

# **OSNOVE PROGRAMIRANJA PYTHON**

Profesor

Milan Paroški

Novi Sad,  
2024/2025



# **Poglavlje 10**

## **Rad s fajlovima u jeziku Python**

### **16.12.2024.**

**Projekat : Jovana Stoislavljević**  
**K2 i K1**

# Sadržaj

1. Uvod
2. Pristup fajlu
3. Učitavanje teksta
4. Učitavanje numeričkih podataka
5. Upis teksta na fajl
6. Strukturirani podaci u formatu JSON
7. Čitanje teksta s Interneta
8. Primeri programa



## 10.1 Uvod

Jedan od načina na koji **Internet omogućava pristup i deljenje podataka i informacija smeštenih na različitim serverima je u obliku ?**

### **fajlova/datoteka**

Osim deljenja podataka i informacija, dele se i programi za upotrebu tih informacija.

**Prema formatu zapisa i načinu pristupa sadržaju fajlova/datoteka, razlikuju se dve vrste fajlova:**

- **tekstualni fajlovi i**
- **binarni fajlovi.**

## Tekstualni fajlovi sadrže znakove strukturirane kao linije teksta.

- Osim znakova koji se mogu štampati, **tekstualni fajlovi sadrže znakove koji nemaju vizuelnu interpretaciju.**
- Takav je npr. znak za kraj linije i prelazak u novu liniju teksta (`\n`).
- Šta se štampa sa: `print("1\n2")`?
- Prikaz ovog znaka (`\n`) na ekranu računara premešta kursor na početak sledećeg reda ekrana.
- **Dobra osobina tekstualnih fajlova je da se mogu kreirati pomoću običnog tekst editora.**

**Binarni fajlovi** se posmatraju kao niz bajtova,  
binarnih cifara(bitova) spojenih u grupu od 8.

Binarne datoteke obično sadrže bajtova koje treba  
tumačiti drugačije u odnosu na tekst karaktera.

Kompajlirani programi su

tipični primeri; ponekad

kompajlirane aplikacije

nazivamo **binarnim**

**Datotekama** (sadrže slike,

zvukove, smanjene oblike

drugih datoteka itd.

ukratko, bilo koji tip

sadržaja datoteke.)

bitovi $b_3b_2b_1b_0$	bitovi $b_6b_5b_4$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	`	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	.	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

- Binarni fajlovi imaju složeniju organizaciju i mogu se koristiti samo pomoću namenskih programa.
- Osnovne operacije za sve tipove fajlova su :
  - pristup ili **otvaranje fajla ?**

**open**

- **čitanje fajla -?**

**read**

- **upis u fajl ?**

**write**

- **zatvaranje fajla ?**

**close**



## 10.2. Pristup fajlu

Prvo je neophodno naredbom **open** uraditi?

**kreiranje objekta u memoriji koji je povezan s određenim fajlom:**

**<file> = open(<parametri>)**

**Šta znači: (<parametri>)?**

Navode se parametri u obliku teksta, kao što su npr.

- putanja fajla, koja može biti jednostavna ('primer.txt'), relativna ili apsolutna, npr.

**'C:\\Users\\korisnik\\Documents\\Desktop\\primer.txt').**

Umesto oznake za specijalni znak "\\" u putanji se može koristiti znak "/",

npr. **'C:/Users/korisnik/Documents/Desktop/primer.txt';**



- Posle toga treba definisati ?

### **način pristupa:**

1. r?

Čitanje

2. w?

Pisanje

3. a ?

Dodavanje

4. rb?

čitanje binarnog zapisa

5. wb ?

pisanje binarnog zapisa



Primer: fajl:singi.txt

C:\Users\Milan\AppData\Local\Programs\Python\Python311\datoteke

Singidunum

Centar Novi Sad

Adresa: Bulevar Mihajla Pupina 4a,

21000 Novi Sad

Telefon: 021/6621 900

Telefon: 021/6621-901

Email: novisad@singidunum.ac.rs

- Primer naredbe za otvaranje tekstualnog fajla **singi.txt** samo radi čitanja je:

***ulaz = open(„singi.txt“, "r")***

- Ukoliko se fajl ne nalazi na tekućem folderu, putanja se može se zadati na jedan od uobičajenih načina, npr.

