Python tutorijal

- praktični primeri -

Sadržaj:

1. Upoznavanje sa razvojnim okruženjem
   - Python proširenje (PTVS) za Visual Studio 2015
   - Druga razvojna okruženja
2. Osnovi Python jezika
   - Sintaksa i konvencije
   - Kontrola toka programa
   - Osnovni tipovi podataka
   - Prosleđivanje parametara (funkcija i komandna linija)
   - Osnovne funkcije za rad sa datotekama
3. Vežbanje
   - Kontrola toka programa, petlje i strukture podataka
   - Funkcije i rad sa datotekama
   - Prosleđivanje ulaznih argumenata programa

Upoznavanje sa Python razvojnim okruženjem

Python razvojno okruženje (1/2)

Visual Studio 2015 i PTVS:

- Python 3.x
- Visual Studio 2015
- Python Tools for Visual Studio (PTVS)

Python in VS 2015:

- Pravljenje novog projekta
- Podešavanje Python run-time okruženja
- *Hello world* program
  - *main* funkcija
  - moduli
  - slobodne funkcije

Python razvojno okruženje (2/2)

Druga razvojna okruženja:

- Visual Studio 2015 Community/Enterprise + PTVS (Win)
- Bilo koji tekst editor: Notepad++, Vi, Gedit
- Python IDE (Win/Linux/MacOS)
- PyCharm (Win/Linux/MacOS)
- Enthought Canopy (Win/Linux/MacOS)

Sintaksa i konvencije (1/7)

Struktura programskog koda:

- Jedan iskaz po liniji

```
a = x + var
```

- Razlomljeni iskazi

```
math.sqrt(math.sin(b) + \
    math.cos(c))
```

- Komentari počinju sa #

```
#Comment msg
a = x + var
```

- Indentacija definiše programski blok

```
# Function foo
def foo(n):
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
        return result
```

```
# Function foo
def foo(n, data):
    ret = []
    if n==False:
        ret = process1(data)
    else:
        ret = process2(data)
    return ret
```

Sintaksa i konvencije (2/7)

Struktura programskog koda:

- Jedan iskaz po liniji

```
a = x + var
```

- Razlomljeni iskazi

```
math.sqrt(math.sin(b) + \
    math.cos(c))
```

- Komentari počinju sa #

```
#Comment msg
a = x + var
```

- Indentacija definiše programski blok

```
# Function foo
def foo(n):
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b !!!
    return result
```

```
# Function foo
def foo(n, data):
    ret = []
    if n==False:
        ret = process1(data)
    else: !!!
        ret = process2(data) !!!
    return ret
```

Sintaksa i konvencije (3/7)

Struktura programskog koda:

- Indentacija definiše programski blok

```
# Function foo
def foo(n):
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
        return result
```

```
# Function foo
def foo(n, data):
    ret = []
    if n==False:
        ret = process1(data)
    else:
        ret = process2(data)
    return ret
```

- Preporučuje se upotreba praznih mesta umesto tabulatora
  - Podesiti editor tako da 1 tab bude jednak 4 prazna mesta

Sintaksa i konvencije (4/7)

Imena promenljivih, funkcija i klasa:

- Mogu početi slovom ili donjom crtom
- Mogu sadržati brojeve
- Nisu dozvoljeni
  - Specijalni znaci: @, %, $ itd.
  - Rezervisanje reči
- Izbegavati korišćenja donje crte na početku
- Rezervisane reči:

```
and as assert break class continue
def del elif else except exec finally
for from global if import in is lambda
nonlocal not or pass print raise return
try while with yield . . .
```

Sintaksa i konvencije (5/7)

Literali:

- Numerički literali:
  - Boolean (True, False)
  - Celobrojna vrednost (1, 65536, 0xDEADBEEF, itd.)
  - Vrednost sa pokretnim zarezom (1.44, 1.2345e+02, itd.)
  - Kompleksna vrednost (4 + 5j)
- Znakovni literali:
  - Niz znakova (“ovo je string”, ‘i ovo je string’, itd.)
  - Nizovi znakova predstavljeni su u ASCII formatu (1 znak – 1 byte)
  - Specijalni karakteri (\t \n itd.)

