Programski jezici

http://www.programskijezici.matf.bg.ac.rs/

Univerzitet u Beogradu
Matematički fakultet

Programske paradigme

Materijali za vežbe

Nastavnik: Milena Vujošević Janičić
Asistent: Branislava Živković

Beograd
2016.

Priprema materijala:
dr Milena Vujošević Janičić, docent na Matematičkom fakultetu u Beogradu
Marjana Šolajić, asistent na Matematičkom fakultetu u Beogradu
Branislava Živković, asistent na Matematičkom fakultetu u Beogradu

Sadržaj

1

Skript programiranje

Potrebno je imati instaliran Python 2.7 na računaru.
Literatura:

(a) https://www.python.org/

(b) http://www.tutorialspoint.com/python

(c) https://wiki.python.org/moin/

1.1 Uvod, kolekcije, matematičke funkcije

1.1.1 Uvodni primeri

Zadatak 1.1 Ispisivanje pozdravne poruke, komentari.

```
# Ovako se pisu komentari
#
# Pokretanje programa iz terminala:
#  $python  hello.py
#
print "Hello world! :)"
```

Zadatak 1.2 Promenljive, niske, formatiran ispis, učitavanje sa standardnog ulaza, aritmetičke i logičke operacije, naredbe grananja.

```

# Promenljive se dinamicki tipiziraju
a = 45
b = 67.45
istina = True
# Niske su konstantne tj. nisu promenljive.
# To znaci da se menjanjem nekog karaktera u niski
# pravi nova niska u memoriji.
niska = "I believe i can fly!"

# Ispis na standardni izlaz
print a
print b
print a, b, istina

# Formatiran ispis
print "\n------Formatiran ispis------\n"
print "Ceo broj: {0:d} \nBroj u pokretnom zarezu {1:f}\nBulovska vrednost: {2:b}\
    \nNiska: {3:s}\n".format(a,b,istina,niska)


# Ucitavanje niske sa standardnog ulaza
print "\n------Ucitavanje sa standardnog ulaza------\n"
string_broj = raw_input("Unesite ceo broj: ")
broj = int(string_broj) # vrsi se konverzija stringa u ceo broj, slicno: float, str

```

```
# Osnovne aritmeticke operacije:
# +, -, *, /, %, ** (stepenovanje)
print "\n------Osnovne aritmeticke operacije------\n"
print broj+4

# Osnovne logicke operacije:
# not, and, or
print "\n------Osnovne logicke operacije------\n"
print istina or False

# Blokovi se ne ogranicavaju viticastim zagradama kao sto je u C-u
# vec moraju biti uvuceni tabulatorom.

# Naredba grananja
print "\n------Naredba granjanja------\n"
if broj%2 == 0:
  print "Unet je paran broj \n"
elif broj%3 == 0:
  print "Unet je broj deljiv sa 3\n"
# Naredbi <<elif>> moze biti vise
else:
  print "Unet je broj koji nije ni paran ni deljiv sa 3\n"

# Naredba <<switch>> ne postoji

# Petlja
print "\n------Petlja <<while>>------\n"
i=1
while i<=10:
  print i
  i=i+1 # i++ ne postoji, moze ili ovako ili i+=1

# Naredba <<break>> iskace iz bloka, isto kao i u C-u
# Naredba <<pass>> je ista kao naredba <<continue>> u C-u

# Funkcije
#
# def ime_funkcije(argumenti):
#     telo funkcije
#

def f1(x,y):
  return x+y

print f1(2,2)

def f2(a):
  return [a,2*a,3*a]

print f2(11.1)
```

Zadatak 1.3 Moduli math i random.

```
# Matematicke funkcije

# Ukljucujemo modul <<math>>
import math

# U ovom moduli se nalaze brojne funkcije kao sto su:
#
# math.sqrt(broj)
# math.log(broj, osnova)
# math.sin(ugao_u_radijanima), math.cos(), ...
# math.exp(stepen)
# math.factorial(broj)
# i druge...
print "\n------Matematicke funkcije------\n"
print math.factorial(6)
print math.log(125, 5)

# Pseudo slucajni brojevi

# Ukljucujemo modul <<random>>
```

```
21 import random
22
23 # Funkcija random() vraca pseudo slucajan broj tipa float iz opsega [0.0, 1.0)
24 print "\n------Pseudo slucajni brojevi------\n"
25 print "Pseudo slucajan broj iz opsega [0.0,1.0)\n"
26 print random.random()
27
28 # Korisne funkcije:
29 #
30 # randint(a,b) - vraca pseudo slucajan ceo broj n iz opsega [a,b]
31 # choice(lista) - vraca pseudo slucajan element iz liste
32 #
```

