# IB111

# Základy programování

František Lachman
lachmanfrantisek@mail.muni.cz

cvičení 8

9. listopad 2017

# Osnova

- kontrolní otázky
- seznam, zásobník, fronta
- slovník
- množina

# Kontrolní otázky

- Co znamenají pojmy abstraktní datový typ a konkrétní datová struktura?
- Jaké operace podporují datové typy fronta a zásobník? Jaký je mezi nimi rozdíl? K čemu je využíváme?
- K čemu slouží datový typ slovník? Jak používáme slovník v Pythonu?

# Kontrolní otázky

- Jaký je rozdíl mezi seznamem a n-ticí?
- Co to je postfixová notace? Za využití které datové struktury ji vyhodnocujeme?
- Jakým způsobem můžeme reprezentovat dvourozměrnou mřížku (například šachovnici)?
- Jak můžeme vypsat všechny unikátní prvky v seznamu?
- Jaký je rozdíl mezi `data[x][y]` a `data[x,y]`?

# Seznam

- vytvoření:

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

- přidání prvku na konec seznamu:

```
my_list.append(10)
print(my_list) # [1, 3, 2, 10]
```

- Přidání prvku na kokrétní index

```
my_list.insert(1, 10) # na index 1
print(my_list) # [1, 10, 3, 2, 10]
```

# Seznam

- odebrání prvního výskytu prvku:

```
my_list.remove(10)
print(my_list) # [1, 3, 2, 10]
```

- odebrání prvku na indexu:

```
del my_list[1] # odebrani prvku na indexu 1
print(my_list) # [1, 2, 10]
```

# Zásobník LI-FO

- vytvoření zásobníku:

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

- přidání prvku do zásobníku:

```
stack.append(4)
print(stack) # [1, 2, 3, 4]
```

# Zásobník LI-FO

- odebrání prvku ze zásobníku

```
top = stack.pop()
print(stack, top) # [1, 2, 3] 4

top = stack.pop()
print(stack, top) # [1, 2] 3
```

# Fronta FI-FO

```
from collections import deque
```

- Vytvoření fronty:

```
queue = deque(["Petr", "Zdenek", "Filip"])
```

- přidání nového prvku (na konec fronty):

```
queue.append("Kuba")
print(queue)
# deque(["Petr", "Zdenek", "Filip", "Kuba"])
queue.append("Roman")
print(queue)
# deque(["Petr", "Zdenek", "Filip", "Kuba", "Roman"])
```

# Fronta FI-FO

- odebrání prvku z fronty:

```
print(queue)
# deque(["Petr", "Zdenek", "Filip", "Kuba", "Roman"])

student = queue.popleft()
print(queue, student)
# deque(["Zdenek", "Filip", "Kuba", "Roman"]) "Petr"

student = queue.popleft()
print(queue, student)
# deque(["Filip", "Kuba", "Roman"]) "Zdenek"
```

# Příklady

## 9.2.1. Má mě rád, nemá mě rád

Vytvořte funkci `game(n)` simulující otrhávání okvětních lístků.

Nejprve si vygenerujeme kytici lineárně uspořádaných `n` květin, každou o `1 až 4` okvětních lístcích. Poté vezmeme první květinu a utrhneme z ní právě jeden lístek a zařadíme ji nakonec. Postup opakujeme dokud nejsou všechny květiny otrhané.

# Slovník

- ilustrace ve sbírce
- Vytvoření:

```
points = {"Jack":66, "Peter":0, "Denis":0}
```

- Přístup:

```
print(points["Jack"]) # 66

points["Tom"] = 60
print(points)
# {"Jack":66, "Peter":0, "Denis":0, "Tom":60}
```

# Slovník

- Změna hodnoty:

```
print(points)
# {"Jack":66, "Peter":0, "Denis":0, "Tom":60}

points["Peter"] += 60 # zmena hodnoty pod nejakym klicem
print(points) # {"Jack":66, "Peter":60, "Denis":0, "Tom":6
```

- Smazání záznamu:

```
del points["Denis"] # smazani zaznamu s klicem "Denis"
# body: {"Jack":66, "Peter":60, "Tom":60}
```

# Slovník

- iterace přes klíče

```
for name in points:
    print(name)
```

- iterace přes hodnoty:

```
for value in points.values():
    print(value)
```

- iterace přes klíče a hodnoty současně:

```
for key, value in points.items():
    print(name, ':', value)
```

# Slovník

- Test příslušnosti:

```
if "Sam" in points:
    print("Sam's record present.")
else:
    print("No record for Sam")
```

- Další užitečné metody:
  - `get(key, default=None)`
  - `setdefault(key, default=None)`
  - `update(dict2)`

# Příklady na slovník

## 9.3.1. Morseova abeceda

Napište funkce `encode` a `decode` pro převod řetězce do a z Morseovy abecedy.

# Příklady na slovník

## 9.3.2. Frekvenční analýza písmen

Napište funkci `freq_analysis(text)`, která spočítá výskyt jednotlivých písmen (znaků) ve vstupním textu a následně tento seznam vypíše setříděný sestupně podle počtu výskytů.

# Množina

- Vytoření prázdné množiny:

```
my_set = set()
```

- přidání prvku:

```
my_set.add(5)
print(my_set) # set([5])

my_set.add(6)
my_set.add(6)
print(my_set) # set([5, 6])
```

# Množina

- přidání kolekce prvků:

```
my_set.update([1, 3, 10, 15])
print(my_set) # set([1, 3, 5, 6, 10, 15])
```

- kardinalita

```
cardinality = len(mnozina) # zjisteni velikosti mnoziny
print(cardinality) # 5
```

# Množina

- výskyt prvku:

```
if 10 in my_set:
    print('10 is member of my_set')
```

- iterace přes prvky:

```
for x in my_set:
    print(x, x * x)
```

# Operátory na množinách

```
a, b = set([1, 3, 5, 7]), set([2, 3, 4, 5])
```

- sjednocení:

```
print(a | b) # set([1, 2, 3, 4, 5, 7])
```

- průnik množin

```
print(a & b) # set([3, 5])
```

- množinový rozdíl:

```
print(a - b) # set([1, 7])
```

# Příklady na množiny

## 9.4.2. Kontrola řádku Sudoku

Napište funkci `sudoku_line_check(line)`, která zkontroluje, zda předaný seznam reprezentuje správný řádek vyplněného Sudoku, tj. obsahuje pouze čísla `1` až `9` a každé z nich právě jedenkrát.

```
print(sudoku_line_check([1, 2, 8, 9, 3, 5, 6, 7, 4]))
# True
print(sudoku_line_check([1, 2, 8, 9, 3, 5, 7, 4]))
# False
print(sudoku_line_check([1, 1, 2, 8, 9, 3, 5, 7, 4]))
# False
print(sudoku_line_check([0, 1, 2, 3, 4, 5, 6, 7, 8]))
# False
```

# Závěr

- seznam,zásobník, fronta
- slovník
- množina