***ulaz = open("c:/Python/singi.txt", "r") ili***

***ulaz = open("c:\\Python\\singi.txt", "r") i***

## 10.3. Učitavanje teksta

**Nakon otvaranja fajla radi čitanja, koriste se odgovarajući metodi za čitanje teksta i njegovo smeštanje u memoriju:**

- metod **read(n)** koristi se za čitanje **n znakova** kao jednog stringa. **Ako se broj znakova izostavi, čita se ceo sadržaj fajla kao jedan string;**
- metod **readline()** koristi se a čitanje sledeće linije teksta kao stringa. Čitanje počinje od prve linije;
- metod **readlines()** koristi se za čitanje svih linija iz fajla, koji se u memoriji predstavljaju u obliku liste stringova.

Program učitava ceo sadržaj tekstualnog fajla, kao jedan string, uključujući i specijalne znakove \n, koji označavaju kraj svake od linija teksta:

```
ulaz = open("singi.txt", "r")  
# Čitanje celog sadržaja fajla kao stringa,  
sadrzaj = ulaz.read()  
print(sadrzaj)  
ulaz.close()
```

**Rezultat, kada je ovaj fajl na dir :  
podaci:**

**Inače ne radi**

Singidunum  
Centar Novi Sad

Adresa: Bulevar Mihajla Pupina 4a,  
21000 Novi Sad

Telefon: 021/6621 900  
Telefon: 021/6621-901

Email: [novisad@singidunum.ac.rs](mailto:novisad@singidunum.ac.rs)

# Readlines

```
ulaz = open("singi.txt", "r")  
# Čitanje celog sadržaja fajla kao stringa, s \n  
sadrzaj = ulaz.readlines()  
print(sadrzaj)  
ulaz.close()
```

Rezultat:

```
['Singidunum\n', 'Centar Novi Sad\n', '\n', 'Adresa: Bulevar  
Mihajla Pupina 4a, \n', '21000 Novi Sad\n', '\n', 'Telefon:  
021/6621 900\n', 'Telefon: 021/6621-901\n', '\n', 'Email:  
novisad@singidunum.ac.rs\n', '\n', '\n', '\n', '\n', '\n', '\n']
```

# Readline

```
ulaz = open("singi.txt","r")  
# Čitanje celog sadržaja fajla kao stringa, s \n  
sadrzaj = ulaz.readline()  
print(sadrzaj)  
ulaz.close()
```

Rezultat:

*Singidunum*



## #Čitanje cele datoteke red po red

with open('singi.txt', 'r') as f:

```
    line = f.readline()
```

```
    while line:
```

```
        line = f.readline()
```

```
        print(line)
```

With se koristi za otvaranje fajla singi.txt koji smešta u promenljivu f

Metod **readline()** čita **jedan red** iz datoteke i vraća ga kao string

**while line:** Proverava da li promenljiva **line** sadrži neku vrednost.

Ako je **line** prazan string ("), petlja se prekida.

**line = f.readline()** - Unutar petlje, poziva se opet **readline()** da pročita sledeći red iz datoteke.

**print(line)** - Ispisuje sadržaj trenutno pročitano g reda na ekran

Veličina fajla:

```
import os  
b = os.path.getsize("singi.txt")  
print(b)
```

Rezultat :

173

Ako se proba sa fajlom singi1.txt ?:

The system cannot find the file specified:

'singi1.txt'



Zašto je rezultat: 173

Kada se u exploreru vidi 1 KB?

