IB111 Úvod do programovaní skrze Python

Autor: Slavomír Krupa,
Text inšpirovaný Valdemarom Švábenským

String

... is immutable.

String

- ★ Sekvencia znakov (čísla, písmená, symboly) uzavretých
- ★ Úvodzovkách
- ★ apostrofoch

```
a = "#@#$56čľýá"
```

```
b = '#@#$56čľýá'
```

```
a == b ???
```

SPECIAL CHAR

```
print("Continue\nHere I am.")
print("What about TAB\t?")
```

```
Continue
Here I am.
What about TAB	?
```

Escape special char

```
print("It's not possible.")
print('It\'s not possible.')
print("Advanced escape\: \"\\n\".  ")
```

```
It's not possible.
It's not possible.
Advance escape: "\n".
```

Length and indices

```
#       012345
s  =  "Python"
```

INDEX

ZNAK

```
a = len(s)
b = s[0]
c = s[2]
d = s[len(s)]
e = s[len(s)-1]
s[0] = s[3]
```

```
a = 6
b='P'
c='t'
d= IndexError
e='n'
 TypeError
```

Indices/Slices

- Volajú sa cez hranaté zátvorky
- Index
  - Jeden parameter - index znaku (indexuje sa od 0)

```
"Python"[index]
```

- Rez - 0 až 3 parametre - oddelené dvojbodkou (dvojtečkou)
  - `start` - je zahrnutý do rezu
  - `end` - nie je zahrnutý do rezu
  - `step`- dĺžka kroku

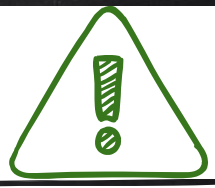
```
"Python"[start:end:step]
```

Slices

```
#      012345
s = "Python"
```

INDEX

ZNAK



```
a = s[0:len(s)]
b = s[1:4]
c = s[2:6]
d = s[3:]
e = s[:3]
f = s[:]
```

```
a='Python'
b='yth'
c='thon'
d='hon'
e='Pyt'
f='Python'
```

Moar slices

```
#      012345
s  =  "Python"
#-     654321
```

Index

znak

Záporný index

```
a = s[-6:-1]
b = s[-5:-2]
c = s[2:5:2]
d = s[::2]
e = s[::-1]
```

```
a='Pytho'
b='yth'
c='to'
e='Pto'
f='nohtyP'
```

Iterate over string

```
encrypted_text = ""
for char in "plain text":
    encrypted_text += char + "Xx"
```

```
print(encrypted_text)
>>"pXxlXxaXxiXxnXx XxtXxeXxxXxtXx"
print(encrypted_text[::3])
>>"plain text"
```

In / not in

```
#       012345
s = "Python"
```

INDEX

ZNAK

```
a = "p" in s
b = "Y" not in s
c = "th" in s
d = "n!" not in s
```

```
a = False
b = True
c = True
d = True
```

Functions

```
"Python".lower()
```

- Volajú sa cez bodkovú notáciu
- Nemodifikujú pôvodnú premennú
- Sú case sensistive
- `lower()` - zmení len znaky abecedy z ľubovoľných na malé
- `upper()` - zmení len znaky abecedy z ľubovoľných na veľké
- `count(str)` - vráti počet výskytov reťazcu
- `find(str)` - vráti prvý index reťazcu
- `replace(str1,str2)` - vymení výskyty `str1` za `str2`
- `split(str)` - rozdelí reťazec na zoznam* podľa oddeľovača

Functions

```
s = "HoHoho!"
```

```
a = s.lower()
b = s.upper()
c = s.count("H")
d = s.count("ho")
```

```
a = "hohoho!"
b = "HOHOHO!"
c = 2
d = 1
```

Functions

```
s = "HoHoho!"
```

```
a = s.replace("o", "a")
b = s.replace("Ho", "Ha")
c = s.replace("o", "")
d = s.find("!")
e = s.find("Hoh")
```

```
a = "HaHaha!"
b = "HaHaho!"
c = "HHh!"
d = 6
e = 2
```

Functions

```
a = "One does not simply parse this" \
    " sentence.".split(" ")
b = "HoHoho!".split("o")
```

```
a = ['One', 'does', 'not', 'simply',
'parse', 'this', 'sentence.']
b = ['H', 'H', 'h', '!']
```

Functions

- Funkcie pre znaky
  - Znak - reťazec dĺžky 1
  - Z ASCII tabuľky - http://www.ascii-code.com/
- `ord(char)` - vracia celočíselný ASCII kód znaku, ktorý je zadaný ako parameter
- `chr(int)` - vracia znak, ktorého kód je zadaný ako parameter

Functions

```
# 65 = A
a = ord('A')
b = chr(65)
c = chr(66)
d = chr(ord('C'))
```

```
a = 65
b = "A"
c = "B"
d = "C"
```


LIST

List (zoznam)

- Umožňuje uložiť viacero hodnôt v danom poradí
  - V pythone dynamická veľkosť - rastie za behu
  - V pythone možné miešať typy uložených objektov

```
empty_list = []
a = ['One', 'does', 'not', 'simply',
'parse', 'this', 'sentence.']
```

String vs zoznam

- Reťazec je špeciálny typ zoznamu, ktorého položky sú jednotlivé znaky
  - Zoznam je meniteľný!

