Ťahák - Základy jazyka Python

Jaroslav Výbošťok, v190427

V tomto ťaháku nájdete:
Vstup a výstup, Cyklus FOR, IF, Cyklus s podmienkou, Stringy, Náhodné hodnoty, Funkcie, Zoznamy, Textové súbory – čítanie a zápis, Slovníky, Lambda funkcie,

Vstup a výstup:

```
print('Hello world!')  #môj prvý program
print()                #zariadkuje, nezabudnúť zátvorky
print('výpočet je', 3 + 4 * 5, 'cm', sep='', end='')
```

Nepovinné parametre **sep** (znak oddeľujúci jednotlivé položky výpisu) a **end** (čo vypíše print na konci – v tomto prípade nezariadkuje)

```
slovo = input('Zadaj slovo')   #načítanie hodnoty typu string
x = int(input('Zadaj číslo:'))
x = int(input('Zadaj ' + str(poradie) + ' číslo:'))  #na rozdiel od printu, input pri
```

výpise výzvy chce iba jeden parameter typu string, preto treba spájať pomocou +

```
input()                        #bude sa čakať na stlačenie klávesu ENTER
```

Základné typy hodnôt: int, float, str, bool

Základné aritmetické operácie a funkcie:

```
// celočíselné delenie         22 // 7 má hodnotu 3
%  zvyšok po delení            22 % 7 má hodnotu 1
** umocňovanie                 2 ** 8 má hodnotu 256
*  viacnásobné zreťazenie      3 * 'la' má hodnotu 'lalala'
```

```
premenná += hodnota     pripočítaj hodnotu k premennej
premenná *= hodnota     vynásob premennú zadanou hodnotou
premenná /= hodnota     vydeľ premennú zadanou hodnotou
premenná //= hodnota    celočíselne vydeľ premennú zadanou hodnotou
```

```
vysledok = abs(-3)                          3
zaokruhlene = round(8.2434343746, 2)        8.24
```

Ďalšie matematické funkcie (napr. sqrt) sa nachádzajú v module **math**.

Cyklus FOR:

```
for i in range(5, 10):
    print(i)
```

```
range(3, 10)          3,4,5,6,7,8,9
range(3, 10, 2)       3,5,7,9
range(10, 1, -1)      10,9,8,7,6,5,4,3,2
range(10, 1)          prázdna postupnosť
range(1, 1)           prázdna postupnosť
```

IF:

```
if teplota >= 30:
    print('Je nam teplo!')
else:
    print('Je nam zima!')
```

```
if 100 > body >= 90:
    print('za', body, 'bodov získavaš jednotku')
elif body >= 80:
    print('za', body, 'bodov získavaš dvojku')
else:
    print('rachol si')
```

OPERÁTORY A LOGICKÉ VÝRAZY:

```
nerovná sa      !=
rovná sa        ==
```

```
and, or, not
x < 10 and y < 20
10 < x < 20
True, False  #T a F sú VEĽKÉ PÍSMENÁ
```

Ternárny operátor:

```
print('Kladne' if cislo >= 0 else 'Zaporne')
x = 10 if a > 100 else 20
```

CYKLUS S PODMIENKOU:

```
pocet = 20
while pocet > 15:
        pocet -= 2
```

ZNAKY A TEXTOVÉ REŤAZCE (STRINGY)

```
slovo = 'Ahoj'
dlzka = len(slovo)                    4  # Vráti dĺžku reťazca, teda počet znakov
```

Pozor: dĺžka reťazca je 4 ale posledný znak má index 3, lebo **sa indexujú od nuly**

```
textove = str(1024)                   '1024'          # Konverzia int na string
slova = slovo + ' Ferko'                              # Spájanie reťazcov
```

Ukážka využitia konverzie čísel na string a spájania stringov:

```
cnv.create_text(50, 100, text='Vysledok je '+str(cislo)+ 'metrov')
```

```
if '42'.isdigit():                    True
        print('Je to číslo')

if 'ibaabeceda'.isalpha():            True
        print('Sú tam iba písmenká')
```

Znaky a ich kódy z tabuľky ASCII

```
znak = 'a'
kod = ord(znak)                       97
znak = chr(66)                        'B'
if znak >= 'a':
        #porovnávať môžeme priamo znaky, netreba ani konvertovať na ordinálne hodnoty
```

