Programski jezici

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

Univerzitet u Beogradu
Matematički fakultet

Programske paradigme

Materijali za vežbe

Nastavnik: Milena Vujošević Janičić
Asistenti: Milan Čugurović, Ivan Ristović

Beograd
2019.

Priprema materijala:
dr Milena Vujošević Janičić, docent na Matematičkom fakultetu u Beogradu
Marjana Šolajić, asistent na Matematičkom fakultetu u Beogradu
Branislava Živković
Nemanja Mićović, asistent na Matematičkom fakultetu u Beogradu
Milica Selaković, asistent na Matematičkom fakultetu u Beogradu
Milan Čugurović, asistent na Matematičkom fakultetu u Beogradu
Ivan Ristović, 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/

Merenje performansi programskih jezika:

- https://benchmarksgame-team.pages.debian.net/benchmarksgame/

1.1 Uvod, kolekcije, matematičke funkcije

1.1.1 Uvodni primeri

Zadatak 1.1 Napisati program koji na standardni izlaz ispisuje poruku *Hello world! :)*.

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

Zadatak 1.2 Napisati program koji za uneta dva cela broja i nisku ispisuje najpre unete vrednosti, a zatim i zbir brojeva, njihovu razliku, proizvod i količnik.

```
# 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

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

string_broj = raw_input("Unesite ceo broj: ")
b = int(string_broj)
```

```
22
niska = raw_input("Unesite nisku: ")
24
# Formatiran ispis
26 print "\n------Formatiran ispis------\n"
print "Brojevi: {0:d} {1:d}\nNiska: {2:s}\n".format(a,b,niska)
28
# Osnovne aritmeticke operacije:
30 # +, -, *, /, %, ** (stepenovanje), // (celobrojno deljenje)
zbir = a + b
32 razlika = a - b
proizvod = a * b
34 kolicnik = float(a) / b
stepen = a ** b
36 div = a // b

38 print "Zbir {0:d}\nRazlika {1:d}\nProizvod {2:d}\nKolicnik {3:f}\nStepen {4:d}\
    nCelobrojni kolicnik {5:d}"
    .format(zbir, razlika, proizvod, kolicnik, stepen, div)
```

Zadatak 1.3 Ako je prvi dan u mesecu ponedeljak napisati funkciju `radni_dan(dan)` koja kao argument dobija dan u mesecu i vraća tačno ako je dan radni dan. Napisati program koji testira ovu funkciju, korisnik sa standardnog ulaza u petlji unosi deset dana i dobija o poruku o tome da li su uneti dani radni ili ne.

```
1 # Funkcije
#
3 # def ime_funkcije(argumenti):
#     telo funkcije
5
def radni_dan(broj):
7     # Osnovne logicke operacije:
        # not, and, or
9
        if broj % 7 == 1 or broj % 7 == 2 or broj % 7 == 3:
11          return True
        elif broj % 7 == 4 or broj % 7 == 5:
13          return True
        # Naredbi <<elif>> moze biti vise
15      else:
            return False
17
# Blokovi se ne ogranicavaju viticastim zagradama kao sto je u C-u
19 # vec moraju biti indentovani. Naredbe na istom nivou indentacije se
# smatraju istim naredbama istog bloka
21
i = 0
23 while i <= 10:
    dan = raw_input("Unesite dan")
25  dan = int(dan)
    if radni_dan(dan):
27      print "Uneti dan je radni dan"
    else:
29      print "Uneti dan je neradan"
    i = i + 1 # i++ ne postoji, moze ili ovako ili i += 1
31
# Naredba <<break>> iskace iz bloka, isto kao i u C-u
33 # Naredba <<pass>> je ista kao naredba <<continue>> u C-u
```

Zadatak 1.4 Napisati program koji na standardni izlaz ispisuje vrednost $6!$, $\log_5 125$ i pseudo slučajan broj iz opsega $[0, 1)$

```
1 # Matematicke funkcije

3 # Ukljucujemo modul <<math>>
import math
5
# U ovom moduli se nalaze brojne funkcije kao sto su:
7 #
# math.sqrt(broj)
9 # 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>>
import random

# Funkcija random() vraca pseudo slucajan broj tipa float iz opsega [0.0, 1.0)
print "\n------Pseudo slucajni brojevi------\n"
print "Pseudo slucajan broj iz opsega [0.0,1.0)\n"
print random.random()

# Korisne funkcije:
#
# randint(a,b) - vraca pseudo slucajan ceo broj n iz opsega [a,b]
# choice(lista) - vraca pseudo slucajan element iz liste
```

Zadatak 1.5 Napisati program koji imitira rad bafera. Maksimalni broj elemenata u baferu je 5. Korisnik sa standardnog ulaza unosi podatke do unosa reči *quit*. Program ih smešta u bafer, posto se bafer napuni unosi se ispisuju na standarni izlaz i bafer se prazni.

```
# LISTA
#
# Notacija: [element1, element2, ...]
#
# Liste mogu sadrzati razlicite tipove podataka
# lista = [1,2,3.4,"Another brick in the wall",True,[5,False,4.4,'Layla']]

# Prazna lista
buffer = []
i = 0
while True:
    unos = raw_input()
    if unos == 'quit':
        break
    # Ubacivanje elementa na kraj
    buffer.append(unos)
    i += 1
    if i == 5:
        # Prolazak kroz listu
        for unos in buffer:
            print unos
        # Praznimo bafer
        buffer = []
        i = 0
```

Zadatak 1.6 Korisnik sa standardnog ulaza unosi ceo broj n, a potom ciklično pomeren rastuće sortiran niz (pr. 56781234) koji ima n elemenata. Napisati program koji na standarni izlaz ispisuje sortiran niz bez ponavljanja elementa.

```
# Korisne funkcije za liste:
#
# list.remove(x) - izbacuje prvo pojavljivanje elementa x iz liste
# list.count(x) - vraca broj koliko puta se element x nalazi u listi
# list.index(x) - vraca indeks prvog pojavljivanja elementa x u listi
# len(lista) - vraca broj elemenata liste
# del lista[a:b] - brise elemente liste od pozicije a do b

n = int(raw_input("Unesite broj elemenata"))

elementi = []
# Funkcija <<range>>
#
# range(kraj)
# range(pocetak, kraj[, korak])
```

```
16
18  for i in range(n):
        element = int(raw_input())
20      # Provera da li se element nalazi u listi
        if not element in elementi:
22          # Ubacivanje elementa u listi
            elementi.append(element)

24  k = 0
    for i in range(n-1):
26      # Pristupanje elementima liste
        if elementi[i] > elementi[i+1]:
28          k = i + 1
            break
30
    prvi_deo = elementi[0:k]
32  drugi_deo = elementi[k:]

34  # Nadovezvivanje dve liste
    sortirani = drugi_deo + prvi_deo
36  print "Sortirani elementi"
    for element in sortirani:
38      print element
```

