Iterácia a zoznamy

Podobná prednáška

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

`for` cyklus

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

```
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.

Cvičenie 1

Máte zoznam:

```
["banana", "apple", False,
 "apple", "banana", True,
 "banana", "pig", None,
 1, "banana", None, True]
```

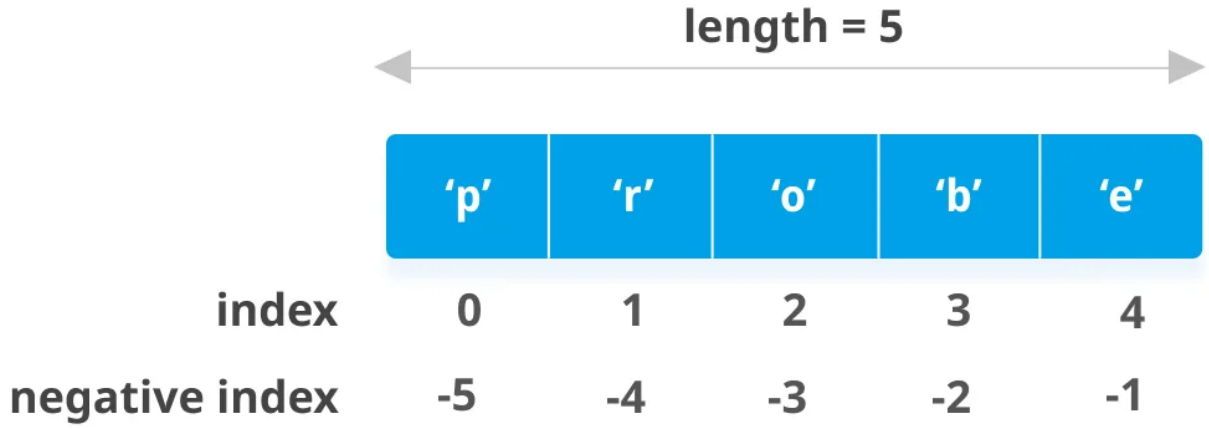
Pomocou `for` cyklu spočítajte:

- Počet banánov
- Počet jabĺk
- Počet ovocia
- Počet datových typov typu `bool`

Indexovanie

```
zoznam = ['p','r','o','b','e']
nulty_prvok = zoznam[0]
prvy_prvok = zoznam[1]

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



Obr. 1: Indexovanie

Pokročilé indexovanie -> slicing

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

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

Je možné vynechať hodnoty:

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

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

- Viac poznania

Cvičenie 2

- Naprogramujte funkciu `middle`, ktorá vezme zoznam a vráti nový zoznam, v ktorom chýba prvá a posledná hodnota, napr:

```
middle(['a', 'b', 'c', 'd'])  # -> ['b', 'c']
```

Návod k ďalšiemu cvičeniu

Metóda `str.split()` rozdelí text zoznam slov.

Metóda `str.join()` zoznam spojí do reťazca.

Príklad použitia:

```
text = "a,b,c"
abc = text.split(',')
s_medzerou = " ".join(abc)
```

Poznámka: Ak vynecháte argument v metóde `str.split`, tak reťazec sa rozdelí podla *whitespace* – medzier, nových riadkov a tabulátorov. Toto je veľmi užitočné, hlavne pre prípady keď máte viac medzier a nechcete rozdelovať za každou jednou:

```
"   1   2   3   ".split() # -> ['1', '2', '3']
```

Cvičenie 3

Rýchla referencia na slicing

Nasledujúca kombinácia slov nedáva zmysel:

```
to a steal TV ideas can from insult one your person
intelligence is but plagiarism nothing to rubs steal
it from in many like is a research computer
```

Ale ak budete čítať každé druhé slovo, tak dostanete dve zmysluplné vety. Naprogramujte program, ktorý vám slová usporiada v správnom poradí.

(Zdroj viet)

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
```

Cvičenie 4

- Naprogramujte funkciu `is_sorted`, ktorá vráti `True` pokiaľ je zoznam zoradený od najmenšej do najväčšej hodnoty, inak vráti `False`.

```
is_sorted([2, 1]) # -> False
is_sorted([]) # -> True
is_sorted([0]) # -> True
is_sorted([1.21, 3.14, 5]) # -> True
```

Cvičenie 5

Pomocou `for`-cyklu, a funkcií `range` a `len` vypíšte súčet prvkov nasledujúcich dvoch zoznamov:

```
from math import pi

lst1 = [8, "kuku", pi, ['a', 'b'], ('A', 'B')]
lst2 = [2, "rica", -1, ['c', 'd'], ('C', 'D')]
```

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!"
```

```
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(vysledok) # -> None, zoznam sa zmenil

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

Cvičenie 6

- Znovu naprogramujte funkciu `middle`, ale teraz namiesto toho, aby ste vrátili z funkcie nový zoznam, zoznam zmeňte, a vráťte `None`:

```
lst = [1, 2, 3]
middle_mutating(lst) # -> None
print(lst) # [2]
```

Návod:

Metóda `list.pop` može brať jeden argumnent, ktorý špecifikuje index, z ktorého prvok vyhodíte:

```
a = [1, 2, 3, 4, 5]
a.pop() # -> 5
a.pop(1) # -> 2
```

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

Z predchádzajúceho slidu je vidieť, že občas potrebujeme hlbokú (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?

Kópie nemeniteľných objektov

Prečo tiež nehovoríme o rozdiele v hlbokej a obyčajnej kópii u nemeniteľných objektoch?

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

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

Ak máte dva objekty, ktoré majú rovnakú hodnotu a nedajú sa zmeniť, tak je vám jedno či sú na rovnakom mieste v pamäti počítača – výsledky vašich programov to neovplyvní.

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

Cvičenie

Celulárny automat (zadanie v ISu v stud. mat.)

Cvičenie

Advent of Code (AoC) je veľmi pekná stránka, kde každý rok pred Vianocami nájdete kvalitne navrhnuté programovacie úlohy so stupňujúcou sa zložitosťou.

Vyriešte príklad 4 z roku 2019: Day 4.

Cvičenie na doma

Naprogramujte si vlastnú verziu `deepcopy`. Predpokladajte, že jediný meniteľný typ v Pythone je `list` (zoznam). Pre riešenie budete pravdepobne chciet využit **rekurziu** – ak vo vašom riešení nie je rekurzívne volanie, tak veľmi pravdepodobne tam máte chybu.

- Než začnete písať nejaký kód premyslite si čo chcete dosiahnuť – ako sa líši hlboká kópia od normálnej, a ako vašu funkciu budete testovať – ako poznáte, že ste získali naozaj hlbokú kópiu a nie obyčajnú.

Cvičenie je relatívne ťažké (aj keď samotný kód sa dá napísať na menej ako 10 riadkov), ale keď úlohu vyriešite samostatne budete sa cítiť veľmi spokojne.