Lekce 7.

Operátor **in**

- operátor **in** se používá k ověření, zda je konkrétní hodnota nebo objekt přítomný v sekvenci nebo sbírce, jako je řetězec, seznam, tuple nebo slovník.

```
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
```

Alternativa pro **for** - **while**

- Operátor while vykonává příkazy dokud není splněna podmínka uvnitř cyklu

```
while true_or_false:
    do_something
```

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

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

Nikdy nekončící program

- Lze ukončit pomocí Ctrl + C

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

Konec cyklu pomocí **break**

- break funguje is pro for cyklus

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

Užitečné ‘built-in’¨ funkce v Pythonu

- min(), max()

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

Užitečné ‘built-in’¨ funkce v Pythonu

- 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
```

Užitečné ‘built-in’¨ funkce v Pythonu

- list() konverze z jiných iterovatelných

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

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

Užitečné ‘built-in’¨ funkce v Pythonu

- reversed()

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

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

- reversed() nevrátí list, ale můžeme si ho vytvořit

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

Užitečné ‘built-in’¨ funkce v Pythonu

- reversed()

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

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

- reversed() nevrátí list, ale můžeme si ho vytvořit

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

Užitečné ‘built-in’¨ funkce v Pythonu

- reversediterátor může předcházet jen jedenkrát

```
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
```

- reversed() lze použít i na string a tuple

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

Užitečné ‘built-in’¨ funkce v Pythonu

- sorted() vytvoří seřazený seznam

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

Užitečné ‘built-in’¨ funkce v Pythonu

- seznam má i metodu sort

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

- nevytvoří se nový seznam, operace je in-place -> seznam se změní, vrátí se None
- In-place operace můžou být rychlejší
- Pokud máte hodně dlouhý seznam, např. 80 procent RAM, nechcete list duplikovat
- Analogicky exituje metoda list.reverse

Užitečné ‘built-in’¨ funkce v Pythonu

- Je možné specifikovat funkci, která mapuje prvky na to, co se má seřadit

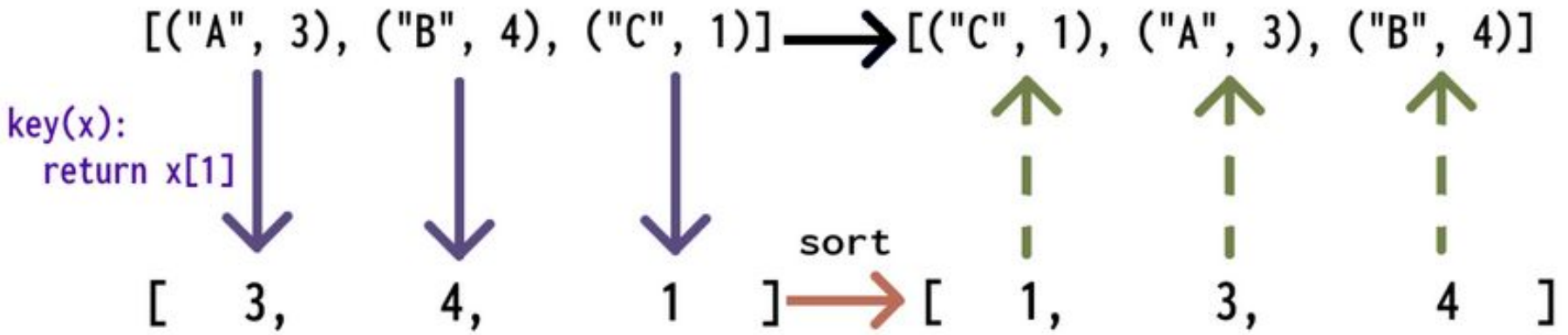
```
salaries = [
      ['Adam', 1000],
      ['Bill', 10**6],
      ['Bob', 4.5]
    ]

def get_salary(pair):
    return pair[1]

sorted(salaries, key=get_salary)
```

Užitečné ‘built-in’¨ funkce v Pythonu

- Je možné specifikovat funkci, která mapuje prvky na to, co se má seřadit



Co dokáže list

- 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]
```

Co dokáže string

- Replace
- Format
- Strip, lstrip, rstrip
- Split a splitlines
- Join
- Find

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

Cvičení 1.

- Napište funkci, která vezme jako argument string a zkontroluje, zda-li obsahuje samohlásku (a,e,i,o,u,y)

Cvičení 2.

- Napište funkci ***count_digits()***, využívající while smyčky, která spočítá kolikaciferné je dané číslo.

Cvičení 3.

- Napište funkci ***faktorial_w()***, využívající while smyčky, která spočte faktoriál zadaného čísla.

Cvičení 4.

- Napište funkci ***is_prime()***, využívající while smyčky, která zjistí, je-li dané číslo prvočíslem.

1. Použitím while smyčky
2. Použitím for smyčky

Cvičení 5.

- Napište funkci ***is_palindrome()***, která zkontroluje zda-li je zadaný string palindrom, čili zdali je stejný, pokud jej přečteme pozpátku.

1. Za použití while smyčky
2. Za použití for smyčky
3. Pomocí rekurzivní funkce