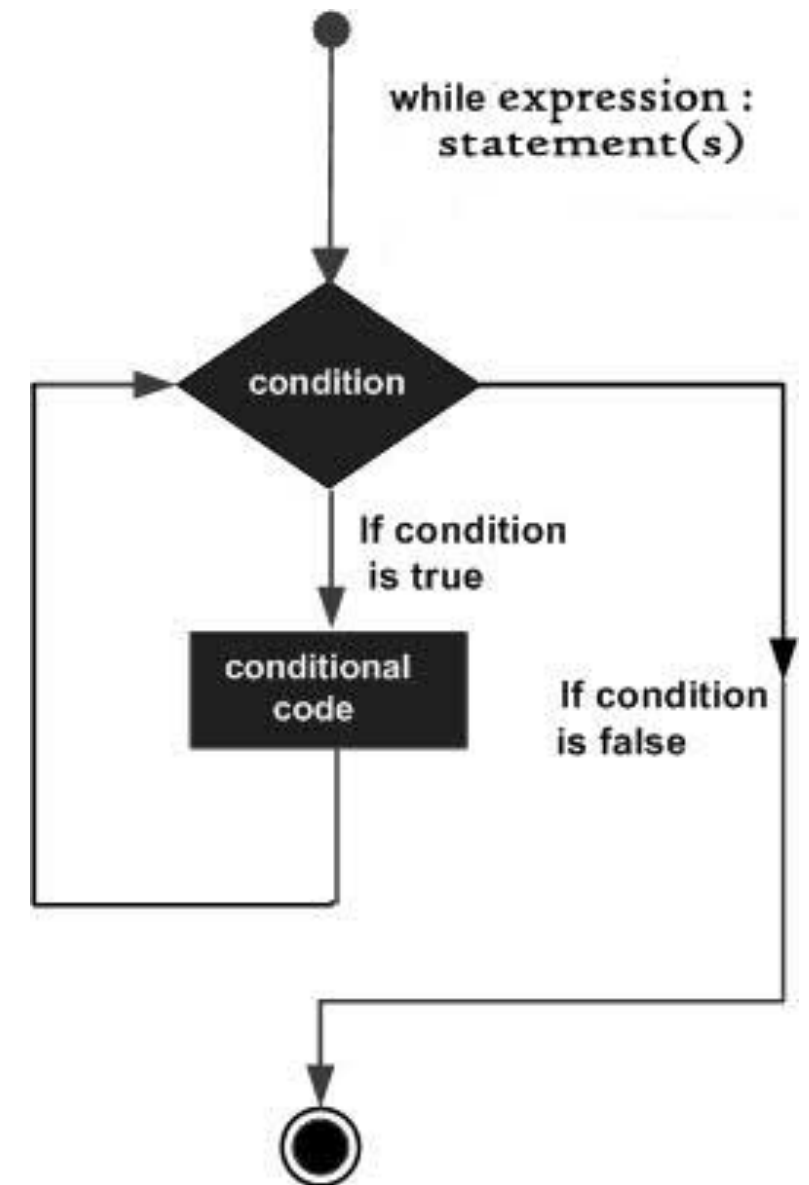
```
zoznam = [1, 5, "abc", [2, 3]]  
for index, hodnota in enumerate(zoznam) :  
    print("index: {}, hodnota: {}".format(index, hodnota))
```

```
index: 0, hodnota: 1  
index: 1, hodnota: 5  
index: 2, hodnota: abc  
index: 3, hodnota: [2, 3]
```


WHILE slučky v jazyku Python

povinná dvojbodka

```
while (podmienka je splnena):  
    # blok opakujucich sa prikazov  
    print('Vzdy pokračuj tymto riadkom')
```



WHILE slučky v jazyku Python

```
pocet = 0
while (pocet < 4):
    print("Pocet: {}".format(pocet))
    pocet = pocet + 1
print("Dovidenia!")
```

```
Pocet: 0
Pocet: 1
Pocet: 2
Pocet: 3
Dovidenia!
```

Príklad

Vytvorte program, ktorý:

- načítava z klávesnice do poľa čísla
- po každom čísle sa spýta, či chceme načítať ďalšie
- na záver vypíše celé pole

```
pole = []
dalsie = "a"
while (dalsie=="a"):
    cislo = float(input("Cislo: "))
    pole.append(cislo)
    dalsie = input("Dalsie? [a/n]: ")
print("Pole: {}".format(pole))
```

Príklad

Vytvorte program, ktorý:

- načítava z klávesnice do poľa čísla
- po každom čísle sa spýta, či chceme načítať ďalšie
- na záver vypíše celé pole

predčasné
ukončenie cyklu

```
pole = []
while True:
    cislo = float(input("Cislo: "))
    pole.append(cislo)
    dalsie = input("Dalsie? [a/n]: ")
    if dalsie=="n":
        break
print("Pole: {}".format(pole))
```