Programování 1: Seznamy

Petr Čermák
cermak@mag.mff.cuni.cz

Katedra Fyziky Kondenzovaných Látek
MFF UK Praha

2020
forked from https://gitlab.kam.mff.cuni.cz/mj/prm1

Teoretický úvod

Python má několik druhů polí

- `list` (seznam)
  - `a = [1,2,3]`
  - různé datové typy
  - dají se měnit (*mutable*)

Teoretický úvod

Python má několik druhů polí

- `list` (seznam)
  - `a = [1,2,3]`
  - různé datové typy
  - dají se měnit (*mutable*)
- `tuple` (n-tice)
  - `a = 1,2,3` nebo `a = (1,2,3)`
  - různé datové typy
  - nedají se měnit (*immutable*)!

Teoretický úvod

Python má několik druhů polí

- `list` (seznam)
  - `a = [1,2,3]`
  - různé datové typy
  - dají se měnit (*mutable*)
- `tuple` (n-tice)
  - `a = 1,2,3` nebo `a = (1,2,3)`
  - různé datové typy
  - nedají se měnit (*immutable*)!
- `array`
  - `a = array(1,2,3)`
  - je potřeba externí knihovna (`array` nebo `numpy`)
  - pouze stejné datové typy
  - mnohem víc možností (maticové operace)

Generátory, ukazatele

Generátor je

- funkce, která umí postupně dávat k dispozici hodnoty
- později se je naučíme psát
- jsou vyhodnoceny až při průchodu cyklem

Generátory, ukazatele

Generátor je

- funkce, která umí postupně dávat k dispozici hodnoty
- později se je naučíme psát
- jsou vyhodnoceny až při průchodu cyklem

Pointery v pythonu

- nejsou
- *mutable* objekty se dají měnit - odkazujeme na ně referencí
- má to své následky, jak uvidíme později

Seznamy a jejich indexování

```
>>> cisla = [11, 22, 33, 44, 55]
```

Indexuje se od nuly!

```
>>> cisla[0]
11

>>> cisla[2]
33

>>> cisla[-1]
55

>>> len(cisla)
5
```

Vnořené seznamy

```
>>> matice = [[11, 12, 13], [21, 22, 23]]

>>> matice[0]
[11, 12, 13]
>>> matice[0][2]
13
```

Na práci s maticemi je lepší použít numpy.array

Cyklus for

```
>>> slova = ["mňau", "haf", "kvák"]
>>> for x in slova:
...     print(x)
mňau
haf
kvák

>>> for x in range(1, 5):
...     print(x)
1
2
3
4

>>> for i in range(len(x)):
...     print(x[i])
```

Operace se seznamy

```
>>> slova = ["mňau", "haf", "kvák"]
>>> slova.append("íá")
>>> slova
['mňau', 'haf', 'kvák', 'íá']

>>> [1, 2] + [3, 4, 5]
[1, 2, 3, 4, 5]

>>> [0] * 5
[0, 0, 0, 0, 0]
```

Řetězec se také chová trochu jako seznam

```
>>> slovo = "vrak"
>>> len(slovo)
4

>>> slovo[1]
'r'

>>> slovo[0] = 'm'
>>>
TypeError: 'str' object does not support item
assignment
```

Řetězec je immutable (jako n-tice tuple)!

Chyták: proměnné na seznamy jen odkazují!

```
>>> x = [1, 2, 3]
>>> y = x
>>> x[2] = 9
>>> y
[1, 2, 9]

>>> m = [[0] * 4] * 3
>>> m
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]

>>> m[0][0] = 1
>>> m
[[1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0]]
```

Vypsání čísel pozpátku

```
#!/usr/bin/env python3
# Načítá čísla ze vstupu ukončená -1,
# vypíše je pozpátku

seznam = []

while True:
    x = int(input())
    if x == -1:
        break
    seznam.append(x)

for i in range(len(seznam)):
    print(seznam[-i-1])
```

Vypsání čísel pozpátku

```
1  #!/usr/bin/env python3
2  # Alternativní řešení
3
4  seznam = [1,2,3]
5
6  #použití slicing
7  for i in seznam[::-1]:
8      print(i)
9
10 #použití funkce reverse
11 seznam.reverse()
12 for i in seznam:
13     print(i)
```

Něco navíc o generátorech

```
>>> range(5)
range(0, 5)

>>> list(range(5))
[0, 1, 2, 3, 4]

>>> list(range(2, 20, 3))
[2, 5, 8, 11, 14, 17]

>>> list(range(10, 0))
[]

>>> list(range(10, 0, -1))
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Úkoly na hodinu

Uložte do seznamu prvních N Fibonacciho čísel.
Úkol v Recodexu: najděte prvočísla mezi 1 a N a celé to obalte do funkce (detaily o funkcích později)

```
1 # ulož do souboru reseni.py
2
3 def prvocisla(N):
4     out = []
5     # udělej něco
6
7     return out
```