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

String vs list

- Prístup cez indexy `list[index]`
- Rezy zoznamu `list[start:end:step]`
- Iterácia cez prvky zoznamu `for item in list:`
- Operátory (`in` / `not in`) `"p" in list`
- Funkcie nad reťazcami `"Python".lower()`

Navyše:

- Zmena prvku zoznamu `list[0] = list[3]`

String vs list

```
list = []
list += "a"
print(list)
#['a']
list.append("b")
print(list)
# ['a', 'b']
```

```
string = ""
string += "a"
print(string)
# 'a'
string.append("b")
print(string)
# 'ab'
```

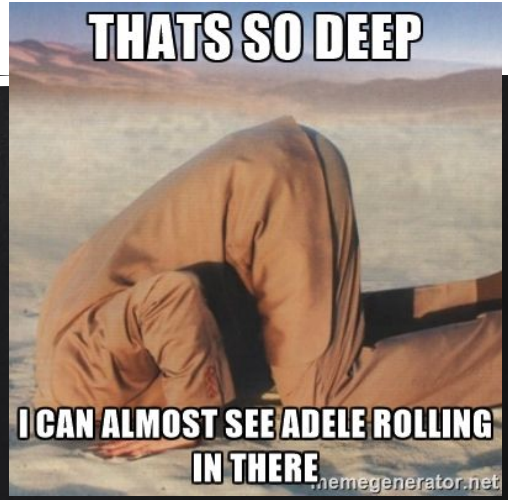
Copy

```
code = [7,3]
copy = code
copy.append(8)
print(code)
# [7, 3, 8]
print(copy)
# [7, 3, 8]
```

```
code = [7,3]
copy = list(code)
copy.append(8)
print(code)
# [7, 3]
print(copy)
# [7, 3, 8]
```

Deep copy*

```
from copy import deepcopy
lst = [['a'], ['b', 'c']]
copy = deepcopy(lst)
```



Zdroj: https://memegenerator.net/instance/46722994

Pair programming

Úloha 1

Napíšte funkciu, ktorá vráti nový reťazec, v ktorom bude každé písmenko zdvojené.

```
>>> duplication('python')
'ppyytthhoonn'
>>> duplication('a')
'aa'
```

Úloha 2

Napíšte funkciu, ktorá spočítá počet výskytov písmena (znaku) A/a.

```
>>> count_a('Liska Adelka')
3
>>> count_a('a')
1
```

Úloha 3

- ★ Napíšte funkciu `nonzero_product(list)`, ktorá vypočíta súčin čísel v zozname `list`, ale ignoruje prípadné nuly.
- ★ Nemusíte ošetrovať prípad keď prídu samé nuly.

```
>>>nonzero_product([1,2,5])
10
>>>nonzero_product([0,1,0])
1
```

Úloha 4

Napíšte funkciu `chunk(string, length)`, ktorá vezme reťazec `string` a na obrazovku postupne vypíše jeho rozkúskované podreťazce dĺžky `length`.

```
>>> chunk('abcd',2)
'ab'
'cd'
>>> chunk('a',8)
'a'
```

Úloha 5

Napište funkci `censorship(string,forbidden)`, ktorá zcenzuruje dodaný reťazec `string` tak, že každý znak, ktorý je v reťazci `forbidden` nahradí za

```
>>>censorship('Fork','ro')
F**k
>>>censorship('Simple','z')
Simple
```

Úloha 6

Napíšte procedúru `diagonal(string, lines_count)`, ktorá vypíše reťazec string šikmo do `lines_count` riadkov

```
>>> diagonal('abcdefgh',2)
a c e g
 b d f h
>>> diagonal('abcdefgh',3)
a d g
 b e h
  c f
```

Úloha 7

- Napíšte funkciu `char_numbers(string)`, ktorá nahradí reťazec malých písmen číslami, ktoré odpovedajú poradiu týchto písmen v anglickej abecede
- Výsledný zoznam je návratovou hodnotou funkcie

```
>>> char_numbers('abcde')
[1,2,3,4,5]
>>> char_numbers('hello')
[8,5,12,12,15]
```

Úloha 1*

- ★ Napíšte funkciu `reverse(string)`, ktorá vezme reťazec string a vráti ho otočený
- ★ Bez použitia reverse a `[::-1]`

```
>>> reverse('abcde')
'edcba'
>>> reverse('a')
'a'
```

Úloha 2*

- ★ Napíšte funkciu `is_palindrome(string)`, ktorá overí či zadaný reťazec je palindróm.
- ★ Použite funkciu z úlohy reverse

```
>>> is_palindrome('jelenovipivonelej')
True
>>> is_palindrome('afk')
False
```

Úloha 3*

Napíšte funkciu `caesar(string, shift)` pre šifrovanie reťazca string Caesarovou šifrou posunom o `shift` celočíselných pozícií

```
>>> caesar('a',2)
c
>>> caesar('ahoj',1)
bipk
```

Úloha 4*

- ★ Napíšte funkciu `random_string(length)`, ktorá vygeneruje náhodné slovo z malých znakov anglickej abecedy.
- ★ Použite správne rozsahy z ASCII tabuľky a funkciu char .

```
>>> random_string(2)
fg
>>> random_string(6)
brdpio
```

Úloha 5*

Napište funkci, 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. Můžete se inspirovat popisem Vigenèrovy šifry.

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

Úlohy*

- ★ https://www.fi.muni.cz/IB111/sbirka/05-retezce_a_seznamy.html
- ★ 5.2,11
- ★ 5.2.13

Credits

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

- Presentation template by SlidesCarnival
- Photographs by Unsplash