IB111

Základy programování

František Lachman lachmanfrantisek@mail.muni.cz

cvičení 8

6. listopad 2018

Osnova

- kontrolní otázky
- datové struktury
- vnitrosemestrálka
- třetí domácí úloha

Docházka

Kontrolní otázky

```
import random
random.randint(1,6)
```

Jaký je rozdíl mezi seznamem a n-ticí?

- 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]
```

Jak odebereme konkrétní prvek ze seznamu?

(známe jeho hodnotu)

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

```
import random
random.randint(1,6)
```

Jak odebereme konkrétní prvek ze seznamu?

(známe jeho index)

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

```
import random
random.randint(1,6)
```

Co to je postfixová notace? Za využití které datové struktury ji vyhodnocujeme?

```
import random
random.randint(1,6)
```

Proč nazýváme zásobník

LI-FO?

```
import random
random.randint(1,6)
```

Jak vytvoříme v Pythonu zásobník?

- vytvoření zásobníku:

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

```
import random
random.randint(1,6)
```

Jak přidáme do zásobníku nový prvek?

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

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

```
import random
random.randint(1,6)
```

Jak odebereme prvek ze zásobníku?

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

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

```
import random
random.randint(1,6)
```

Jakým způsobem můžeme reprezentovat dvourozměrnou mřížku (například šachovnici)?

```
import random
random.randint(1,6)
```

Jak můžeme vypsat všechny unikátní prvky v seznamu?

```
import random
random.randint(1,6)
```

Jaký je rozdíl mezi `data[x][y]` a `data[x,y]`?

```
import random
random.randint(1,6)
```

Co je to `fronta`, jak ji nazýváme a jak ji v Pythonu vytvoříme?

Fronta FI-FO

```
from collections import deque
```

- Vytvoření fronty:

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

```
import random
random.randint(1,6)
```

Jak přidáme nový prvek do fronty?

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

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

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

```
import random
random.randint(1,6)
```

Jak odebereme prvek z fronty?

```
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 (`"leaf"`). 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
```

První vnitrosemestrálka

Příklad 1

```
def numbers(n):
    if n <= 0:
        return None
    last = 1
    print(last)
    for _ in range(n-1):
        if last%2 == 0:
            last = last**2 + last + 3
        else:
            last -= 1
        print(last)
```

První vnitrosemestrálka

Příklad 2

```
def table(size):
    if size <= 0:
        return None
    for i in range(size):
        for j in range(size):
            number = (size - i - j - 1) % size + 1
            print(number, end=" ")
        print()
```

První vnitrosemestrálka

Příklad 3

```
def eleven(n):
    digit_sum = 0
    i = 0
    while n > 0:
        digit_sum += (n % 10) * (-1) ** i
        i += 1
        n //= 10
    return digit_sum % 11 == 0
```

```
def eleven(n):
    digits = [int(d) for d in str(n)]

    result = 0
    for i in digits[0::2]:
        result += i
    for i in digits[1::2]:
        result -= i

    return result % 11 == 0
```

```
def eleven(n):
    digits = [int(d) for d in str(n)]
    result = sum(digits[0::2]) - sum(digits[1::2])
    return result % 11 == 0
```

První vnitrosemestrálka

Příklad 4

```
def same_number_of_vowels(text1, text2):
    vowels = ["a", "e", "i", "o", "u", "y"]
    text1_lower = text1.lower()
    text2_lower = text2.lower()
    count1 = 0
    count2 = 0

    for vowel in vowels:
        count1 += text1_lower.count(vowel)
        count2 += text2_lower.count(vowel)

    return count1 == count2
```

```
def is_vowel(char):
    return char in ['a', 'e', 'i', 'o', 'u']


def count_vowels(text):
    count = 0
    for c in text:
        if is_vowel(c):
            count += 1
    return count


def same_number_of_vowels(text1, text2):
    return count_vowels(text1) == count_vowels(text2)
```

Domácí úkol 4

Závěr

- seznam
- zásobník
- fronta
- slovník
- množina
- vyhodnocení vnitrosemestrálky
- domácí úkol