 singi.txt

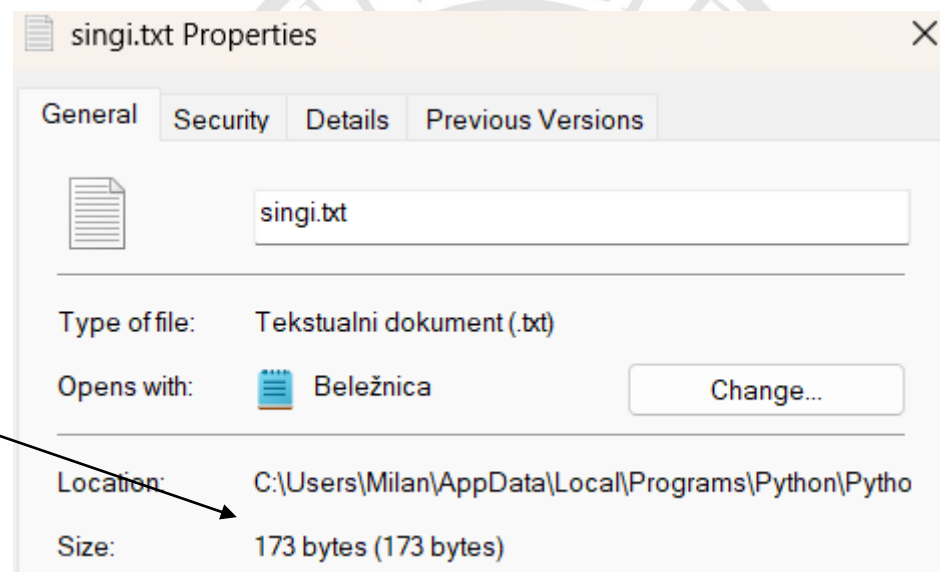
27.11.2023. 12:44

Text Document

1 KB

173B se vidi kao 1KB

Properties:



Ukoliko je fajl prevelik da se ceo učitava u memoriju i obradi, njegovi delovi (obično pojedinačne linije) čitaju se u petlji i redom obrađuju:

```
ulaz = open("singi.txt", "r")
```

```
linija = ulaz.readline()
```

```
# prva linija teksta
```

```
i=0
```

```
while linija != "": # Obrada jedne linije teksta ...
```

```
    i=1+i
```

```
    linija = ulaz.readline() # čitanje sledeće linije
```

```
    print("Broj linije", i, " ", linija)
```

```
print("KRAJ! ")
```



C:/Users/Milan/AppData/Local/Programs/Python/Python311/singi.txt ==

Broj linije 1      Centar Novi Sad

Broj linije 2

Broj linije 3      Adresa: Bulevar Mihajla Pupina 4a,

Broj linije 4      21000 Novi Sad

Broj linije 5

Broj linije 6      Telefon: 021/6621 900

Broj linije 7      Telefon: 021/6621-901

Broj linije 8

Broj linije 9      Email: novisad@singidunum.ac.rs

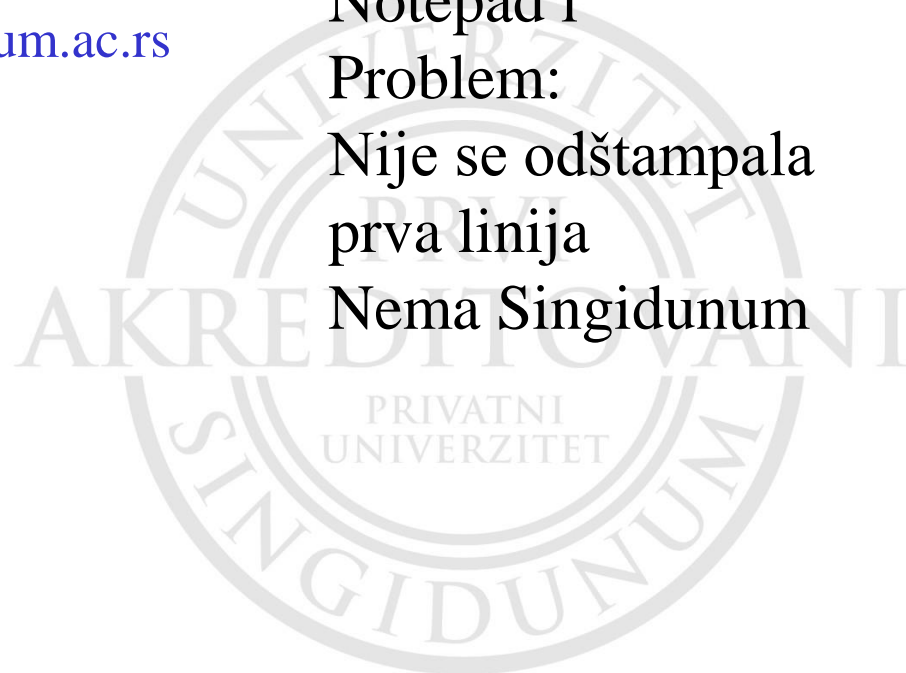
Broj linije 10

KRAJ!

