

Viac k zoznamom a cyklom

Najlepší operátor v Pythone -> in

```
lst = [1,2,3,4]
```

```
1 in lst # True
```

```
8 in lst # False
```

```
adage = """Wirth's law:  
Software gets slower more quickly  
than hardware gets faster."""
```

```
"h" in adage # True
```

```
"Wirth" in adage # True
```

```
"Bill" in adage # False
```

Poznámka: Toto nie je to isté in ako v syntaxe pre for-cyklus.

Alternatíva for -> while

```
while true_or_false:  
    do_something
```

Kým je podmienka splnená (True) tak vykonávajú nasledujúce príkazy.

Napr.:

```
i = 3  
while i > 0:  
    print(i)  
    i -= 1
```

má rovnaký efekt ako:

```
for i in range(3, 0, -1):  
    print(i)
```

Nikdy nekončiaci program

```
while True:  
    print("HELP!")
```

Ctrl + C je váš kamarát :-)

Skorý výstup z cyklu break

```
i = 3
while True:
    print(i)
    i -= 1
    if i == 0:
        break
```

- ▶ break funguje rovnako aj s for.

Užitočné 'built-in' funkcie so zoznamami

Všetky built-in funkcie v Pythone

▶ min, max

```
lst = [3217, 12, 1]  
min(lst) # 1  
max(lst) # 3217
```

Užitočné 'built-in' funkcie so zoznamami

▶ all, any

```
is_even = []  
for i in [2, 4, 6, 8, 15]:  
    is_even.append(i % 2 == 0)  
  
print(all(is_even)) # False;    15 % 2 -> 1  
print(any(is_even)) # True
```

Užitočné 'built-in' funkcie so zoznamami

- ▶ `list` - konverzia z iných iterovateľných

```
text = "A B C"  
characters = list(text) # ['A', ' ', 'B', ' ', 'C']
```

```
r = range(0, 4)  
lst = list(r) # [1, 2, 3]
```


Užitočné 'built-in' funkcie so zoznamami

▶ reversed

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

for i in reversed(lst):
    print(i)
```

▶ reversed nevráti zoznam, ale môžete zoznam vytvoriť:

```
lst = [1, 2, 3]
type(reversed(lst)) # list_reverseiterator
rev_lst = list(reversed(lst))
```

reversed

- ▶ reverseiterator môžete prechádzať len 1-krát!

```
a = [1,2,3]
rev_a = reversed(a)
for i in rev_a:
    print(i) # 3 2 1

for i in rev_a:
    print(i) # ... nič sa nevypíše
```

- ▶ Funguje aj str a tuple:

```
s = "racecar"
s = "".join(reversed(s))
```

sorted

- ▶ vytvoří zoradený zoznam

```
lst = [2, 3, 1]
ascending = sorted(lst) # [1, 2, 3]
descending = sorted(lst, reverse=True) # [3, 2, 1]
```

sorted vs list.sort

Zoznamy majú navyše aj metódu `sort`:

```
a = [2, 1, 3]
a.sort() #
print(a)
```

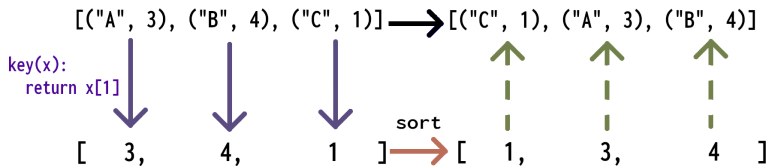
- ▶ Nový zoznam sa nevytvorí! Operácia je *in-place* -> zoznam sa zmení, vráti sa `None`
- ▶ *In-place* operácie môžu byť rýchlejšie.
- ▶ Tiež, pokiaľ máte zoznam, ktorý je veľmi dlhý (napr. 80 % vašej RAM), tak ho fakt nechcete duplikovať.
- ▶ Analogicky, existuje metóda `list.reverse`.

```
sorted(list, key=fun)
```

- ▶ Je možné špecifikovať funkciu, ktorá mapuje prvky na to čo sa má zoradiť:

```
salaries = [  
    ['Adam', 1000],  
    ['Bill', 10**6],  
    ['Bob', 4.5]  
]  
  
def get_salary(pair):  
    return pair[1]  
  
sorted(salaries, key=get_salary)
```

```
sorted(list, key=fun)
```



Obr. 1: sort s pomocou key

Čo list dokáže?

- ▶ append
- ▶ pop
- ▶ extend
- ▶ count
- ▶ remove
- ▶ clear
- ▶ insert
- ▶ index
- ▶ copy

```
3 * [1, 2] # -> [1, 2, 1, 2, 1, 2]
[1, 2] + [3, 4] # -> [1, 2, 3, 4]
```

Čo dokáže str?

Zoznam metód z dokumentácie Pythonu.

Moje obľúbené:

- ▶ replace
- ▶ format
- ▶ strip, lstrip, rstrip
- ▶ split a splitlines
- ▶ join
- ▶ find

```
3 * "A" + " " + "HELP!"    # -> "AAA HELP!"
```