IB111

Základy programování

František Lachman
lachmanfrantisek@mail.muni.cz

cvičení 8

8. listopad 2017

Osnova

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

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
```

První vnitrosemestrálka

Příklad 1

```
def numbers(n):
    element = 1
    for _ in range(n):
        print(element, end=" ")
        element = 3 * element - 1
```

První vnitrosemestrálka

Příklad 2

```
def table(size):
    for y in range(size):
        for x in range(size):
            current = x + y + 1
            print((current - 1) % size + 1, 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
```

```python
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
```

```python
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 text(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)

    if count1 > count2:
        return text1
    return text2
```

```
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 text(text1, text2):
    if count_vowels(text1) > count_vowels(text2):
        return text1
    return text2
```

První vnitrosemestrálka

Příklad 5

```
def triangle(size, width, base_angle, turtle):
    turtle.forward(size)
    turtle.right(180 - base_angle)
    turtle.forward(width)
    turtle.right(180 - base_angle)
    turtle.forward(size)
```

První vnitrosemestrálka

Příklad 5 - pokračování

```
def mill(size, width):
    turtle = Turtle()
    angle = math.degrees(math.asin((width / 2) / size))
    base_angle = (180 - 2 * angle) / 2

    turtle.left(angle)
    for i in range(4):
        triangle(size, width, base_angle, turtle)
        turtle.right((360 - 4 * 2 * angle) / 4)
```

Bonusový příklad

9.5.1. Vyhodnocení logického výrazu v postfixové notaci

Funkce vrátí `True`, pokud je předaný výrok pravdivý, jinak `False`. Výraz se skládá z následujících znaků:

```
'T': True
'F': False
'N': negace
'K': konjunkce
'A': disjunkce
'C': implikace
```

Bonusový příklad

```
def evaluate_logic_expression(expr):
    pass

print(evaluate_logic_expression("TFK"))
# False
print(evaluate_logic_expression("FNTFCNK"))
# True
```

Třetí domácí úkol

Padací piškvorky

Vytvořte program hrající hru Padající piškvorky proti uživateli (na plánu zadané velikosti). Tato variace piškvorek se hraje na dvourozměrném hracím plánu. Hra je podobná klasickým piškvorkám s tím rozdílem, že pokud jste na tahu, nevolíte konkrétní čtvereček, do kterého byste umístili svůj symbol, ale sloupec. Symbol v daném sloupci spadne dolů (nejvíce, co to jde). Vyhrává ten, kdo poskládá 4 své symboly v řadě, sloupci nebo diagonále.

Třetí domácí úkol - zadání I

- Funkce `show_state(state)`, která vypíše daný plán na standardní výstup.
- Plán je reprezentován seznamem seznamů stejné délky, které obsahují znaky `X` a `O`, nebo `_` (mezera pro neobsazené pole).

Třetí domácí úkol - zadání II

```
>>> state = [[' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', 'O'],
    [' ', ' ', ' ', ' ', 'X'],
    [' ', 'O', ' ', ' ', 'X']]
>>> show_state(state)

        O
        X
  O     X
- - - - -
0 1 2 3 4
```

Třetí domácí úkol - zadání III

- Funkce `strategy(state, symbol)`, která pro daný plán a symbol vrátí pozici (sloupec) optimálního tahu.

```
>>> state = [[' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', 'X', ' ']]
>>> strategy(state, 'O')
2
```

- Za "chytrá" řešení budou bonuseové body.

Třetí domácí úkol - zadání III

- Funkce `tictactoe(rows, cols, human_starts=True)`, která umožňuje hrát hru padajících piškvorek na plánu o daném počtu řádků a sloupců.
- Můžete předpokládat, že zadaná velikost plánu je rozumná (alespoň `3` a méně než `50` sloupců a řádků).
- Parametr `human_starts` určuje, zda začíná hráč nebo počítač. Výpis průběhu hry by měl vypadat stejně, jako v následujících příkladech.

Třetí domácí úkol - zadání IV

- Funkce kontroluje, zda jsou tahy zadané hráčem platné a pokud nejsou, vyzve ho k novému zadání. Pro hru počítače volejte výše uvedené funkce `show_state(state)` a `strategy(state, symbol)`

Třetí domácí úkol - příklad I

```
Na tahu je hrac
Do jakeho sloupce chces hrat? (do 0 do 9)? 5




          X
- - - - - - - - - -
0 1 2 3 4 5 6 7 8 9
```

Třetí domácí úkol - příklad II

```
Na tahu je pocitac
Pocitac hraje do sloupce cislo 9




          X       O
- - - - - - - - - -
0 1 2 3 4 5 6 7 8 9
```

Třetí domácí úkol - příklad III

```
Na tahu je hrac
Do jakeho sloupce chces hrat? (do 0 do 9)? 6




          X X     O
- - - - - - - - - -
0 1 2 3 4 5 6 7 8 9
```

Třetí domácí úkol - příklad IV

```
Na tahu je pocitac
Pocitac hraje do sloupce cislo 5




          O
          X X       O
- - - - - - - - - -
0 1 2 3 4 5 6 7 8 9
```

Třetí domácí úkol - příklad V

```
Na tahu je hrac
Do jakeho sloupce chces hrat? (do 0 do 9)? 4




          O
        X X X     O
- - - - - - - - - -
0 1 2 3 4 5 6 7 8 9
```

Třetí domácí úkol - příklad VI

```
Na tahu je pocitac
Pocitac hraje do sloupce cislo 7




          O
        X X X O   O
- - - - - - - - - -
0 1 2 3 4 5 6 7 8 9
```

Třetí domácí úkol - příklad VII

```
Na tahu je hrac
Do jakeho sloupce chces hrat? (do 0 do 9)? 3




          O
      X X X X O   O
- - - - - - - - - -
0 1 2 3 4 5 6 7 8 9
Vyhral jsi!
```

Závěr

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