Zadatak 1.7 Napisati funkciju `max_list(lista)` koja vraća najveći element u listi listi. Napisati program koji testira ovu funkciju.

```
def max_list(lista):
2       # Mozemo indeksirati liste unazad, pozicija -1 odgovara poslednjem elementu
        maximum = lista[-1]
4       for element in lista:
            if maximum < element:
6               maximum = element
        return maximum
8
    lista = [[1,2,3], [1,2,5], ['abc','abc','abc'], ['abc', 'ab', 'abcd'], ['a','b','c']
        > ['a', 'b']]
10  print max_list(lista)
```

Zadatak 1.8 Napisati program za rad sa stekom.

- Definisati stek koji sadrži elemente $9, 8, 7$
- Dodati na stek elemente 6 i 5
- Skinuti sa steka element i ispisati ga na standardni izlaz

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

Zadatak 1.9 Napisati program koji za uneti prirodan broj n ispisuje vrednosti funkcije x^2 u celobrojnim tačkama u intervalu $[0, n]$. Zadatak rešiti korišćenjem mape.

```
# KATALOG
2   #
    # Katalog je kolekcija uredjenih parova oblika (kljuc, vrednost)
4   #
    # Notacija: {kljuc:vrednost, kljuc:vrednost, ...}
6
```

```
mapa = {} # prazna mapa

n = int(raw_input("Unesite n: "))

for x in range(n):
    mapa[x] = x**2

# Prolazak kroz mapu
print "\n------Prolazak kroz katalog------\n"
for kljuc in mapa:
    print "f({0:s}) =  {1:s}\n".format(str(kljuc),str(mapa[kljuc]))

# Korisne funkcije
#
# map.keys() - vraca listu kljuceva iz kataloga
# map.values() - vraca listu vrednosti iz kataloga
# map.has_key(kljuc) - vraca True/False u zavisnosti od toga da li se element
# sa kljucem kljuc nalazi u katalogu
```

Zadatak 1.10 Sa standardnog ulaza se unose reči do reči *quit*. Napisati program koji ispisuje unete reči eliminišući duplikate.

```
lista = []

while True:
    unos = raw_input()
    if unos == "quit":
        break
    lista.append(unos)

# Pravljenje skupa od liste
skup = set(lista)

for i in skup:
    print i
```

Zadatak 1.11 Napisati funkciju `min_torka(lista)` koja vraća najmanji element u torci torki. Napisati program koji ovu funkciju testira.

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

def min_torka(torke):
    # Pristupanje elementima u torki
    minimum = torke[0]
    for torka in torke:
        # Ukoliko torke ne sadrze elemente istog tipa na istim pozicijama, i dalje ih
     mozemo porediti,
        # ali poredjenje se vrsi na osnovu imena tipa elementa leksikografski
        # npr.  element tipa List < element tipa String < element tipa Tuple i slicno
        if minimum > torka:
            minimum = torka
    return minimum

torke = ((1,2,'a'),(1,2,'b'),([1,2,3], 'Bugs', 4), ([1,1,1], 'Bunny', 6),(1,2,['a','b
    ']),(1,2,'ab'))

print min_torka(torke)
```

1.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.12 **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.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
POKRETANJE: pogodi_broj
 -------- IGRA: Pogodi broj ---------
 Unesite Vase ime:
 Milica
 Zdravo Milica. :)
 Zamislio sam neki broj od 1 do 100. Da li mozes da pogodis koji je to broj?
 Unesi broj
 50
 Broj koji sam zamislio je MANJI od 50
 Unesi broj
 25
 Broj koji sam zamislio je VECI od 25
 Unesi broj
 30
 Broj koji sam zamislio je MANJI od 30
 Unesi broj
 27
 "BRAVO!!! Pogodio si! Zamislio sam 27. Bilo je lepo igrati se sa tobom. :)
```

[Rešenje 1.12]

Zadatak 1.13 **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 $(0, 0), (0, 1), (1, 1), (1, 0)$.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
POKRETANJE: aproksimacija_PI
 Izracunavanje broja PI metodom Monte Karlo
 Unesite broj iteracija:
 5
 Tacka:
 (0.14186247318019474, 0.15644650897353152)
 Tacka:
 (0.8910898038304426, 0.2200563958299553)
 Tacka:
 (0.641604107090444, 0.03712366524007682)
 Tacka:
 (0.4893727376942526, 0.17230005349431587)
 Tacka:
 (0.6199558112390107, 0.32122922953511124)
 Tacka:
 (0.5821041171248978, 0.025052299437462566)
 Broj PI aproksimiran metodom Monte Karlo: 4.0
```

[Rešenje 1.13]

Zadatak 1.14 **X-O** Napisati program koji implementira igricu X-O sa dva igrača.

Primer 1

```
Pokretanje: XO
Interakcija sa programom:
 IGRA: X-O pocinje
 Unesite ime prvog igraca:
 Ana
 Zdravo Ana
 Unesite ime drugog igraca:
 Petar
 Zdravo Petar!
 Igrac ('Ana', 'X') igra prvi.
 X : ('Ana', 'X')
 O : ('Petar', 'O')
 Zapocnimo igru
 TABLA
  1 2 3
  --- --- ---
 1 | - | - | - |
  --- --- ---
 2 | - | - | - |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Ana unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 1
 Unesite kolonu:
 1
 TABLA
  1 2 3
  --- --- ---
 1 | X | - | - |
  --- --- ---
 2 | - | - | - |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Petar unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 1
 Unesite kolonu:
 2
 TABLA
  1 2 3
  --- --- ---
 1 | X | O | - |
  --- --- ---
 2 | - | - | - |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Ana unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 2
 Unesite kolonu:
 2
 TABLA
  1 2 3
  --- --- ---
 1 | X | O | - |
  --- --- ---
 2 | - | X | - |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Petar unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 2
 Unesite kolonu:
 3
 TABLA
  1 2 3
  --- --- ---
 1 | X | O | - |
  --- --- ---
 2 | - | X | O |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Ana unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 3
 Unesite kolonu:
 3
 TABLA
  1 2 3
  --- --- ---
 1 | X | O | - |
  --- --- ---
 2 | - | X | O |
  --- --- ---
 3 | - | - | X |
  --- --- ---
 BRAVO!!!!!!! Igrac Ana je pobedio!
```

[Rešenje 1.14]

1.1.3 Zadaci za vežbu

Zadatak 1.15 **Ajnc** Napisati program koji implementira igricu Ajnc 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. Detaljan opis igre može se naći na narednoj adresi: https://en.wikipedia.org/wiki/Blackjack

Primer 1

