IB113 Úvod do programování a algoritmizace

Cvičení 6

Jaromír Plhák

Domácí úkol 2 + řešení z minula

- Nějaké dotazy?

Úkol – opakování řetězců

- 5.3.1. Caesarova šifra
  - Napište funkci, která zašifruje text tak, že posune všechna jeho písmena v abecedě o n dopředu (cyklicky), můžete se inspirovat popisem Caesarovy šifry.

Seznam

- Strukturovaný datový typ pro uchovávání více položek v daném pořadí
  - Libovolný typ položek (ale radši nemíchat)
  - Libovolný počet položek (může růst za běhu programu)
- Definice: list_name = [item_1, item_2, . . .]

>>> empty_list = []

>>> pythons = ["Cleese", "Idle", "Jones", "Gilliam", "Chapman"]

- V jiných programovacích jazycích se používají pole (*array*)

Řetězec vs. seznam

- Řetězec se teoreticky dá uložit jako seznam, jehož položky jsou jednotlivé znaky
- Ale není to to samé! (seznam se dá měnit po znacích)

```
>>> string = "Python"
>>> lst = ['P', 'y', 't', 'h', 'o', 'n']
```

Jednotlivé položky seznamu

- Zpřístupnění položek: přes index jako u řetězca

>>> pythons = ["Cleese", "Idle", "Jones", "Gilliam", "Chapman"]

>>> pythons [3]

'Gilliam‘

- Změna položky: přes index (seznam je **měnitelný**)!

>>> pythons [4] = "Chapman (R.I.P.)"

>>> pythons

['Cleese ', 'Idle ', 'Jones ', 'Gilliam ', 'Chapman (R.I.P.) ']

Rozsah indexů

- Jako u řetězců:
  - Počáteční i koncový index
  - Jen počáteční / jen koncový index
  - Záporné indexy
  - Krok
- Pozor na neplatné indexy!

```
>>> pythons = ["Cleese", "Idle", "Jones", "Gilliam", "Chapman", "Palin"]
>>> pythons [1:4]
['Idle', 'Jones', 'Gilliam']
>>> pythons [1:]
['Idle', 'Jones', 'Gilliam', 'Chapman', 'Palin']
>>> pythons [:4]
['Cleese', 'Idle', 'Jones', 'Gilliam']
```

Iterace nad seznamem 1

- Jako u řetězců
- For cyklus přes prvky

```
for prime in [2, 3, 5, 7]:
    print (prime)
```

- Proměnná prime postupně nabývá hodnoty všech prvků seznamu (dle jejich pořadí)

Iterace nad seznamem 2

- Cyklus for přes indexy

```
lst = ["x", "y", "z"]
for i in range (len (lst)):
    print (lst [i])
```

Vložení hodnoty do seznamu

- values = []
- values.append(3)
- values.append(11)
- values.append(6)
- values[3] = 2 # IndexError: list assignment index out of range

Aritmetické operátory nad seznamem

- Jako u řetězců (+, *)

```
>>> aList = [2, 3, 5, 7]
>>> bList = [11 , 13, 17, 19]
>>> aList + bList
[2, 3, 5, 7, 11, 13, 17, 19]
>>> aList * 2
[2, 3, 5, 7, 2, 3, 5, 7]
```

Operátory in, not in

- Jako u řetězců
- Test na výskyt položky v seznamu

```
>>> 'Jackson' in pythons
False
>>> 5 * 8 in [30 , 40, 50]
True
>>> ['a'] in [['a'], ['a', 'b']]
True
>>> 258 not in []
True
```

Operátory is, is not

- Test na identitu objektov, nie na rovnosť prvkov

```
>>> x = [1, 2, 3]
>>> y = [1, 2, 3]
>>> x == y
True # Equal elements
>>> x is y
False # But different objects
```

Kopírovaní seznamu - vytvoření aliasu

```
>>> badguys1 = ["Joker", "Bane"]
>>> badguys2 = badguys1 # New reference
>>> print(badguys1)
['Joker', 'Bane']
>>> print(badguys2)
['Joker', 'Bane']
>>> badguys1 is badguys2
True
```

```
>>> badguys1[0] = "Scarecrow" # Beware !
>>> print (badguys1)
['Scarecrow', 'Bane']
>>> print(badguys2)
['Scarecrow', 'Bane']
>>> badguys1 is badguys2
True
```

Kopírovaní seznamu – plytká kopie

```
>>> badguys1 = ["Joker", "Bane"]
>>> badguys2 = badguys1[:] # Shallow copy
>>> print(badguys1)
['Joker', 'Bane']
>>> print(badguys2)
['Joker', 'Bane']
>>> badguys1 is badguys2
False
```

```
>>> badguys1[0] = "Scarecrow" # Beware !
>>> print(badguys1)
['Scarecrow', 'Bane']
>>> print(badguys2)
['Scarecrow ', 'Bane']
>>> badguys1 is badguys2
False
```

Úkoly

- 5.1.1. Součet, maximum a hledání
  - Napište funkce nad seznamem čísel, které zjistí:
    - součet všech čísel v seznamu,
    - nejvyšší číslo v seznamu,
    - zda se určitá hodnota vyskytuje v seznamu,
  - Ekvivalenty operací max, sum a in (ale s použitím pouze základních operací nad seznamy).
- 5.1.2. Součin nenulových čísel
  - Napište funkci, která vypočítá součin čísel v seznamu, ale ignoruje přitom případné nuly.
- 5.1.3. Modifikace vs. vytváření nového seznamu
  - Napište funkci double_all, která dostane na vstupu seznam čísel a každý jeho prvek vynásobí dvěma. Dále napište funkci create_doubled, která dostane na vstupu seznam čísel a vrátí nový seznam získaný ze vstupního tak, že každý prvek vynásobí dvěma. Na rozdíl od předchozí funkce však nemění předaný seznam.
- 5.1.4. Zploštění
  - Napište funkci, jejímž vstupem je seznam seznamů a výstupem je seznam, který obsahuje prvky všech jednotlivých seznamů.