Prechádzanie reťazca po znakoch

```
for znak in slovo:
        print(znak)
```

alebo – hlavne ak potrebujeme vedieť pozície znaku v reťazci –

```
for i in range(len(slovo)):
        print(slovo[i])
```

Užitočné metódy stringov

```
velke = veta.upper()                  #vráti string konvertovaný na veľké písmená
male = veta.lower()                   #na malé písmená
orezane = veta.strip()                #odreže na začiatku a konci medzery, \n, tab
rozdelene = veta.split(' ')           #rozdelí na zoznam podľa zadaného oddeľovača
nahradeny = veta.replace('ab', 'ba')  #nahradí podreťazce v reťazci
kolko = veta.count('ab')              #spočíta počet výskytov podreťazca
kde = veta.find('a')                  #vráti polohu prvého nájdeného výskytu
```

NÁHODNÉ HODNOTY – modul random:

```
from random import *

cislo = randrange(100)                        #od 0 do 99
cislo = randrange(10, 100)                    #od 10 do 99
cislo = randrange(10, 100, 5)                 #náhodne z 10,15,20...95
farba = choice(('green', 'red', 'blue'))      #náhodný výber jednej z možností
```

```
vyzrebovane = sample((5,8,7,1,4,13,15),3)
#vyberie náhodnú trojicu z daných prvkov, pričom sa nebudú opakovať
shuffle(zoznam)        #zamieša prvky priamo v zozname (nevracia hodnotu)
```

FUNKCIE A GLOBÁLNE PREMENNÉ

return vráti hodnotu a ukončí funkciu
Vždy treba písať () - bez zátvoriek je to adresa funkcie (viď command pre button)

```
def nazov():            #nezabudnúť zátvorky
      print('tralala')
      return 10

def sparametrami(px, py, farba='blue', farba2='red'):
      global x, y       #premenné x,y v tejto funkcii budú globálne
      g.create_rectangle(px, py, px+10, py+10, fill=farba)
      x += 1
      y += 5            #musia byť inicializované v hlavnom programe
```

ZOZNAMY (POLIA)

Indexuje sa od 0 do len(zoznam)-1

"Deklarácia", resp. inicializácia: **a = [0] * 5** je to isté ako a=[0, 0, 0, 0, 0]
resp. rovno dávame počiatočné hodnoty: a = [21,13,'red',5,29.17,None,42]

dlzka = **len**(a) - vráti počet prvkov zoznamu,

```
for i in range(len(zoznam)): #akurát to sedí na 0 až len(zoznam)-1
  print(a[i])
```

Cyklus cez prvky zoznamu:

```
#vypíše rovno prvky zoznamu (bez range)
for prvok in zoznam:
  print(prvok)
```

"Naťahovanie zoznamu"

```
zoz = []                #Vytvor prázdny zoznam
zoz.append(42)          #Pridaj prvok na koniec zoznamu
zoz += [10, 'red', 50]
```

Zoznam vs. string

zoznam = **list**('ahoj') #takto spravím zo immutable stringu zoznam, aby som mohol meniť jeho jednotlivé znaky, dostanem tak zoznam stringov: ['a', 'h', 'o', 'j']

S = ''.**join**(zoznam) #všetky prvky spojí do reťazca a ako oddeľovač medzi nimi dá prázdny string, takže vyjde z toho jeden string z pôvodných prvkov zoznamu

Rezy (Slice):
'Vyrezávame' (vyberáme) časti zoznamov alebo textových reťazcov
Výsledok rezu na stringu je string, výsledok rezu na zozname je zoznam

```
print(zoz[1:4]) #vypíše prvky s indexami 1 až 3 (teda 4. nie)
print(zoz[:5], zoz[7:], zoz[::2], zoz[::-1])
```

Rôzne metódy zoznamu

```
zoznam.count(hodnota)  #vráti počet výskytov hodnoty v poli (alebo v n-tici)
zoznam.index(hodnota)  #vráti index prvého výskytu hodnoty v poli (alebo v n-tici)
zoznam.append(hodnota) #pridá novú hodnotu na koniec pôvodného poľa
zoznam.insert(index, hodnota) #vloží hodnotu do pôvodného poľa pred zadaný index
zoznam.pop()  #odstráni posledný prvok pôvodného poľa a vráti tento prvok ako hodnotu
zoznam.pop(0) #odstráni prvý prvok pôvodného poľa a vráti tento prvok ako hodnotu
zoznam.remove(hodnota) #vyhodí z pôvodného poľa prvý výskyt hodnoty
zoznam.sort() #vzostupne utriedi pôvodné pole, prvky poľa sa musia dať porovnávať
```