```
POKRETANJE: ajnc
INTERAKCIJA SA PROGRAMOM:
 -------- IGRA: Ajnc ---------
 Unesite Vase ime:
 Pavle
 Zdravo Pavle. :)
 Igra pocinje
 Vase karte su:
 1 Herc 5 karo
 Hit ili stand?[H/S]
 H
 Vase karte su:
 1 Herc 5 karo 5 tref
 Cestitam!!! Pobedio si!
 Bilo je lepo igrati se sa tobom. :)
```

Primer 2

```
POKRETANJE: ajnc
INTERAKCIJA SA PROGRAMOM:
 -------- IGRA: Ajnc ---------
 Unesite Vase ime:
 Pavle
 Zdravo Pavle. :)
 Igra pocinje
 Vase karte su:
 Q Tref 7 karo
 Hit ili stand?[H/S]
 H
 Vase karte su:
 Q Tref 7 karo K Herc
 Zao mi je, izgubio si!:(
 Bilo je lepo igrati se sa tobom. :)
```

Zadatak 1.16 **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. Detaljan opis igre može se naći na narednoj adresi: https://en.wikipedia.org/wiki/Connect_Four

Primer 1

```
POKRETANJE: cetri_u_nizu
INTERAKCIJA SA PROGRAMOM:
 IGRA: Cetiri u nizu pocinje
 Unesite ime prvog igraca:
 Ana
 Zdravo Ana
 Unesite ime drugog igraca:
 Petar
 Zdravo Petar!
 Igrac ('Ana', 'C') igra prvi.
 C : ('Ana', 'C')
 Z : ('Petar', 'Z')
 Zapocnimo igru
 TABLA
  1 2 3 4 5 6 7 8
  --- --- --- --- --- --- --- ---
 1 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 2 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 3 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 4 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 5 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 6 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 7 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 8 | - | - | - | - | - | - | - | - |
 Ana unesite koordinate polja
 koje zelite da popunite
 u posebnim linijama:
 Unesite vrstu:
 1
 Unesite kolonu:
 1
 TABLA
  1 2 3 4 5 6 7 8
  --- --- --- --- --- --- --- ---
 1 | c | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 2 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 3 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 4 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 5 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 6 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 7 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 8 | - | - | - | - | - | - | - | - |
 Petar unesite koordinate polja
 koje zelite da popunite
 u posebnim linijama:
 Unesite vrstu:
 2
 Unesite kolonu:
 1
 TABLA
  1 2 3 4 5 6 7 8
  --- --- --- --- --- --- --- ---
 1 | c | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 2 | z | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 3 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 4 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 5 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 6 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 7 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 8 | - | - | - | - | - | - | - | - |
```

```
Ana unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
1
Unesite kolonu:
2
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
Petar unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
2
Unesite kolonu:
2
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | z | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
Ana unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
1
Unesite kolonu:
3
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | c | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | z | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
```

```
Petar unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
2
Unesite kolonu:
3
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | c | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | z | z | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
Ana unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
1
Unesite kolonu:
4
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | c | c | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | z | z | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
BRAVO!!!!!!! Igrac Ana je pobedio!
```

1.2 Datoteke, niske, JSON format, datum

1.2.1 Uvodni primeri

Zadatak 1.17 Korisnik na standardni ulaz unosi dve niske. Napisati program koji prvo pojavljivanje druge niske u prvoj zamenjuje karakterom $. U slučaju da nema pojavljivanja druge niske u prvoj i da je druga niska kraća ispisuje nadovezane niske sa separatorom -. Ako je druga niska duža od prve program treba da ispiše drugu nisku i njenu dužinu.

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

niska1 = raw_input("Unesite nisku: ")
niska2 = raw_input("Unesite nisku: ")

# Duzinu niske racunamo koristeci funkciju len(niska)
if len(niska1) >= len(niska2):
    # Funkcija count
    # niska.count(podniska [, pocetak [, kraj]]) - vraca broj koliko se puta
    # podniska nalazi u niski (u intervalu od pocetak do kraj)
    n = niska1.count(niska2)
    if n > 0:
        # 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
        i = niska1.find(niska2)
        # Karakterima u niski mozemo pristupati koristeci notaciju [] kao kod listi
        niska1 = niska1[0 : i] + '$' + niska1[i+len(niska2) : ]
        print niska1
    else:
        # Funkcija join
        # niska_separator.join([niska1,niska2,niska3,...]) - spaja listu niski
    separatorom
        print '-'.join([niska1,niska2])
else:
    print "Duznina niske {0:s} je {1:d}".format(niska2, len(niska2))

# 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)
```

Zadatak 1.18 Napisati program koji ispisuje tekući dan u nedelji, dan, mesec i vreme u formatu *hh:mm:ss*.

```
# Datumi
# Ukljucujemo klasu datetime iz modula datetime
from datetime import datetime

# Nov objekat datuma:
# datetime(godina, mesec, dan [, sat [, minut [, sekund]]])
# Korisne funkcije:
# datetime.now() - vraca trenutno vreme odnosno datum
# datetime.year, datetime.month, datetime.day, datetime.hour, datetime.minute,
    datetime.second,
# datetime.strftime(format) - vraca string reprezentaciju objekta datuma na osnovu
    zadatog formata
# datetime.strptime(niska, format) - vraca objekat datetime konstruisan na osnovu
    niske u zadatom formatu
# datetime.time([sat [, minut [, sekund]]]) - vraca objekat koji predstavlja vreme
```

```
# datetime.date(dan, mesec, godina) - vraca objekat datuma
# format:
#   %A - dan u nedelji (Monday, Tuesday,...)
#   %w - dan u nedelji (0, 1, 2,..., 6)
#   %d - dan (01, 02, 03,...)
#   %B - mesec (January, February,...) %b (Jan, Feb, ...)
#   %m - mesec (01, 02, ...)
#   %Y - godina (1992, 1993,...) %y (92, 93, ...)
#   %H - sat (00, 01, ..., 23)
#   %M - minut (00, 01, ..., 59)
#   %S - sekund (00, 01, ..., 59)

print "\n-----Datumi-----\n"
print datetime.now().strftime("Dan u nedelji: %A/%w, Dan: %d, Mesec: %B/%m, Godina: %
    Y, Vreme: %H:%M:%S\n")
print datetime.now().time()
print datetime.now().date()
```