Zašto ima 10 linija?

Otvoriti singi.txt u  
Notepad i  
Problem:

Nije se odšampala  
prva linija  
Nema Singidunum



## Zasto nije odstampana prva linija?

```
ulaz = open("singi.txt", "r")
```

```
linija = ulaz.readline()
```

```
# prva linija teksta
```

```
i=0
```

```
print("Broj linije",i, "    ",linija)
```

```
while linija != "": # Obrada jedne linije teksta ...
```

```
    i=1+i
```

```
    linija = ulaz.readline() # čitanje sledeće linije
```

```
    print("Broj linije",i, "    ",linija)
```

```
print("KRAJ! ")
```

Sa ovim se  
štampa i prva  
linija  
Zašto?

Prvi print je  
za i=1 a treba  
i za i=0

== RESTART:

C:/Users/Milan/AppData/Local/Programs/Python/Python311/podaci.txt ==

Broj linije 0      Singidunum

Broj linije 1      Centar Novi Sad

Broj linije 2

Broj linije 3      Adresa: Bulevar Mihajla Pupina 4a,

Broj linije 4      21000 Novi Sad

Broj linije 5

Broj linije 6      Telefon: 021/6621 900

Broj linije 7      Telefon: 021/6621-901

Broj linije 8

Broj linije 9      Email: [novisad@singidunum.ac.rs](mailto:novisad@singidunum.ac.rs)

Broj linije 10

KRAJ!



#Učitamo čitav fajl i stavimo u string varijablu

```
s = open('singi.txt').read()
```

```
print(s)
```

Rezultat:

Singidunum

Centar Novi Sad

Adresa: Bulevar Mihajla Pupina 4a,

21000 Novi Sad

Telefon: 021/6621 900

Telefon: 021/6621-901

Email: [novisad@singidunum.ac.rs](mailto:novisad@singidunum.ac.rs)

Ovakav način je pogodan  
za čitanje celog fajla, bez  
potrebe za ispitivanjem  
uslova za završetak petlje.



```
#Još jedan način za ispisivanje celog sadržaja  
for c in open('singi.txt'):  
    print(c)
```

Pojašnjenje:

**1.open('singi.txt')** - Metod **open()** otvara datoteku sa imenom **singi.txt** u podrazumevanom **read** (čitanje) režimu. Datoteka se otvara liniju po liniju.

**2.for c in open('singi.txt'):** - **for** petlja iterira kroz sve redove datoteke **singi.txt**. Promenljiva **c** predstavlja **jedan red datoteke** pri svakoj iteraciji. Linije se čitaju automatski jedna po jedna.

**3.print(c)** - Ispisuje svaki red datoteke na standardni izlaz

Jezik Python omogućava čitanje i obradu podataka s fajla pomoću petlje for, tako što se koristi forma naredbe:

```
lines = [line for line in open('singi.txt')]
```

```
print(lines)
```

```
['Singidunum\n', 'Centar Novi Sad\n', '\n', 'Adresa: Bulevar  
Mihajla Pupina 4a, \n', '21000 Novi Sad\n', '\n', 'Telefon:  
021/6621 900\n', 'Telefon: 021/6621-901\n', '\n', 'Email:  
novisad@singidunum.ac.rs']
```

*da li bi ovo radilo:*

```
lines = [linija for linija in open('singi.txt')]
```

```
print(lines)
```

Sta je \n ? Inače rezultat je isti kao i sa naredbom readlines



```
lines = [line.strip() for line in open('singi.txt')]  
print(lines)
```

Rezultat:?

```
['Singidunum', 'Centar Novi Sad', '', 'Adresa: Bulevar Mihajla  
Pupina 4a,', '21000 Novi Sad', '', 'Telefon: 021/6621 900',  
'Telefon: 021/6621-901', '', 'Email:  
novisad@singidunum.ac.rs']
```

**String metoda strip uklanja sve blanko znakove na početku i na kraju stringa.**

```
str='  Dobar dan  '  
print(str.strip())
```

Dobar dan



Ako želimo da uklonimo blanko znakove samo sa kraja linije,  
koristićemo metodu ?

