IB111

Základy programování

František Lachman
lachmanfrantisek@mail.muni.cz

cvičení 5

18. říjen 2017

Osnova

- kontrolní otázky
- seznamy
- příklady na seznamy
- řetězce
- příklady na řetězce

Kontrolní otázky

- Jak vytvořit seznam obsahující čísla od `1` do `5`? (uveďte několik různých způsobů)
- Je u datové struktury seznam důležité pořadí prvků?
- Může v Pythonu seznam obsahovat položky různého typu?

Kontrolní otázky

- Jakým příkazem přidáme do seznamu nový prvek?
- Co znamená „indexování od nuly“?
- Co znamená zápis `alist[3:7]`?
- Proč není dobrý nápad dát proměnné obsahující seznam jméno `list`?

Kontrolní otázky

- Řetězec je v mnoha ohledech podobný jako „seznam znaků“. V čem se liší?
- Jaký je význam funkcí `chr` a `ord`?
- Jak zjistíme délku seznamu?
- Jak zjistíme poslední prvek seznamu?
  Zkuste najít 3 různé způsoby.

Seznamy

```
# vytvoreni seznamu s peti prvky
>>> values = [7, 2, 8, 2, 9]
>>>
>>>
# pridani noveho prvku na konec seznamu
>>> values.append(6)
>>> print(values)
[7, 2, 8, 2, 9, 6]
```

Seznamy

```
# pristup k prvkum
>>> print(values[2])
8
>>> print(values[-1])
6
```

```
# zmena hodnoty prvku
>>> values[0] = 15
>>> print(values)
[15, 2, 8, 2, 9, 6]
```

Funkce na seznamech

```
>>> values = [7, 2, 8, 2, 9]
>>> print(len(values))
5
>>> print(min(values))
2
>>> print(max(values))
9
>>> print(sum(values))
28
>>> print(8 in values)
True
```

Procházení prvky seznamu

```
values = [7, 2, 8, 2, 9]
for value in values:
    print(value, end=" ")
```

```
7 2 8 2 9
```

Odkazy a kopie

- proměnná na seznam "ukazuje", neobsahuje jej

```
values = []
values.append(3)
values.append(11)
values[0] = 2
a = values
a[0] = 4
a.append(5)

print("values:", values)
print("a:", a)
```

```
values: [4, 11, 5]
a: [4, 11, 5]
```

Odkazy a kopie

- kopii seznamu vytvoříme pomocí `a = list(values)`

```
values = []
values.append(3)
values.append(11)
values[0] = 2
a = list(values)
a[0] = 4
a.append(5)

print("values:", values)
print("a:", a)
```

```
values: [2, 11]
a: [4, 11, 5]
```

Slicing

```
>>> s = [3, 4, 5, 9, 1, 4, 6]
>>> print(s[1:])
[4, 5, 9, 1, 4, 6]
>>> print(s[:4])
[3, 4, 5, 9]
>>> print(s[1:4])
[4, 5, 9]
```

```
>>> print(s[:5:2])
[3, 5, 1]
>>> print(s[::2])
[3, 5, 1, 6]
>>> print(s[::-1])
[6, 4, 1, 9, 5, 4, 3]
```

Příklady - seznamy

- 5.1.1. Součet, maximum a hledání

Napište funkce nad seznamem čísel, které zjistí:

`my_sum(numbers)` - součet všech čísel v seznamu
`my_max(numbers)` - nejvyšší číslo v seznamu,
`my_in(x, array)` - zda se určitá hodnota vyskytuje v seznamu.

Tedy ekvivalenty operací `max`, `sum` a `in` (ale s použitím pouze základních operací nad seznamy).

Příklady - seznamy

- 5.1.2. Součin nenulových čísel

Napište funkci `nonzero_product(numbers)`, která vypočítá součin čísel v seznamu, ale ignoruje přitom případné nuly.

```
>>> print(nonzero_product([0, 2, 3, 0, 0, 3]))
18
>>> print(nonzero_product([0, 0, 0, 0]))
1
```

Příklady - seznamy

- 5.1.3. Modifikace vs. vytváření nového seznamu

Napište funkci `double_all(numbers)`, která dostane na vstupu seznam čísel a každý jeho prvek vynásobí dvěma. Dále napište funkci `create_doubled(numbers)`, 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.

Řetezce

- datový typ `str`
- nemodifikovatelný (immutable)

```
text1 = "postovni holub" # pouzivame uvozovky
print(type(text1))
text2 = 'postovni holub' # pouzivame apostrofy
print(type(text2))

print((text1 == text2)) # oba retezce jsou stejne
text3 = "" # prazdny retezec (obsahuje nula znaku)
```

```
<class 'str'>
<class 'str'>
True
```

Čísla a řetězce

```
cislo = 6
print(type(cislo))
text = "6"
print(type(text))
print((text == cislo))
```

```
<class 'int'>
<class 'str'>
False
```

Speciální znaky

- začínají `\` (např `\n` pro nový řádek)
- znak `\` "deaktivujeme" zápisem `\\`

Víceřádkové řetězce

```
text = """Tento retezec
ma dva radky."""
print(text)
```

- použití jako víceřádkový komentář (hodnotu řetězce nikam neukládáme)

```
"""
def foo():
    some problematic code
"""
```

Operace na řetězcích - `+`, `*`, `len`

```
text = "Ahoj" + " " + "Hugo"
print(text[0])
print(text[8])
print(len(text))
print("Ha" * 3)
```

```
A
o
9
HaHaHa
```

Řetězce nelze modifikovat

```
>>> text = "Ahoj Hugo"
>>> text[0] = "O" # vyhodi chybu
---------------------------------------------------
TypeError    Traceback (most recent call last)
<ipython-input-3-dce0d8239f5c> in <module>()
      1 text = "Ahoj Hugo"
----> 2 text[0] = "O" # vyhodi chybu

TypeError: 'str' object does not support item assignment
```

Operace na řetězcích - `in`

```
text = "Ahoj Hugo"
print("A" in text)
for znak in text:
    print(znak, end=" ")
```

```
True
A h o j   H u g o
```

Operace na řetězcích - `ord`, `chr`

```
print(chr(97))
print(ord("A"))
print("ahoj".upper())
```

```
a
65
AHOJ
```

Operace na řetězcích - slicing

```
text = "Ahoj Hugo"
print(text[2:5])
print(text[1:])
print(text[:])
print(text[::2])
print(text[-5:-1])
print(text[::-1])
```

```
oj
hoj Hugo
Ahoj Hugo
Ao uo
 Hug
oguH johA
```

Příklady na řetězce

- 5.2.1. Prokládání textu textem
  Napište funkci `dummy(text, rubbish)`, která mezi každá dvě písmena daného textu vloží dodaný text.

```
>>> print(dummy('pampeliska', 'XX'))
pXXaXXmXXpXXeXXlXXiXXsXXkXXa
```

- 5.2.3. Pozpátku
  Napište funkci `reverse(text)`, která vám vrátí řetězec s písmeny uspořádanými pozpátku.

Příklady na řetězce

- 5.2.4. Cenzura
  Napište funkci `censorship(text)`, která zcenzuruje dodaný řetězec tak, že každé druhé písmeno nahradí za `X`.

```
>>> print(censorship("Vinnetou: Jsem krestan."))
VXnXeXoX:XJXeX XrXsXaX.
```

Příklady na řetězce

- 5.2.8. Palindrom
  Napište funkci `palindrom(text)`, která vrátí, zda je řetězec palindromem. Palindromem je takové slovo či věta, která má při čtení v libovolném směru stejný význam, například nepotopen či jelenovi pivo nelej (mezery můžete ignorovat).

```
>>> print(palindrom("JELENOVIPIVONELEJ"))
True
```

Příklady na řetězce - šifry

- 5.3.1. Caesarova šifra
  Napište funkci `caesar(text, klic)`, která zašifruje text tak, že posune všechna jeho písmena v abecedě o n dopředu (cyklicky).

```
>>> print(caesar('zirafa', 3))
CLUDID
```

Příklady na řetězce - šifry

- 5.3.2. Vigenèrova šifra
  Napište funkci `vigenere(text, key)`, která zašifruje text podle předem daného klíče. Pro posun písmen zdrojového textu se postupně používají písmena z klíče: `a` posouvá o `0`, `b` o `1`, … `z` o `25`. Pokud je klíč kratší než zdrojový text, jsou použita písmena z klíče opět od začátku.

```
>>> print(vigenere('pampeliska', 'klic'))
ZLUROWQUUL
```

Závěrem

- práce se seznamy a obvyklé funkce na nich
- kopie x odkazy na řetězce
- řetězce x seznamy
- operace na řetězcích
- druhý domácí úkol do pátku
- blíží se první vnitrosemestrálka