Zadatak 1.19 Napisati program koji ispisuje sadrzaj datoteka *datoteka.txt* na standardni izlaz karakter po karakter.

```
# Datoteku otvaramo koristeci funkciju open koja vraca
# referencu na otvoreni tok podataka.
#
# rezimi:
# - "r"  -> read
# - "w"  -> write
# - "a"  -> append
# - "r+" -> read + append
#
# Datoteku smo duzni da zatvorimo sa 'datoteka.close()',
#
# datoteka = open("datoteka.txt","r")
# kod koji obradjuje datoteku
# ...
# datoteka.close()

# Python nudi 'with' koji omogucava da
# se datoteka automatski zatvori, cak i u situaciji
# kada se ispali izuzetak. Ovo je preporuceni nacin
# za citanje tokova podataka u Python-u.
with open("datoteka.txt", "r") as datoteka:
    # Citamo datoteku liniju po liniju, a potom
    # u liniji citamo karakter po karakter.
    for linija in datoteka:
        for karakter in linija:
            print karakter
    # datoteka.close() nam nije neophodno,
    # Python ce sam zatvoriti datoteku kada
    # se zavrsi 'with' blok
```

Zadatak 1.20 Napisati program koji ispisuje sadrzaj datoteka *datoteka.txt* na standardni izlaz liniju po liniju.

```
# Ispitivanje da li je otvaranje datoteke uspelo:
try:
    with open("datoteka.txt","r") as g:
        # Liniju po liniju mozemo ucitavati koristeci petlju
        # tako sto 'iteriramo' kroz Datoteku
        print "-----Iteriranje kroz datoteku <<for>> petljom-----\n"
        # Metod f.readline() cita jednu liniju iz Datoteke
        for linija in g:
            print linija
except IOError:
    # Ukoliko ne uspe otvaranje datoteke, Python ispaljuje
    # izuzetak 'IOError'.
    print "Nije uspelo otvaranje datoteke."
```

Zadatak 1.21 Napisati program koji dodaje u datoteku *datoteka.txt* nisku *water* a potom ispisuje njen sadržaj na standardni izlaz.

```
# f.readlines() i list(f)
# vracaju listu linija datoteke
#
# f.write(niska) upisuje nisku u datoteku
print "-----Upisivanje u datoteku-----\n"
# razresiti konfuziju izmedju a+ (upisuje na kraj) i r+ (upisuje na pocetak, tj.
    prepisace postojeci sadrzaj)
with open("datoteka.txt","a+") as h:
    h.write("water\n")
    # h.flush() - da budemo sigurni da je prebaceno iz bafera
    # nakon write file pointer se pomerio pa da bismo procitali ceo sadrzaj vracamo
    se na pocetak datoteke
    h.seek(0,0)
    print h.readlines()
```

Zadatak 1.22 Korisnik na standarni ulaz unosi podatke o imenu, prezimenu i godinama. Program potom kreira JSON objekat *junak*, koji ima podatke *Ime*, *Prezime* i *Godine*, i ispisuje ga na standardni izlaz, a potom i u datoteku *datoteka.txt*.

```
# JSON format
#
# Funkcije za rad sa JSON formatom se nalaze u modulu json
import json

ime = raw_input("Unesite ime: ")
prezime = raw_input("Unesite prezime: ")
godine = int(raw_input("Unesite godine: "))

# json.dumps(objekat) vraca string koji sadrzi JSON reprezentaciju objekta x

print "\n-----JSON reprezentacija objekta-----\n"
junak = {"Ime": ime, "Prezime":prezime, "Godine":godine}
print json.dumps(junak)

# json.dump(x,f) upisuje string sa JSON reprezentacijom objekta x u datoteku f

f = open("datoteka.json","w")
json.dump(junak,f)
f.close()
```

JSON

JSON (https://www.json.org/) reprezentacija objekata predstavlja jednostavni i pregledan način za serijalizaciju objekata. Koristi se svuda, pre svega u web programiranju kada je potrebno proslediti objekte preko mreže ali i lokalno kada je na primer potrebno sačuvati neki objekat u bazi podataka. JSON je nezamenjiv kod takozvanih *RESTful servisa* (servisa zasnovanih na REST-u, videti https://en.wikipedia.org/wiki/Representational_state_transfer). Korisnici obično pošalju zahtev preko HTTP protokola, a servis odgovara JSON reprezentacijom objekta koji ima informacije koje je korisnik tražio.

Zadatak 1.23 Napisati program koji iz datoteke *datoteka.txt* učitava JSON objekat, a potom na standardni izlaz ispisuje podatke o *imenu*, *prezimenu* i *godininama*.

```
# json.load(f) ucitava iz datoteke string koji sadrzi
# JSON format objekta i vraca referencu na mapu koja
# je konstruisana na osnovu .json datoteke.
print "\n-----Ucitavanje objekta iz datoteke-----\n"
f = open("dat4.json","r")
x = json.load(f)
print x['Ime']
print x['Prezime']
print x['Godine']
f.close()
```

Zadatak 1.24 Napisati program koji od RESTful servisa https://quotes.rest uzima citat dana pomoću GET zahteva na https://quotes.rest/qod i ispisuje citat na standardni izlaz. Napomena: Potrebno je instalirati modul *requests* kako bi kod ispod radio (`pip install requests`,

pip je Python modul menadžer, ukoliko nije instaliran može se instalirati pomoću `apt-get install python-pip` ili bilo kod drugog package-managera).

```
import requests

url = 'https://quotes.rest/qod'
response = requests.get(url)

if response.ok:
    json = response.json()
    print json
    citat = json['contents']['quotes'][0]['quote']
    print "---------------"
    print citat
else:
    print "Neuspesno dovlacenje citata."
```

1.2.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.25 Napisati program koji sa standardnog ulaza učitava ime datoteke i broj n i računa broj pojavljivanja svakog n-grama u datoteci koji su sačinjeni od proizvoljnih karaktera i rezultat upisuje u datoteku `rezultat.json`.