Zadatak 1.4 Liste.

```
1  # LISTA
2  #
3  # Notacija: [element1, element2, ...]
4  #
5  # Liste mogu sadrzati razlicite tipove podataka
6  lista = [1,2,3.4,"Another brick in the wall",True,[5,False,4.4,'Layla']]
7
8  print "\n------Lista------\n"
9  print lista
10
11 # Prazna lista
12 prazna = []
13
14 # Pristupanje elementima liste
15 print "\n------Pristupanje elementima liste------\n"
16 # Indeksiranje elemenata liste
17 print lista[0]
18 print lista[3][1]
19 print lista[0:3]
20 # Mozemo indeksirati liste unazad, pozicija -1 odgovara poslednjem elementu
21 print lista[-1]
22 # Ukoliko pokusama da pristupimo elementu liste
23 # koji se nalazi na poziciji van opsega interpreter ce nam prijaviti gresku
24 # IndexError: list index out of range
25 # print lista[100]
26
27 print "\n------Provera da li se element nalazi u listi-----\n"
28 if 1 in lista:
29   print "1 se nalazi u listi\n"
30
31 print "\n------Korisne funkcije za rad sa listama------\n"
32 # Ubacivanje elementa na kraj
33 print "Ubacivanje elementa na kraj liste\n"
34 lista.append(3.14)
35 print lista
36
37 # Ubacivanje elementa na odredjenu poziciju u listi
38 print "\nUbacivanje elementa na odredjenu poziciju u listi\n"
39 # list.insert(pozicija, element)
40 lista.insert(2, "Jana")
41 print lista
42
43 #
44 # Korisne funkcije:
45 #
46 # list.remove(x) - izbacuje prvo pojavljivanje elementa x iz liste
47 # list.count(x) - vraca broj koliko puta se element x nalazi u listi
48 # list.index(x) - vraca indeks prvog pojavljivanja elementa x u listi
49 # len(lista) - vraca broj elemenata liste
50 # del lista[a:b] - brise elemente liste od pozicije a do b
51 #
52 # Nadovezvivanje dve liste
53 print "\n------Nadovezvivanje dve liste------\n"
54 lista = lista+["Plava", "Zuta", "Crna"]
55 print lista
56
57
58 # Prolazak kroz listu
```

```
print "\n------Prolazak kroz listu petljom <<for>>------\n"
for i in lista:
  print i

# Poredjenje listi
#
# Dve liste se porede tako sto se njihovi elementi porede redom leksikografski
print "\n------Poredjenje listi------\n"
print "[1,2,3] < [1,2,5]"
print [1,2,3] < [1,2,5]
print "\n['abc','abc','abc'] < ['abc', 'ab', 'abcd']"
print ['abc','abc','abc'] < ['abc', 'ab', 'abcd']
print "\n['a','b','c'] > ['a', 'b']"
print ['a','b','c'] > ['a', 'b']


# Koriscenje liste kao stek strukture podataka
stek = [9,8,7]
# Operacija push je implementirana funkcijom append
stek.append(6)
stek.append(5)
print "\n------Ispisujemo stek------\n"
print stek
# Operacija pop je implementirana funkcijom pop
print "\n------Ispisujemo element dobijem funkcijom pop------\n"
print stek.pop()
print "\n------Ispisujemo znanje nakon pozivanja funkcije pop------\n"
print stek
```

