Iterácia a zoznamy

Podobná prednáška

MIT kurz - tuples, lists,. . . (youtube video)

Opakovanie

Ľubovolné množstvo opakovaní sme už docielili pomocou rekurzie, ale toto je pohodlnejšie väčšinou:

```
for i in [1, 2, 3, 4, 5]:
    print(i)
    print(i**2)
```

- `for .. in ..:` je syntax
- `i` je názov premennej (je možné použiť akýkoľvek validný názov)
- [1, 2, 3, 4, 5] je iterovatelný objekt, tu konkrétne zoznam s prvkami 1, 2, 3, 4 a 5.
- v indentovanom bloku sú príkazy, ktoré sa majú opakovať

Iterovatelné znamená, že na tom funguje `for` cyklus.

Indexovanie

```
zoznam = [1, 2, 3, 4]
nulty_prvok = zoznam[0]
prvy_prvok = zoznam[1]

retazec = "12345"
retazec[0]  # funguje rovnako
```

Pokročilé indexovanie -> slicing

```
zoznam = [1, 2, 3, 4, 5, 6, 7, 8]

print(zoznam[1:3])
print(zoznam[1:6:2])
print(zoznam[-1])
print(zoznam[-4:-1])
```

```
zoznam[od : do_ale_bez : krok]
```

Je možné vynechať hodnoty:

```
zoznam[::3]
```

Pri vynechaní sa nastavia praktické hodnoty, t.j.:
`zoznam[0:len(zoznam):1]`

- Viac poznania

range

```
rozsah = range(5)
for i in rozsah:    # 0, 1, 2, 3, 4
    print(i)

zoznam = list(rozsah)  # [0, 1, 2, 3, 4]
```

Tiež je možné špecifikovať začiatok a krok:

```
range(3, 10, 2)
```

len

Funkcia `len` vráti počet prvkov:

```
a = [1, 2, 3]
len(a)   # 3

b = " 1234"
len(b)   # 5

c = (1,)
len(c)   # 1

d = []
len(d)   # 0
```

Dva typy objektov

Objekty bude praktické rozdeliť na dve kategórie:

- meniteľné - tie ktoré je možné zmeniť
- nemeniteľné - . . .

(anglicky mutable / immutable)

Toto nemá nič s premennými! Hovoríme o objektoch – t.j. to čo sa ukrýva v premennej.

To kam ukazuje premenná je možné meniť v Pythone ľubovoľne:

```
a = 12
a = "dvanast"
a = [1, 2, 3]
```

Konečne zmena!

Typy `str`, `tuple`, `bool`, `int`, `float` sú nemeniteľné.

```
text = "12345"
text[0] = "jedna"   # Error

trojica = (1, 2, 3)
trojica[0] = 32      # Error
```

Typ `list` je meniteľný!

```
zoznam = [1, 2, 3, 4]
zoznam[0] = "jedna!"
```

range(len(...))

```
a = [1, 2, 3]
for i in a:
    i = i - 1

print(a)
```

```
a = [1, 2, 3]
for i in range(len(a)):
    a[i] = a[i] - 1

print(a)
```

Metódy zoznamu, ktoré menia zoznam

```
zoznam.append(5)  # prida na koniec
zoznam.pop()  # odstrani posledny element
```

```
vysledok = zoznam.append(5)
print(type(vysledok)) # -> None, zoznam sa zmenil

vysledok = zoznam.pop()
print(type(vysledok)) # -> 5,  zoznam sa zmenil
```

Meniteľnosť prináša viac možností na chyby

```
def pow2(zoznam):
    for i in range(len(zoznam)):
        zoznam[i] = zoznam[i]**2
    return zoznam


x = [1, 2, 3, 4]
y = pow2(x)

for i in range(len(x)):
    print(y[i] - x[i]) # ma byt (x**2 - x)
    # co sa stalo?
```

Kde je chyba a ako ju opraviť?

Vizualizacia cez PythonTutor

Identita zoznamu

```
a = [1, 2, 3]
b = a
print("id(a): ", id(a))
print("id(b): ", id(b))

a.append(b)
print(b)

print("id(a): ", id(a))
print("id(b): ", id(b))

print(b[-1])

print(a is b)
```

Kópie

```
a = [1, 2, 3]

b = []
for i in a:
    b.append(i)
```

alebo

```
a = [1, 2, 3]
b = a[:]   # slice robí kópie
```

alebo

```
from copy import copy
a = [1, 2, 3]
b = copy(a)
```

Zoznam v zozname

```
a = [1, 2, 3]
b = a[:] # alebo iny sposob z pred. slidu

b.append(a)

c = b[:]

a.append(12)
print(b)
print(c)
```

deepcopy

Potrebujeme rekurzívnu kópiu.

Dobré cvičenie, ale niekto to už vymyslel:

```
from copy import deepcopy

c = deepcopy(a)
```

deepcopy

```
from copy import deepcopy
a = [1, 2, 3]
b = deepcopy(a)

b.append(a)

c = deepcopy(b)

a.append(12)
print(b)
print(c)
```

Čo sa vypíše?

Prečo nehovoríme o nemeniteľných objektoch tiež?

Z knižnice copy (to z čoho sme importovali pred chvíľou):

```
def _copy_immutable(x):
    return x
```

n-tice -> nemeniteľné zoznamy

```
a = (0, 2, 3)
a[0] = 1    # Error
```

Ale, toto je možné urobiť:

```
a = ([0], 2, 3)
a[0][0] = 1
```

Stackoverflow je váš kamarát / domov / rodina

Veľké množstvo otázok je spojených s meniteľnými objektmi:

- Ako chápať meniteľný parameter do funkcie
- Premenné a meniteľné objekty tu a tu
- "In-place" operácie, kde 2 premenné ukazujú na rovnaký objekt ale pozor na nemeniteľné objekty
- copy, deepcopy, priradenie
- Problém pri duplikácií zoznamov "vynásobením" (alebo tu)
- Pozor na `a = b = []`! `a`, `b` sú ten istý zoznam