Primer 1

```
POKRETANJE: python n-anagram.py
INTERAKCIJA SA PROGRAMOM:
 Unesite ime datoteke:
 datoteka.txt
 Unesite n
 2
```

Sadržaj datoteka koje se koriste u primeru 1.25:

Listing 1.1: *datoteka.txt*

```
Ovo je datoteka dat
```

Listing 1.2: *rezultat.json*

```
{
'a ': 1, 'ka': 1, 'ot': 1, 'ek': 1,
' d': 2, ' j': 1, 'da': 2, 'e ': 1,
'o ': 1, 'to': 1, 'at': 2, 'je': 1,
'Ov': 1, 'te': 1, 'vo': 1
}
```

[Rešenje 1.25]

Zadatak 1.26

U datoteci `korpa.json` se nalazi spisak kupljenog voća u json formatu:

```
[ { 'ime' : ime_voca, 'kolicina' : broj_kilograma } , ...]
```

U datotekama `maxi_cene.json`, `idea_cene.json`, `shopngo_cene.json` se nalaze cene voća u json formatu:

```
[ { 'ime' : ime_voca, 'cena' : cena_po_kilogramu } , ...]
```

Napisati program koji izračunava ukupan račun korpe u svakoj prodavnici i ispisuje cene na standardni izlaz.

Primer 1

```
POKRETANJE: python korpa.py
INTERAKCIJA SA PROGRAMOM:
 Maxi: 631.67 dinara
 Idea: 575.67 dinara
 Shopngo: 674.67 dinara
```

[Rešenje 1.26]

Sadržaj datoteka koje se koriste u primeru 1.26:

Listing 1.3: *korpa.json*

```
[ {"ime" : "jabuke" ,"kolicina": 3.3},
{"ime": "kruske" , "kolicina": 2.1},
{"ime": "grozdje" , "kolicina":  2.6},
```

Listing 1.4: *maksi_cene.json*

```
[ {"ime" :"jabuke", "cena" : 59.9},
{"ime" :"kruske", "cena" : 120},
{"ime" :"grozdje", "cena" : 70},
{"ime" :"narandze", "cena" : 49.9},
{"ime" :"breskve", "cena" : 89.9} ]
```

Listing 1.5: *idea_cene.json*

```
[ {"ime" :"jabuke", "cena" : 39.9},
{"ime" :"kruske", "cena" : 100},
{"ime" :"grozdje", "cena" : 90},
{"ime" :"breskve", "cena" : 59.9} ]
```

Listing 1.6: *shopngo_cene.json*

```
[ {"ime" :"jabuke", "cena" : 69.9},
{"ime" :"kruske", "cena" : 100},
{"ime" :"grozdje", "cena" : 90},
{"ime" :"maline", "cena" : 290},
```

[Rešenje 1.26]

1.2.3 Zadaci za vežbu

Zadatak 1.27 Napisati program koji iz datoteke `ispiti.json` učitava podatke o ispitima i njihovim datumima. Ispisati na standardni izlaz za svaki ispit njegovo ime i status "Prosao" ukoliko je ispit prosao, odnosno "Ostalo je jos n dana.", gde je n broj dana od trenutnog datuma do datuma ispita.

Primer 1

```
POKRETANJE: python ispiti.py
INTERAKCIJA SA PROGRAMOM:
 Relacione baze podataka Prosao
 Vestacka inteligencija Prosao
 Linearna algebra i analiticka geometrija Prosao
```

Sadržaj datoteka koje se koriste u primeru 1.27:

Listing 1.7: *ispiti.json*

```
[  {'ime': 'Relacione baze podataka',
'datum': '21.09.2016.'},
{'ime': 'Vestacka inteligencija',
'datum': '17.06.2017.'},
{'ime': 'Linearna algebra i analiticka geometrija',
'datum': '08.02.2017.'} ]
```

Zadatak 1.28 Napisati program koji izdvaja sve jednolinijske i višelinijske komentare iz .c datoteke čije ime se unosi sa standardnog ulaza, listu jednih i drugih komentara upisuje u datoteku `komentari.json`. Jednolinijski komentari se navode nakon // a višelinijski između /∗ i ∗/.

Primer 1

```
POKRETANJE: python komentari.py
INTERAKCIJA SA PROGRAMOM:
 Unesite ime datoteke:
 program.c
```

Sadržaj datoteka koje se koriste u primeru 1.28:

Listing 1.8: *program.c*

```c
#include <stdio.h>

// Primer jednolinijskog komentara

int main(){
/*
Na ovaj nacin ispisujemo tekst
na standardni izlaz koristeci jezik C.
*/
printf("Hello world!");

// Na ovaj nacin se ispisuje novi red
printf("\n");
/*
Ukoliko se funkcija uspesno zavrsila
vracamo 0 kao njen rezultat.
*/
return 0;
}
```

Listing 1.9: *komentari.json*

```
{
  'jednolinijski' : ['Primer jednolinijskog komentara',
                     'Na ovaj nacin se ispisuje novi red'],
  'viselinijski' : ['Na ovaj nacin ispisujemo tekst na standardni
                     izlaz koristeci jezik C.',
                     'Ukoliko se funkcija uspesno zavrsila
                     vracamo 0 kao njen rezultat.']
}
```

Zadatak 1.29 Napisati program upoređuje dve datoteke čija imena se unose sa standardnog ulaza. Rezultat upoređivanja je datoteka `razlike.json` koja sadrži broj linija iz prve datoteke koje se ne nalaze u drugoj datoteci i obratno. *Napomena* Obratiti pažnju na efikasnost.

Primer 1

```
POKRETANJE: python razlika.py
INTERAKCIJA SA PROGRAMOM:
 Unesite ime datoteke:
 dat1.txt
 Unesite ime datoteke:
 dat2.txt
```

Sadržaj datoteka koje se koriste u primeru 1.29:

Listing 1.10: *dat1.txt*

```
//netacno

same=1;

for(i=0;s1[i]!='\0' && s2[i]!='\0';i++) {
if(s1[i]!=s2[i]) {
```

```
same=0;
break;
}
}
return same;
```

Listing 1.11: *dat2.txt*

```
//tacno

for(i=0;s1[i]!='\0' && s2[i]!='\0';i++){

if(s1[i]!=s2[i])
return 0;
}
return s1[i]==s2[i];
```

