IB111 Úvod do programovaní skrze Python

Autor: Slavomír Krupa,
Text inšpirovaný: Valdemarom Švábenským

VNITRO

Table

```
def table(n):
    for i in range(n):
        for j in range(1, n + 1):
            print(j ** i, end=' ')
        print()
```

Same ascii

```
def print_same_ascii(text, list_char_order):
    for i in range(len(text)):
        if text[i] == chr(list_char_order[i]):
            print(text[i], end='')
```

Pyramid dice

```
def pyramid_dice(n):
    counts = [0, 0, 0, 0]
    for _ in range(n):
        counts[randint(0, 3)] += 1
    print(0 not in counts)
    min_count = min(counts)
    for i in range(len(counts)):
        if counts[i] == min_count:
            print(i + 1,end=' ')
```

Recursive max

```
def recursive_max(l):
    if len(l) > 1:
        maximum = recursive_max(l[1:])
        if maximum > l[0]:
            return maximum
        else:
            return l[0]
    else:
        return l[0]
```

Filter chars

```
def filter_chars(text, prohibited_chars):
    for char in prohibited_chars:
        text = text.replace(char, '')
    return text
```

OOP

OOP

- ★ Objektovo orientované programovanie je paradigma (myšlienkový vzor) programovania postavená na 3 základných princípoch
  - Abstraktný dátový typ
  - ~~Dedičnosť~~
  - ~~Polymorfizmus~~

Trieda

- Trieda abstraktný dátový typ (ADT) v OOP
  - Výrobná šablóna pre nové dátové štruktúry
  - “Výrobný vzor” pre tzv. objekty

```
class Square:
    def __init__(self, a):
        self.a = a
```

Trieda

Objekt

- Objekt - inštancia triedy, má svoj stav a správanie
  - Stav - uchovávané dáta, hodnoty atribútov
  - Správanie- funkcie; metódy patriace objektu

```
class Square:
    def __init__(self, a):
        self.a = a

s = Square(10)
```

Objekt

Objekt

- ★ Atribúty a metódy sú predpísané triedou
- ★ Triedu môžeme používať ako nový dátový typ (podobne ako int, string, bool, list..)

```
first = Square(10)
print(first.a)
other = Square(5)
print(other.a)
```

Prvý Objekt

Druhý Objekt

Vytváranie objektu

- Špeciálna metóda na vytvorenie objektu
  - Konštruktor - inicializátor
  - Jeho úlohou je nastaviť hodnoty atribútov
  - Vždy obsahuje jeden atribút `self`

```
class Square:
    def __init__(self, a):
        self.a = a
```

Vytváranie objektu

- Volanie názvu triedy `Square()` zavolá konštruktor (metódu `__init__`)
  - Parameter `self` sa používa "skryto"
    - nie je nutné ho písať pri volaní

```
class Square:
    def __init__(self, a):
        self.a = a
s = Square(10)
```

Vypisovanie objektu

★ Volanie `print` na objekte typu `Square()` zavolá metódu `__str__`

```
class Square:
    def __str__(self):
    return "Square: side {}".format(self.a)

s = Square(10)
print(s)
```

Atribút

- ★ Dátová zložka (premenná patriaca objektu)
- ★ Od lokálnych premenných odlíšená prefixom `self`

```
class Rectangle:
    def __init__(self, a, b):
        self.a = a
        self.b = b
```

Atribút

- Prístup k atribútom cez “bodkovú” notáciu
  - V rámci triedy `self.attribute`
  - Cez objekt `object.attribute`

```
class Square:
...
    def volume(self):
        return self.a ** 2
o = Square(10)
print(o.a)
```

Metóda

- ★ Funkcia ktorá ovplyvní objekt, na ktorom je definovaná
- ★ Má prístup k atribútom objektu cez `self`
- ★ Je definovaná nad triedou

```
class Square:
...
    def volume(self):
        return self.a ** 2
o = Square(10)
print(o.a)
```

Metóda

- Prístup k metódam cez “bodkovú” notáciu
  - V rámci triedy `self.method(params,...)`
  - Cez objekt `object.method(params,...)`

```
class Square:
...
    def __str__(self):
    return "Square: " + str(self.volume())
o = Square(10)
print(o.volume())
```

Metóda

```
class Square:
...
    def compare(self, square):
    if square.a > self.a:
        print("Other square is bigger")
    else:
        print("I am bigger square")

x = Square(10)
y = Square(11)
x.compare(y)
y.compare(x)
```

Úloha 1

- Vytvorte triedu `Rectangle` reprezentujúcu obdĺžnik a doplňte metódy na vypočítanie obsahu (`volume`) a obvodu(`perimeter`) a výpisu (__str__).
- `r = Rectangle(10,20)`
- `print(r.volume()) => 200`
- `print(r.perimeter()) => 60`
- `print(r) => “Rectangle a=10 b=20”`

Úloha 2

- Rozšírte triedu `Rectangle` o metódu `compare(self,rectangle)`, ktorá porovná obsah objektu `rectangle` s obsahom `self` a výpíše kto má väčší obsah.
- `r = Rectangle(10,20)`
- `s = Rectangle(15,15)`
- `r.compare(s) => “Other rectangle is bigger”`

DU

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash