PYTHON

SEZNÁMENÍ S PROGRAMOVACÍM JAZYKEM

Cykly for a while
Seznamy a slovníky
Funkce

Lukáš Kotek

CYKLUS WHILE

- **Cyklus while**
  - Používá se, když neznáme předem počet opakování
  - Nejjednodušší cyklus v Pythonu
- **Klíčová slova (platí i pro cyklus for)**
  - **`continue`** – přeruší aktuální vykonávání cyklu
    - pouze jeden průběh → pokračuje se dalším
  - **`break`** – ukončí cyklus definitivně
    - i když je podmínka na začátku stále splněna

```
while VYRAZ:
    # Kod, který se ma opakovane vykonavat, dokud je
    # VYRAZ pravdivy – vraci hodnotu True
    pass    # Nedela nic, blok nesmi byt prazdny
```

- **Cyklus while v kostce? „Prováděj pokyny, dokud platí…"**

https://www.sallyx.org/sally/python/python5.php#while

SEZNAMY A SLOVNÍKY

- **Seznamy a n-tice**
  - **Seznam** (list) → editovatelný typ
  - **N-tice** (tuple) → sada konstant
  - Mohou obsahovat více různých datových typů
  - Lze vzájemně vnořovat

```
seznam = [1, "Lukas", 7.9]
ntice = (2, 4, 6, 8)
vnoreny = [1, [8, 3.14], True]

seznam.append(7)      # Pripoji prvek 7 na konec seznamu

# Co vrati zapisy: seznam[2] … vnoreny[1][1] ..?
```

Slovníky (dict)

```
slovnik = {"a": 7, "b": 54, "c":"Nohy"}
slovnik.update({"d":42})     # Prida zaznam do slovniku
```

CYKLUS FOR

- **Specifika cyklu for v Pythonu**
  - Používá se nad iterovatelnými objekty, což je:
    - Cokoliv, co lze **postupně procházet**
    - Cokoliv, co má **pevný počet** prvků
    - Seznamy, slovníky, řetězce, funkce vracející n-tice apod.

```
for i in [prvek1, prvek2, ...]:
    # Do promenne i se postupne priradi jednotlivé prvky
    # pass    # Nedela nic, blok nesmi byt prazdny

range(5)       # vrati prvky 0,1,2,3,4
range(2,8,2)   # vrati prvky 2,4,6
range(5,0,-1)  # vrati prvky 5,4,3,2,1
```

Chová se obdobně jako `foreach` **v mnoha jiných jazycích**

https://www.sallyx.org/sally/python/python5.php#for

SYNTAKTICKÉ TRIKY

```
# Zkrácený zápis podmínky zkombinovaný s přiřazením
prom = 1 if True else 0
print(prom)

# Zkrácený zápis cyklu (vytvoření seznamu)
print([x**2 for x in range(5)])

# Zkrácený zápis cyklu (vytvoření seznamu) s podmínkou
print([x**2 for x in range(5) if x != 3])

# Dva for cykly (vytvoření seznamu)
print([i*j for i in range(4) for j in range(4)])

# Operator pro rozbalovani prvku seznamu
seznam = [1, 2, 3, 7]
*seznam
```

http://www.sallyx.org/sally/python/syntakticky-cukr.php

FUNKCE A JEJICH DEFINICE

- **Funkce v Pythonu**
  - Není syntaktický rozdíl v zápisu funkce a „procedury“
  - Funkce mohou mít větší množství argumentů
    - Argumentem může být cokoliv
    - Argument může mít defaultní hodnotu

```
def nazev_funkce(arg=2):
    # Kod, který se mavykonat v tele funkce
    pass       # Nedela nic, blok nesmi byt prazdny
    return arg**2  # Vraceni hodnoty

nazev_funkce(7)  # Volani funkce, které vrati 49
nazev_funkce()   # Vrati 4, pouzije se defaultni hodnota
```

https://docs.python.org/2/tutorial/controlflow.html#defining-functions

ARGUMENTY PŘEDANÉ PROGRAMU

- **Jak parsovat argumenty**
  - Nutné použít modul sys, popř. další specializovaný (např. **`argparse`**)
  - Argumenty indexujeme klasicky od prvku s indexem 0
- **Simulace funkce „main"**
  - Použití bloku `if __name__ == "__main__":`

```
import sys

if __name__ == "__main__":
    # Telo se vykona pouze tehdy, když je kod primo
    # spusten a není importovan do jineho programu
    print(sys.argv[0])  # Nazev programu
    print(sys.argv[1])  # Prvni predany argument

# ---------------- Spusteni programu ----------------- #
python nazev_programu.py 02
# Vypise:
# nazev_programu.py
# 02
```

Otázky?