Listing 1.12: *razlike.json*

```
{
  'dat1.txt' : 7,
  'dat2.txt' : 4
}
```

1.3 Argumenti komandne linije, sortiranje, obilazak direktorijuma

1.3.1 Uvodni primeri

Zadatak 1.30 Napisati program koji na standarni izlaz ispisuje argumente komandne linije.

```
# modul sys ima definisan objekat argv koji predstavlja listu argumenata komandne
    linije (svi argumenti se cuvaju kao niske karaktera)

import sys

if len(sys.argv) == 1:
    print "Niste naveli argumente komandne linije"
    # funkcija exit() iz modula sys prekida program
    # (ukoliko se ne prosledi argument, podrazumevano
    # se salje None objekat)
    exit()

# ispisujemo argumente komandne linije
# prvi argument, tj. sys.argv[0] je uvek ime skript fajla koji se pokrece
for item in sys.argv:
    print item
```

Zadatak 1.31 Napisati program koji na standrani izlaz ispisuje oznaku za tekući i roditeljski direktorijum, kao i separator koji se koristi za pravljenje putanje.

```
import os

print os.curdir     # oznaka tekuceg direktorijuma
print os.pardir     # oznaka za roditeljski direktorijum tekuceg direktorijuma
print os.sep        # ispisuje separator koji koristi za pravljenje putanja
```

Zadatak 1.32 Napisati program koji imitira rad komande *ls*. Program na standardni izlaz ispisuje sadržaj tekućeg direktorijuma.

```
import os

# funkcija za prosledjenu putanju direktorijuma vraca listu imena
```

```
# svih fajlova u tom direktorijumu, . je zamena za putanju tekuceg direktorijuma
print os.listdir(os.curdir)

# os.walk() - vraca listu torki (trenutni_direktorijum, poddirektorijumi, datoteke)
# os.path.join(putanja, ime) - pravi putanju tako sto nadovezuje na
#   prosledjenu putanju zadato ime odvojeno /

print "\n-----Prolazak kroz zadati direktorijum-----\n"
for (trenutni_dir, poddirektorijumi, datoteke) in os.walk(os.curdir):
    print trenutni_dir
    for datoteka in datoteke:
        print os.path.join(trenutni_dir, datoteka)

# os.path.abspath(path) - vraca apsolutnu putanju za zadatu relativnu putanju nekog
    fajla
# os.path.isdir(path) - vraca True ako je path putanja direktorijuma, inace vraca
    False
# os.path.isfile(path) - vraca True ako je path putanja regularnog fajla, inace vraca
     False
```

Zadatak 1.33 Napisati program koji na standarni izlaz ispisuje sve apsolutne putanje regularnih fajlova koji se nalaze u tekućem direktorijumu.

```
import os

print "\n-----Regularni fajlovi zadatog direktorijuma-----\n"
for ime in os.listdir(os.curdir):
    # Funkcija 'join' vrsi konkatenaciju putanja koristeci sistemski separator
    if os.path.isfile(os.path.join(os.curdir, ime)):
        print os.path.abspath(os.path.join(os.curdir, ime))
```

Zadatak 1.34 U datoteci *tacke.json* se nalaze podaci o tačkama u sledećem formatu.

Listing 1.13: *tacke.json*

```
[ {"teme":"A" , "koordinate": [10.0, 1.1]},
 {"teme":"B" , "koordinate": [1.0, 15.0]},
 {"teme":"C" , "koordinate": [-1.0, 5.0]} ]
```

Napisati program koji učitava podatke o tačkama iz datoteke *tacke.json* i sortira i po udaljenosti od koordinatnog početka. Na standarni izlaz ispisati podatke pre i posle sortiranja.

```
# Sortiranje
#
# sorted(kolekcija [, poredi [, kljuc [, obrni]]]) - vraca sortiranu kolekciju
#
# kolekcija - kolekcija koju zelimo da sortiramo
# poredi - funkcija poredjenja
# kljuc - funkcija koja vraca kljuc po kome se poredi
# obrni - True/False (opadajuce/rastuce)
#
# za poziv sorted(kolekcija) koristi se funkcija cmp za poredjenje
# cmp(x, y) -> integer
# vraca negativnu vrednost za x<y, 0 za x==y, pozitivnu vrednost za x>y
# ako su x i y niske, cmp ih leksikografski poredi

import json
import math

# l = ["A", "C", "D", "5", "1", "3"]
# print l
# print "sortirana lista: ", sorted(l)

# u sledecem primeru je neophodno da definisemo svoje funkcije za poredjenje i
    vracanje kljuca jer je kolekcija lista recnika i za to cmp nema definisano
    ponasanje
with open("tacke.json","r") as f:
    tacke = json.load(f)

# funkcija koja tacke x i y poredi po njihovoj udaljenosti od koordinatnog pocetka
def poredi(x,y):
```

```
    if (x[0]*x[0] + x[1]*x[1]) > (y[0]*y[0] + y[1]*y[1]):
29      return 1
    else:
31      return -1
# funkcija kljuc kao argument ima element kolekcije koja se poredi, u ovom slucaju je
     to jedan recnik
33 # povratna vrednost funkcije kljuc je u stvari tip argumenata funkcije poredi
def kljuc(x):
35      return x["koordinate"]

37 sortirane_tacke = sorted(tacke, poredi, kljuc) # ili sorted(tacke, poredi, kljuc,
    True) ako zelimo opadajuce da se sortira
print "Tacke pre sortiranja:"
39 for item in tacke:
        print item["teme"],
41 print "\nTacke nakon sortiranja: "
for item in sortirane_tacke:
43      print item["teme"],
print
```

1.3.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.35 Napisati program koji računa odnos kardinalnosti skupova duže i šire za zadati direktorijum. Datoteka pripada skupu duže ukoliko ima više redova od maksimalnog broja karaktera po redu, u suprotnom pripada skupu šire. Sa standardnog ulaza se unosi putanja do direktorijuma. Potrebno je obići sve datoteke u zadatom direktorijumu i njegovim poddirektorijumima (koristiti funkciju os.walk()) i ispisati odnos kardinalnosti skupova duže i šire.

Primer 1

```
POKRETANJE: python duze_sire.py
INTERAKCIJA SA PROGRAMOM:
 Unesite putanju do direktorijuma:
 ..
 Kardinalnost skupa duze : Kardinalnost skupa sire
 10 : 15
```

[Rešenje 1.35]

Zadatak 1.36 Napisati program koji obilazi direktorijume rekurzivno i računa broj datoteka za sve postojeće ekstenzije u tim direktorijumima. Sa standardnog ulaza se unosi putanja do početnog direktorijuma, a rezultat se ispisuje u datoteku `rezultat.json`.