Zadatak 1.5 Skup, katalog, uređene n-torke.

```


# SKUP
#
# Pravljenje skupa od liste
print "\n------Pravljenje skupa od liste------\n"
lista1 = [4,56,34,2,5,6,4,4,6]
skup = set(lista1)

for i in skup:
  print i


# Funkcija <<range>>
#
# range(kraj)
# range(pocetak, kraj[, korak])
print "\n------Funkcija <<range>>------\n"
brojevi = range(10)
for i in brojevi:
  print i

# KATALOG
#
# Katalog je kolekcija uredjenih parova oblika (kljuc, vrednost)
#
# Notacija: {kljuc:vrednost, kljuc:vrednost, ...}
print "\n------Katalog------\n"
prazna_mapa = {} # prazna mapa

mapa1 = {'a' : 3, 'b' : 4, 'c' : 5}

print mapa1

mapa = {"kljuc1":67.7, 6:"Vrednost 2"}

# Pristupanje elementima u mapi
print "\n------Pristupanje elementima u katalogu------\n"
print mapa['kljuc1']

# Prolazak kroz mapu
print "\n------Prolazak kroz katalog------\n"
for kljuc in mapa:
```

```
    print "'{0:s}' => {1:s}\n".format(str(kljuc),str(mapa[kljuc]))
44
# Korisne funkcije
46 #
# map.keys() - vraca listu kljuceva iz kataloga
48 # map.values() - vraca listu vrednosti iz kataloga
# map.has_key(kljuc) - vraca True/False u zavisnosti od toga da li se element
50 # sa kljucem kljuc nalazi u katalogu

52 # Uredjene N-TORKE
print "\n------Torke------\n"
54 torka = ("Daffy","Duck",11)

56 # Pristupanje elementima u torki
print "\n------Pristupanje elementima u torki------\n"
58 print torka[1]

60 print "\n------Ispisivanje torke------\n"
print torka
62
# Poredjenje torki
64 #
# Dve torke se porede tako sto se njihovi elementi porede redom leksikografski
66 print "\n------Poredjenje torki------\n"
print "(1,2,'a') < (1,2,'b')"
68 print (1,2,'a') < (1,2,'b')
print "\n([1,2,3], 'Bugs', 4) < ([1,1,1], 'Bunny', 6)"
70 print ([1,2,3], 'Bugs', 4) < ([1,1,1], 'Bunny', 6)
# Ukoliko torke ne sadrze elemente istog tipa na istim pozicijama, i dalje ih mozemo
    porediti,
72 # ali poredjenje se vrsi na osnovu imena tipa elementa leksikografski
# npr.  element tipa List < element tipa String < element tipa Tuple i slicno
74 print "\n(1,2,['a','b']) < (1,2,'ab')"
print (1,2,['a','b']) < (1,2,'ab')
```

1.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.6 **Pogodi broj** Napisati program koji implementira igricu "Pogodi broj". Na početku igre računar zamišlja jedan slučajan broj u intervalu [0,100]. Nakon toga igrač unosi svoje ime i započinje igru. Igrač unosi jedan po jedan broj sve dok ne pogodi koji broj je računar zamislio. Svaki put kada igrač unese broj, u zavisnosti od toga da li je broj koji je unet veći ili manji od zamišljenog broja ispisuje se odgovarajuća poruka. Igra se završava u trenutku kada igrač pogodio zamišljen broj.

[Rešenje 1.6]

Zadatak 1.7 **Aproksimacija broja PI metodom Monte Karlo** Napisati program koji aproksimira broj PI koriscenjem metode Monte Karlo. Sa standardnog ulaza unosi se broj N. Nakon toga N puta se bira tačka na slučajan način tako da su obe koordinate tačke iz intervala [0,1]. Broj PI se računa po sledecoj formuli:

$$PI = 4 * A/B$$

- A je broj slučajno izabranih tačaka koje pripadaju krugu poluprečnika 0.5, sa centrom u tački $(0.5, 0.5)$
- B je broj slučajno izabranih tačaka koje pripadaju kvadratu čija temena su tačke $(0, 0), (0, 1), (1, 1), (1, 0)$.

[Rešenje 1.7]

Zadatak 1.8 **X-O** Napisati program koji implementira igricu X-O sa dva igrača.