Zabudované funkcie pre prácu so zoznamom

```
sucet = sum(zoznam)
najmensi = min(zoznam)
najvacsi = max(zoznam)
utriedeny = sorted(zoznam)
```

TEXTOVÉ SÚBORY – ČÍTANIE PO RIADKOCH

Pozn: Na súťažiach ako KSP/Zenit/OI používajte funkciu input() akoby ste vstup riadok po riadku zadávali ručne

Napr: Vstup: 15 8 13 79 4 12 6

```
      zoz = input().split()              #vráti zoznam STRINGOV
      zoznamcisel = list(map(int,zoz))   #vysledkom bude zoznam hodnot typu int
```

Čítanie v cykle

```
subor = open('udaje.txt')
for riadok in subor:
      print(riadok.strip())
subor.close()
```

alebo využitím funkcie readline()

```
subor = open('udaje.txt')
riadok = subor.readline()
while riadok != '' :    #kým nedôjdem na koniec súboru
      print(riadok)
      #...
      riadok = subor.readline()
      #načítam ďalší riadok, ktorý sa spracuje v ďalšej iterácii cyklu
subor.close()
```

Iné možnosti

```
subor = open('udaje.txt')
riadok = subor.readline()     #načíta jeden riadok ako string, na konci bude znak \n
riadok = riadok.strip()
#zruší netlačiteľné znaky na začiatku a konci reťazca – teda aj \n prípadne medzery

riadky = subor.readlines() #načíta zvyšné riadky ako ZOZNAM stringov
print(riadky)     ['Prvy riadok\n', 'Druhy riadok\n']
subor.close()
```

Príklad načítavania údajov oddelených medzerou do dvojrozmerného poľa stringov:

```
subor = open('udaje.txt')
mapa = []
for riadok in subor:
      mapa.append(riadok.split())   #split sa postará aj o znak konca riadku
subor.close()
```

Pozn. Existujú aj ďalšie spôsoby: file.read(), konštrukcia with, ...

Zápis do súboru

```
subor = open('udaje.txt', 'w')      #založí nový prázdny súbor pripravený na zápis
subor.write('Prvy\n')
subor.write('Druhy')
subor.close()                 #Údaje sa reálne zapíšu až pri zatvorení súboru!!!
```

Iný prístup:

```
subor = open('udaje.txt', 'w')
print('Prve', cislo, 'lalala' ,file=subor)
```

```
subor.close()
#print môže zapísať viac parametrov, automaticky zariadkuje, možno využiť sep...
```

SLOVNÍKY (ASOCIATÍVNE POLIA)

```
slovnik = {} alebo rovno slovnik = {'auto':'car','pes':'dog','pocet':2}
slovnik['farba'] = 'color'    #Pridá nový kľúč alebo prepíše hodnotu
slovnik['pocet'] = 1
print(slovnik['pocet'])

print(slovnik.values())       #vráti zoznam hodnôt
print(slovnik.keys())         #vráti zoznam kľúčov
print(slovnik.items())        #vráti zoznam dvojíc (kľúč, hodnota)

print(slovnik.get('kitten', 'Nie je tam'))
slovnik['pocet'] = slovnik.get('pocet',0) + 1 #užitočné použitie metódy get – ak kľúč
neexistuje, zriadi ho s hodnotou jeden, ak existuje, zvýši ho o jedna
```

LAMBDA FUNKCIE – ABY SA NEMUSELA DEFINOVAŤ FUNKCIA S NÁZVOM

```
Button(text='…', command = lambda: print('Toto je telo lambda funkcie')).pack()
g.bind('<Button-1>', lambda event: print('Kliklo sa na', event.x, event.y))
#lambda funkcia s parametrom

zoznammocnin = list(map(lambda x: x**2,zoznamcisel))

g.bind('<Button-1>', lambda event: (print('Prvy prikaz'), print('druhy prikaz')) )
#používať s rozumom a výnimočne
```