Primer 1

```
POKRETANJE: python ekstenzije.py
INTERAKCIJA SA PROGRAMOM:
 Unesite putanju do direktorijuma:
 .
```

Sadržaj datoteka koje se koriste u primeru 1.36:

Listing 1.14: *rezultat.txt*

```
1 {
2  'txt' : 14,
3  'py' : 12,
4  'c' : 10
5 }
```

[Rešenje 1.36]

Zadatak 1.37 U datoteci `radnici.json` nalaze se podaci o radnom vremenu zaposlenih u preduzeću u sledecem formatu:

```
1    [  { 'ime' : ime\_radnika, 'odmor' : [pocetak, kraj], 'radno_vreme'
     : [pocetak, kraj] }, ...]
```

Napisati program koji u zavisnosti od unete opcije poslodavcu ispisuje trenutno dostupne radnike odnosno radnike koji su na odmoru. Moguće opcije su 'd' - trenutno dostupni radnici i 'o' - radnici koji su na odmoru. Radnik je dostupan ukoliko nije na odmoru i trenutno vreme je u okviru njegovog radnog vremena.

Primer 1

```
POKRETANJE: python odmor.py
INTERAKCIJA SA PROGRAMOM:
 "Unesite opciju koju zelite
 d - dostupni radnici
 o - radnici na odmoru :
 m
 Uneta opcija nije podrzana
```

Primer 2

```
POKRETANJE: python odmor.py
INTERAKCIJA SA PROGRAMOM:
 "Unesite opciju koju zelite
 d - dostupni radnici
 o - radnici na odmoru :
 d
 Pera Peric
```

[Rešenje 1.37]

Sadržaj datoteka koje se koriste u primeru 1.37:

Listing 1.15: *radnici.json*

```
1 [  { 'ime' : 'Pera Peric',
2 'odmor' : ['21.08.2016.', '31.08.2016.'],
3 'radno_vreme' : ['08:30', '15:30'] } ]
```

Zadatak 1.38 Napisati program koji učitava ime datoteke sa standardnog ulaza i na standardni izlaz ispisuje putanje do svih direktorijuma u kojima se nalazi ta datoteka.

Primer 1

```
POKRETANJE: python pojavljivanja.py
INTERAKCIJA SA PROGRAMOM:
 Unesite ime datoteke:
 1.py
 /home/student/vezbe/cas1/1.py
 /home/student/vezbe/cas7/1.py
 /home/student/vezbe/cas9/1.py
```

[Rešenje 1.40]

1.3.3 Zadaci za vežbu

Zadatak 1.39 Napisati program koji ispisuje na standardni izlaz putanje do lokacija svih Apache virtuelnih hostova na računaru. Smatrati da je neki direktorijum lokacija Apache virtuelnog hosta ukoliko u sebi sadrži `index.html` ili `index.php` datoteku.

Primer 1

```
POKRETANJE: python apache.py
INTERAKCIJA SA PROGRAMOM:
 /home/student/PVEB/prviPrimer
 /home/student/licna_strana
 /home/student/PVEB/ispit/jun
```

Zadatak 1.40 Napisati program koji realizuje autocomplete funkcionalnost. Sa standardnog ulaza korisnik unosi delove reči sve dok ne unese karakter !. Nakon svakog unetog dela reči ispisuju se reči koje počinju tim karakterima. Spisak reči koje program može da predloži se nalazi u datoteci `reci.txt`.

Primer 1

```
POKRETANJE: python autocomplete.py
INTERAKCIJA SA PROGRAMOM:
 ma
 mac macka mama maceta madjionicar
 mac
 mac macka maceta
 !
```

Sadržaj datoteka koje se koriste u primeru 1.40:

Listing 1.16: *reci.txt*

```
mac pesma skola macka mama maceta igra madjionicar
```

1.4 Rešenja

Rešenje 1.12 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 = False
while not pogodio:
    print "Unesi broj:"
    broj = int(raw_input())
    if broj == zamisljen_broj:
        pogodio = True
    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 1.13 Aproksimacija broja PI metodom Monte Karlo

```
# Aproksimacija broja PI metodom Monte Karlo

import random, math

def dist(A, B):
    """Funkcija izracunava euklidsko rastojanje izmedju tacaka A i B"""
    return math.sqrt((A[0]-B[0])**2 + (A[1]-B[1])**2)

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

i = N
while i >= 0:
    tacka = (random.random(), random.random())
    # Ukoliko se tacka nalazi u krugu, povecavamo broj tacaka u krugu
    if dist(tacka, (0.5, 0.5)) <= 0.5:
        A = A + 1
    B = B + 1
    i = i - 1

# Alternativno resenje:
# Generisemo N tacaka unutar kvadrata
# list comprehensions:
# http://www.pythonforbeginners.com/basics/list-comprehensions-in-python
xs = [(random.random(), random.random()) for x in range(N)]
# Izdvajamo tacke koje su unutar kvadrata
# lambde:
# https://pythonconquerstheuniverse.wordpress.com/2011/08/29/lambda_tutorial/
inside = filter(lambda (x, y): dist((0.5, 0.5), (x, y)) <= 0.5, xs)
```

```
A = len(inside)
B = N

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

Rešenje 1.14 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, "0")

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

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

print "Zapocnimo igru \n"

ispisi_tablu(tabla)

na_redu = 0
iteracija = 0
igraci = [prvi_igrac, drugi_igrac]
while iteracija < 9:
    korak(igraci[na_redu])
    if pobedio(tabla) == True:
        print "BRAVO!!!!!!! Igrac {0:s} je pobedio!\n".format(igraci[na_redu][0])
        break
    na_redu = (na_redu+1)%2
    iteracija = iteracija + 1

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

Rešenje 1.25

```
# dat.txt:
# Ovo je datoteka dat
#
# rezultat.json:
#
# {"a ": 1, "ka": 1, "ot": 1, "ek": 1, " d": 2, " j": 1, "da": 2, "e ": 1, "o ": 1, "
    to": 1, "at": 2, "je": 1, "Ov": 1, "te": 1, "vo": 1}

import json

ime_datoteke = raw_input("Unesite ime datoteke: ")
n = int(raw_input("Unesite broj n: "))