[Rešenje 1.8]

1.1.3 Zadaci za vežbu

Zadatak 1.9 **Anjc** Napisati program koji implementira igricu Anjc sa jednim igračem. Igra se sa špilom od 52 karte. Na početku igrač unosi svoje ime nakon čega računar deli dve karte igraču i dve karte sebi. U svakoj sledećoj iteraciji računar deli po jednu kartu igraču i sebi. Cilj igre je sakupiti karte koje u zbiru imaju 21 poen. Karte sa brojevima nose onoliko bodova koliki je broj, dok žandar, dama, kralj nose 10 bodova. Karta As može da nosi 1 ili 10 bodova, u zavisnosti od toga kako igraču odgovara. Igrač koji sakupi 21 je pobedio. Ukoliko igrač premaši 21 bod, pobednik je njegov protivnik. https://en.wikipedia.org/wiki/Blackjack

Zadatak 1.10 **4 u liniji** Napisati program koji implementira igricu 4 u nizu sa dva igrača. Tabla za igru je dimenzije 8x8. Igrači na početku unose svoja imena, nakon čega računar nasumično dodeljuje crvenu i žutu boju igračima. Igrač sa crvenom bojom igra prvi i bira kolonu u koju ce da spusti svoju lopticu. Cilj igre je da se sakupe 4 loptice iste boje u liniji. Prvi igrač koji sakupi 4 loptice u liniji je pobedio. https://en.wikipedia.org/wiki/Connect_Four

1.2 Datoteke, niske, JSON format, datum

1.2.1 Uvodni primeri

Zadatak 1.11 Funkcije za rad sa niskama.

```
# Niske
#
# Mozemo ih pisati izmedju jednostrukih i dvostrukih navodnika

niska1 = 'Ovo je neka niska.'
niska2 = "People are strange when you're a stranger ."

print "\n------Niske------\n"
print niska1
print niska2

# Karakterima u niski mozemo pristupati koristeci notaciju [] kao kod listi
print "\n------Pristupanje karakterima u niski------\n"
print niska2[4]
print niska2[6:10]

# Duzinu niske racunamo koristeci funkciju len(niska)
print "\n------Duzina niske------\n"
print len(niska1)

# Funkcija count
# niska.count(podniska [, pocetak [, kraj]]) - vraca broj koliko se puta
# podniska nalazi u niski (u intervalu od pocetak do kraj)
print "\n------Funkcija <<count>>------\n"
print niska2.count("strange")

# Funkcija find
# niska.find(podniska [, pocetak [, kraj]]) - vraca poziciju prvog pojavljivanja
# podniska u niski (u intervalu od pocetak do kraj), -1 ukoliko se podniska ne nalazi
    u niski
print "\n------Funkcija <<find>>------\n"
print niska2.find("are")

# Funkcija join
# niska_separator.join([niska1,niska2,niska3,...]) - spaja listu niski separatorom
print "\n------Funkcija <<join>>------\n"
print ' '.join(["Olovka",'pise','srcem.'])


# Korisne funkcije za rad sa niskama:
#
# niska.isalnum()
#       isalpha()
#       isdigit()
#       islower()
```

```
#       isspace()
#       isupper()
# niska.split(separator) - razlaze nisku u listu koristeci separator
# niska.replace(stara, nova [, n]) - zamenjuje svako pojavljivanje niske stara
# niskom nova (ukoliko je zadat broj n, onda zamenjuje najvise n pojavljivanja)
```

1.3 Rešenja

Rešenje 1.6 Pogodi broj

```
# Pogodi broj

import random

print "-------- IGRA: Pogodi broj ---------\n"

zamisljen_broj = random.randint(0,100)

ime = raw_input("Unesite Vase ime: ")

print "Zdravo {0:s}. :) \nZamislio sam neki broj od 1 do 100. Da li mozes da pogodis
    koji je to broj? ".format(ime)

pogodio = 0;
while not pogodio:
     print "Unesi broj:"
     broj = int(raw_input())
     if broj == zamisljen_broj:
         pogodio = 1
     elif broj > zamisljen_broj:
         print "Broj koji sam zamislio je MANJI od {0:d}.".format(broj)
     else:
         print "Broj koji sam zamislio je VECI od {0:d}.".format(broj)

print "BRAVO!!! Pogodio si! Zamislio sam {0:d}. Bilo je lepo igrati se sa tobom. :)".
    format(zamisljen_broj)
```