Sintaksa i konvencije (6/7)

Aritmetički operatori (C\C++ like):

- Operatori se ponašaju očekivano: +, -, /, *, %
- Specijalni operatori:
  - Operator // (deljenje sa zaokruživanje na prvi manji broj)
  - Operator ** (a ** b – broj *a* stepenovan *b* puta)

Operatori poređenja (C\C++ like):

- Operatori se ponašaju očekivano: >, <, ==, !=, >=, <=
  - Povratna vrednost je True ili False

Logički operatori:

- Operatori se ponašaju očekivano: *and*, *or*, *not*
  - Povratna vrednost je True ili False

Sintaksa i konvencije (7/7)

Operatori dodele (C\C++ like):

- Operatori se ponašaju očekivano: `=`, `+=`, `-=`, `/=`, itd.

Operatori za rukovanje bitima (C\C++ like):

- Operatori se ponašaju očekivano: `<<`, `>>`, `&`, `|`, `^`, `~`

Operatori identiteta (*Identity operators*):

- Proverava da li su dve vrednosti (ili promenljive) identične, tj. da li se nalaze na istoj memorijskoj lokaciji
  - Ako su dve vrednosti jednake ne implicira da su identične
- Operatori identiteta: *is*, *is not*

Kontrola toka programa (1/4)

Iskazi if/else/elif:

- Iskazi *if* i *else* imaju istu semantiku kao u C\C++ jeziku
- Iskaz *elif* je skraćeno od ‘else-if’

```
# if/else/elif example
if x < 0:
    print(“x < 0”)
elif x == 0:
    print(“x = 0”)
else:
    print(“x > 0”)
```

```
# if/else example
if x < 0:
    print(“x < 0”)
else if x == 0:
    print(“x = 0”)
else:
    print(“x > 0”)
```

Kontrola toka programa (2/4)

while petlja:

- *while* petlja služi za uzastopno izvršenje niza iskaza dok je određeni uslov zadovoljen

while_stmt ::= "while" expression ":" suite
["else" ":" suite]

```
# while loop example
x = 100
print(“x < 0”)
while x !=  0:
    print(“x = ”, x)
    x -= 1
```

Kontrola toka programa (3/4)

for petlja:

- *for* petlja služi za prolazak (itreriranje) kroz skupove podataka kao što su znakovni nizovi, torke i liste (sekvence)

  for_stmt ::= "for" target_list "in" expression_list ":" suite
  ["else" ":" suite]

```
# for loop example
x = 1, 2, 3, 4, 5
for it in x:
    print("it = ", it)
```

- Za potrebe inicijlaizacije koristiti **range(*start*, *stop*, [*step*])** klasu

```
# for loop example
for it in range(5):
    x = it
```

Kontrola toka programa (4/4)

Iskaze **return**, **break** i **continue** koristiti u skladu sa C\C++ semantikom

- ***break*** iskaz prekida izvršavanje petlje unutar koje se nalazi
- ***continue*** iskaz započinje novi ciklus petlje unutar koje se nalazi
- ***return*** iskaz napušta izvršavanje funkcije unutar koje se nalazi

Osnovni tipovi podataka (1/4)

Osnovne kolekcije su:

- **Lista** (list)
  - Može se menjati (mutable)
- **Rečnik** (dict)
  - Može se menjati (mutable)
- **Torke** (tuples)
  - Ne mogu se menjati (immutable)
- **Skupovi** (sets)
  - Mogu se menjati (mutable)
- **Znakovni nizovi** (strings)
  - Ne mogu se menjati (immutable)

```
l = [] #empty list
l = [1, 2, “a”, ‘b’]

d = {} #empty dictionary
d = {“key1”: 1, “key2”:2}

t = ((),) #empty tuple
t =(1, 2, ‘a’)

s = set() #empty set
s ={1, 2, 2, 3, 3, 3, 4}

s = “” ili None #empty string
s = “some string”
```

- Za više detalja o strukturama podataka, funkcijama i metodama pogledati zvaničnu dokumentaciju (https://docs.python.org/3/)

Osnovni tipovi podataka (2/4)

Operacije nad kolekcijama (sekvencama):

- s1 + s2 – spajanje
- s1 * n – uzastopno se *s* ponavljanja *n* puta
- v1, v2, v3, v4 = s – raspakivanje (npr: *s* je torka od 4 elementa)
- s[i] – indeksiranje
- s[i:j] – isecanje
- s[i:j:k] – isecanje sa korakom *k*
- x *in* s – da li *s* sadrži *x*
- len(s) – dužina *s*
- any(), sum(), min(), max()