# Otvaramo datoteku i citamo njen sadrzaj
f = open(ime_datoteke, "r")
sadrzaj = f.read()
f.close()

recnik = {}
i = 0
# Prolazimo kroz sadrzaj i uzimamo jedan po jedan n-gram
while i < len(sadrzaj) - n:
    ngram = sadrzaj[i : i+n]
    # Ukoliko se n-gram vec nalazi u recniku,
    #  povecavamo mu broj pojavljivanja
    if ngram in recnik:
        recnik[ngram] = recnik[ngram]+1
    # Dodajemo n-gram u recnik i postavljamo mu broj na 1
    else:
        recnik[ngram] = 1
    i = i + 1

f = open("rezultat.json", "w")
json.dump(recnik,f)
f.close()
```

Rešenje 1.26

```
import json

```

```
def cena_voca(prodavnica, ime_voca):
  for voce in prodavnica:
    if voce['ime'] == ime_voca:
      return voce['cena']

# Ucitavamo podatke iz datoteka
f = open('korpa.json', "r")
korpa = json.load(f)
f.close()

f = open('maxi_cene.json', "r")
maxi_cene = json.load(f)
f.close()

f = open('idea_cene.json', "r")
idea_cene = json.load(f)
f.close()

f = open('shopngo_cene.json', "r")
shopngo_cene = json.load(f)
f.close()

maxi_racun = 0
idea_racun = 0
shopngo_racun = 0
i = 0
# Za svako voce u korpi dodajemo njegovu cenu u svaki racun posebno
while i < len(korpa):
    ime_voca = korpa[i]['ime']
    maxi_racun = maxi_racun + korpa[i]['kolicina']*cena_voca(maxi_cene, ime_voca)
    idea_racun = idea_racun + korpa[i]['kolicina']*cena_voca(idea_cene, ime_voca)
    shopngo_racun = shopngo_racun + korpa[i]['kolicina']*cena_voca(shopngo_cene,
    ime_voca)
    i += 1

print "Maxi: " + str(maxi_racun) + " dinara"
print "Idea: " + str(idea_racun) + " dinara"
print "Shopngo: " + str(shopngo_racun) + " dinara"
```

Rešenje 1.35

```
import os

dat_u_duze = 0
dat_u_sire = 0

# Funkcija koja obilazi datoteku i vraca 1 ukoliko datoteka pripada skupu duze
# odnosno 0 ukoliko datoteka pripada skupu sire
def obilazak(ime_datoteke):
    br_linija = 0
    najduza_linija = 0
    with open(ime_datoteke, "r") as f:
        for linija in f:
            br_linija = br_linija + 1
            if len(linija) > najduza_linija:
                najduza_linija = len(linija)
    if br_linija > najduza_linija:
        return 1
    else:
        return 0

ime_direktorijuma = raw_input("Unesite putanju do direktorijuma: ")

for (tren_dir, pod_dir, datoteke ) in os.walk(ime_direktorijuma):
    for dat in datoteke:
        if obilazak(os.path.join(tren_dir, dat)) == 0:
            dat_u_sire += 1
        else:
            dat_u_duze += 1

print "Kardinalnost skupa duze: kardinalnost skupa sire"
print str(dat_u_duze)+":"+str(dat_u_sire)
```

Rešenje 1.36

```
import os
import json

ime_direktorijuma = raw_input("Unesite putanju do direktorijuma: ")

ekstenzije = {}

for (tren_dir, pod_dir, datoteke) in os.walk(ime_direktorijuma):
    for dat in datoteke:
        pozicija = dat.find(".")
        # Ukoliko datoteka ima ekstenziju, pretpostavljamo da su datoteke imenovane tako da posle . ide ekstenzija u ispravnom obliku
        if pozicija >= 0:
            # Ukoliko ekstenzija postoji u mapi, povecavamo njen broj
            if dat[pozicija:] in ekstenzije:
                ekstenzije[dat[pozicija:]] += 1
            else:
            # Dodajemo novu ekstentiju u mapu i postavljamo njen broj na 1
                ekstenzije[dat[pozicija:]] = 1

with open("rezultat.json","w") as f:
    json.dump(ekstenzije, f)
```

Rešenje 1.37

```
import json, os, sys
from datetime import datetime

try:
    with open("radnici.json", "r") as f:
        radnici = json.load(f)
except IOError:
    print "Otvaranje datoteke nije uspelo!"
    sys.exit()

opcija = raw_input("Unesite opciju koju zelite (d - dostupni radnici, o - radnici na odmoru): \n")

if opcija != "d" and opcija != "o":
    print "Uneta opcija nije podrzana."
    exit()

tren_dat = datetime.now()

# funkcija datetime.strptime(string, format) pravi objekat tipa datetime na osnovu zadatih podataka u stringu i odgovarajuceg formata, na primer ako je datum zapisan kao "21.08.2016" odgovarajuci format je "%d.%m.%Y." pa se funkcija poziva sa datetime.strptime("21.08.2016", "%d.%m.%Y.")

for radnik in radnici:
    kraj_odmora = datetime.strptime(radnik['odmor'][1],"%d.%m.%Y.").date()
    pocetak_odmora = datetime.strptime(radnik['odmor'][0],"%d.%m.%Y.").date()
    kraj_rad_vrem = datetime.strptime(radnik['radno_vreme'][1],"%H:%M").time()
    pocetak_rad_vrem = datetime.strptime(radnik['radno_vreme'][0],"%H:%M").time()
    if opcija == "o":
        # Ukoliko je radnik trenutno na odmoru ispisujemo ga
        if  pocetak_odmora < tren_dat.date() < kraj_odmora:
            print radnik["ime"]
    else:
        # Ukoliko je radnik trenutno dostupan i nije na odmoru, ispisujemo ga
        if not (pocetak_odmora < tren_dat.date() < kraj_odmora) and pocetak_rad_vrem < tren_dat.time() < kraj_rad_vrem:
            print radnik["ime"]
```

Rešenje 1.40

```
import os
```

```
ime_datoteke = raw_input("Unesite ime datoteke: ")

# pretrazujemo ceo fajl sistem, odnosno pretragu krecemo od root direktorijuma /
# imajte u vidu da ce vreme izvrsavanja ovog programa biti veliko posto se pretrazuje
    ceo fajl sistem, mozete ga prekinuti u svakom trenutku sa CTRL+C
for (tren_dir, pod_dir, datoteke) in os.walk("/"):
    # objekat datoteke predstavlja listu imena datoteka iz direktorijuma
    # ta imena poredimo sa zadatim
    for dat in datoteke:
        # ako smo naisli na trazenu datoteku, pravimo odgovarajucu putanju
        if dat == ime_datoteke:
            print os.path.join(os.path.abspath(tren_dir), ime_datoteke)
```