Rešenje 1.7 Aproksimacija broja PI metodom Monte Karlo

```
# Aproksimacija broja PI metodom Monte Karlo


import random

print "Izracunavanje broja PI metodom Monte Karlo \n"
N = int(raw_input("Unesite broj iteracija: "))
# Broj tacaka koje se nalaze u krugu
A = 0
# Broj tacaka koje se nalaze u kvadratu
B = 0

i = N
while i >= 0:
    tacka = (random.random(), random.random())
    print "Tacka: "
    print tacka
    # Ukoliko se tacka nalazi u krugu, povecavamo broj tacaka u krugu
    if ((float(tacka[0])-0.5)**2 + (float(tacka[1])-0.5)**2) < (0.5**2):
        A = A + 1
    B = B + 1
    i = i - 1

print "Broj PI aproksimiran metodom Monte Karlo: "
print 4.0*A/B
```

Rešenje 1.8 X-O

```
# X-O
#
#  - | O | X
# --- --- ---
#  X | - | -
# --- --- ---
#  - | X | O

import random

def ispisi_tablu(tabla):
    print "\n    TABLA \n"
    print "    1   2   3   "
    print "   --- --- ---  "
    indeks = 1
    for i in tabla:
        print indeks,"|",i[0],"|",i[1],"|",i[2],"|"
        print "   --- --- --- "
        indeks = indeks + 1
    print "\n"

def pobedio(tabla):
    if (tabla[0][0] != "-" and tabla[0][2] != "-") and ((tabla[0][0] == tabla[1][1]
    == tabla[2][2]) or (tabla[0][2] == tabla[1][1] == tabla[2][0])):
        return True
    for i in range(3):
        if (tabla[0][i] != "-" and tabla[i][0] != "-") and ((tabla[0][i] == tabla[1][
    i] == tabla[2][i]) or (tabla[i][0] == tabla[i][1] == tabla[i][2])):
            return True
    return False

def ucitaj_koordinate(ime):
    while True:
        print "{0:s} unesite koordinate polja koje zelite da popunite u posebnim
    linijama:\n".format(ime)
        x = int(raw_input("Unesite vrstu: "))
        y = int(raw_input("Unesite kolonu: "))
        if 1<=x<=3 and 1<=y<=3:
            return x-1,y-1
        else:
            "Morate uneti brojeve 1,2 ili 3\n"

def korak(igrac):
    while True:
        x,y = ucitaj_koordinate(igrac[0])
        if tabla[x][y] == "-":
            tabla[x][y] = igrac[1]
            ispisi_tablu(tabla)
            break
        else:
            print tabla[x][y]
            print "Uneto polje je popunjeno!\n"

print "IGRA: X-O pocinje\n"

ime1 = raw_input("Unesite ime prvog igraca: ")
print "Zdravo {0:s}!\n".format(ime1)
ime2 = raw_input("Unesite ime drugog igraca: ")
print "Zdravo {0:s}!\n".format(ime2)

indikator = random.randint(1,2)
if indikator == 1:
    prvi_igrac = (ime1, "X")
    drugi_igrac = (ime2, "O")
else:
    prvi_igrac = (ime2, "X")
    drugi_igrac = (ime1, "O")

print "Igrac {0:s} igra prvi. \n".format(prvi_igrac)
print "X : {0:s}\n".format(prvi_igrac)
print "O : {0:s}\n".format(drugi_igrac)

tabla = [['-','-','-'],['-','-','-'],['-','-','-']]
```

```
71
print "Zapocnimo igru \n"
73
ispisi_tablu(tabla)
75
na_redu = 0
77 iteracija = 0
igraci = [prvi_igrac, drugi_igrac]
79 while iteracija < 9:
    korak(igraci[na_redu])
81     if pobedio(tabla) == True:
        print "BRAVO!!!!!!! {0:s} je pobedio!\n".format(igraci[na_redu][0])
83         break
    na_redu = (na_redu+1)%2
85     iteracija = iteracija + 1

87 if iteracija == 9:
    print "NERESENO! Pokusajte ponovo.\n"
```