`rstrip`

```
str='  Dobar dan  '
```

```
print(str.rstrip())
```

Rezultat:

Dobar dan

Kako mozemo proveriti da je uklonio blanko znakove samo sa  
kraja linije?



```
str=' Dobar dan '  
print(str)  
print(len(str))  
str1=str.rstrip()  
print(str1)  
print(len(str1))
```

Dobar dan  
17  
Dobar dan  
13



```
str='  Dobar dan  '  
print(str.lstrip())
```

Dobar dan



```
str='  Dobar dan  '  
print(str)  
print(len(str))  
str1=str.lstrip()  
print(str1)  
print(len(str1))
```

```
Dobar dan  
17  
Dobar dan  
13  
>>>
```



# Kopiranje fajlova i direktorijuma

Modul **shutil** ima opcije koje nam omogućavaju **kopiranje, premeštanje, preimenovanje i brisanje** datoteka.

Putanja se u ovom slučaju piše koristeći \ ili /

```
import shutil
```

```
shutil.copy(izvor, odredište)
```

```
import shutil
```

```
shutil.copy("C:\\Users\\Milan\\AppData\\Local\\Programs\\Python\\Python311\\datoteke\\singi.txt", "C:\\Milan\\test")
```

Ako ne postoji direktorijum „test“ šta će se desiti?

kopiraće singi.txt u fajl test – izbrisati DIR test i probati ponovo

## 2.čas

17 i 31



Dok **shutil.copy()** kopira samo fajl, **shutil.copytree()**

kopira čitav folder i svaki drugi folder ili datoteku koji su sadržani u njemu. Pozivanjem `shutil.copytree(izvor, odrediste)` će se kopirati folder sa putanje izvora na putanju odredišta

```
import shutil
```

```
import os
```

```
# Source path
```

```
src = "C:/Milan/test"
```

```
print("spisak fajlova:")
```

```
print(os.listdir(src))
```

```
# Destination path
```

```
dest = "C:/Odluke,,
```

```
print("Destination path:", dest)
```

```
shutil.copytree(src, dest)
```

spisak fajlova:

['podaci.txt', 'za citanje.py']

Destination path: C:/Odluke

Pogledati c:\Odluke

**Uslov:** Na C disku da nema kataloga Odluka. Inače javi grešku  
Sada kada postoji DIR odluke probati ponovo pokrenuti program



**SINTAKSA: shutil.move(izvor, odredište)**

**Ispraznimo C:\Odluke**

**Samo promenimo u poslednjem redy copy u move:**

*import shutil*

*import os*

*# Source path*

*src = "C:/Milan/test"*

*print("spisak fajlova:")*

*print(os.listdir(src))*

*# Destination path*

*dest = "C:/Odluke"*

*print("Destination path:", dest)*

*shutil.move(src, dest)*

Rezultat=nema vise  
kataloga Milan/test



# Preimenovanje

*#Premeštanje sa promenom imena*

***import shutil***

***shutil.move("C:\\Milan\\test\\podaci.txt", "C:\\Milan\\test1\\1.txt")***

*Uslov:*

*da postoji podfolder: test1*

Da li će ovaj program kopirati 1.txt na podfolder proba?:

***import shutil***

***shutil.move("C:\\Milan\\test\\1.txt", "C:\\Milan\\test\\proba")***

Preimenovaće 1.txt u proba i na Milan\\test neće biti ništa

#Promena imena datoteka

```
os.rename(current_file_name, new_file_name)
```

```
import os
```

```
os.rename("C:/Milan/test/podaci.txt", "C:/Milan/test/podaci1.txt")
```

*#Brisanje datoteke*

```
import os
```

```
os.remove("C:/Milan/test/podaci.txt")
```