Osnovni tipovi podataka (3/4)

Konverzije:

- int(x, (b)) - Tip *string* u tip *int* baze *b*
- float(x) - Tip *string* u tip *float*
- complex(r,i) - Kompleksni broj od *r* i *j*
- str(x) - Bilo koji tip *string*
- chr(i) - Tip *int* u tip *char* (do 255)
- ord(c) - Tip *char* (Unicode) u tip *int*
- hex(i) - Tip *int* u tip *string*, heksadecimalni zapis
- bin(i) - Tip *int* u tip *string*, binarni zapis
- oct(i) - Tip *int* u tip *string*, oktalni zapis

Osnovni tipovi podataka (4/4)

Logički izrazi:

- **True**, logički tačna trvdnja
- **False**, logički netačna trvdnja
- **0**, None i/ili prazna kolekcija
- **!=0**, kolekcija koja nije prazna

Prosleđivanje ulaznih argumenata funkcijama

Prosleđivanje ulaznih argumenata funkcijama obavlja se tako što se u pozivu funkcije redom navode promenljive koje se prosleđuju

```
#This prints a passed info into this function
def print_info(name, surname, age = 35):
    print("Name: ", name, \
          "Surname: ", surname, \
          "Age ", age)
    return
```

(i) print_info("Petar", "Petrovic", 32)

(ii) print_info("Petar", "Petrovic")

(iii) print_info("Petar") !!! Greška

Napomene:

1. Za sve sekvence koji se mogu menjati (**mutable**) prosleđuje se **referenca**
2. Za sve sekvence koji se ne mogu menjati (**immutable**) pravi se nova **kopija**
3. Jednostavan način da se sekvence pošalje preko vrednosti je *foo(**s[:]**)*

Prosleđivanje argumenata komandne linije

Protrebno je koristiti Python **sys** modul

- *sys.argv* predstavlja listu ulaznih argumenata
- *len(sys.argv)* daje broj ulaznih argumenata
- *sys.argv[0]* je ime programa

```
# Input arguments example (input.py)
import sys
print("argv len: ", len(sys.argv))
print("argv[0]: ", sys.argv[0])
print("argv[1] : ", sys.argv[1])
print("argv[2] : ", sys.argv[2])
```

Poziv: **$python input.py jedan 2**

Izlaz:

```
argv len: 3
argv[0]: C:\path\input.py
argv[1]: jedan
argv[2]: 2
```

Rad sa datotekama

Za rukovanje sa datotekama koriste se funkcije:

- open(*file_name*, *mode*), close()
- write(*str*), read(*size*), readline()
- Kao u C\C++ jeziku režimi rada sa datotekama su: read (‘r’), write (‘w’), append (‘a’), read-write(‘r+’)

```
# Read from input and write to output file
import sys
fin = open(“input_file_name”, ‘r’)
fout = open(“output_file_name”, ‘w’)
for line in fin:
    fout. write(line)
fin.close()
fout.close()
```

Dodatni materijali

Više informacija o svemu do sada pomenutom kao i Python API dostupno je u zvaničnoj **Python** web **dokumentaciji**:

https://docs.python.org/3/

Primer

Napisati progam koji listu parova prvih 10 brojeva čuva u rečniku pod nazivom koji definiše korisnik

- Ime liste treba preuzeti na ulazu
- Parovi brojeva (1,2), (3,4) ... (9,10) predstavljaju torke
- Lista se čuva u rečniku koji mapira vrednost string na listu

Zadaci

1. Napisati funkciju koja računa zbir prvih *N* prirodnih brojeva;
2. Napisati funkciju koja računa zbir kvadrata prvih *N* prirodnih brojeva; parametar *N* se unosi kao ulazni argument programa;
3. Napiši program koji na ulazu prima dva stringa i na osnovu njih formira i ispisuje novi string koji se sastoji od dva puta ponovljena prva tri karaktera iz prvog stringa i poslednja tri karaktera drugog stringa;
4. Inicijalizovati listu sa prvih 100 brojeva i ispisati je u obrnutom redosledu;
5. Koristeći strukturu rečnink izračunati broj ponavljanja reči koje se nalaze u datoteci dict_test.txt;
6. Napisati program koji u listu smešta četiri torke formata (*integer*, *float*, *string*) i ispisuje ih, nakon čega briše prvu torku koja je ubačena u listu;
7. Ponoviti zadadak 6 ali umesto liste koristiti skup;