# Brisanje

## **Brisanje fajla:**

```
import os  
os.unlink('C:/milan/test/podaci2.txt')
```

## **Brisanje praznog foldera?:**

```
import os  
os.rmdir('C:/milan/test/test')
```

## **Brisanje punog foldera:**

```
import shutil  
shutil.rmtree('C:/milan/test/test')
```



#Listanje direktorijuma

import os

a=os.listdir("C:/odluke/")

print(a)

[]

#listanje tekućeg direktorijuma

import os


print(os.listdir(path=None))


['podaci.txt', 'singi.txt', 'za citanje.py']


#kreiranje direktorijuma

import os

os.mkdir("C:/Milan/test/nov")

 podaci.txt

 za citanje.py

 nov

*#Promena direktorijuma*

*import os*

*print(os.listdir(path=None))*

*os.mkdir("nov2")*

*os.chdir("nov2")*

*print(os.listdir(path=None))*

Rezultat:

`['podaci.txt', 'singi.txt', 'za citanje.py']`

`[]`



Može i ovako:

```
import os  
os.mkdir("nov1")
```

#Brisanje dir.

```
import os  
os.rmdir("nov1")
```



#Provera postojanja putanje ?

```
import os
```

```
print(os.path.exists("C:/Users/Milan/Links/"))
```

```
print(os.path.exists("C:/Users/Milan/Link/"))
```

Rezultat:

True

False





#Provera postojanja fajla?

```
import os
```

```
print(os.path.exists("C:/Milan/test/podaci.txt"))
```

```
print(os.path.exists("C:/milan/test/s.txt"))
```

Rezultat:

True

False



#Provera da li je datoteka ?

```
import os
```

```
print(os.path.isfile("C:/Milan/test/podaci.txt"))
```

```
print(os.path.isfile("C:/milan/test"))
```

Rezultat:

True

False



#Provera da li je direktorijum ?

```
import os
```

```
print(os.path.isdir("C:/Milan/test/podaci.txt"))
```

```
print(os.path.isdir("C:/milan/test"))
```

Rezultat:

False

True



## 10.4 Učitavanje numeričkih podataka

Numerički podaci su na fajlu zapisani u obliku teksta, uglavnom niza cifara.

Prilikom čitanja, tekstualni zapis broja potrebno je pretvoriti u binarni zapis brojeva u memoriji. Podaci na fajlu međusobno su razdvojeni nekim nenumeričkim znacima (delimiterima), kao što su npr. prazno mesto ili kraj linije '\n'.

Upotreba standardnih delimitera omogućava automatizaciju podele učitano stringa na delove pomoću string funkcije `split()`.

```
a="100 200 300 400".split()
```

```
print(a)
```

daje listu stringova:

Rezultat: ['100', '200', '300', '400']

```
a="100 200 300 400"
```

```
print(a)
```

***Rezultat?***

***100 200 300 400***

## **Sintaksa** : *str.split(separator, maxsplit)*

maxsplit: To je broj, koji nam govori da podelimo nisku na maksimalno obezbeđen broj puta. Ako nije obezbeđena, podrazumevana vrednost je -1 što znači da ne postoji ograničenje.

*word = ' Srem, Banat, Bačka, Mačva '*

*# maxsplit: 0*

*print(word.split(', ', 0))*

*# maxsplit: 4*

*print(word.split(', ', 4))*

*# maxsplit: 1*

*print(word.split(', ', 1))*

*# maxsplit: 2*

*print(word.split(', ', 2))*

['Srem, Banat, Bačka, Mačva']  
['Srem', 'Banat', 'Bačka', 'Mačva']  
['Srem', 'Banat, Bačka, Mačva']  
[' Srem', 'Banat', 'Bačka, Mačva ']

Svaki od stringova se zatim može konvertovati u odgovarajući tip podatka u memoriji pomoću ugrađene funkcije eval, npr.:

*# Otvaranje fajla i čitanje svih podataka*

*f = open("brojevi.txt", "r")* *# lista brojeva*

*s = f.read()*

*# Podela stringa s na delove i konverzija s eval()*

*brojevi = [eval(x) for x in s.split()]*

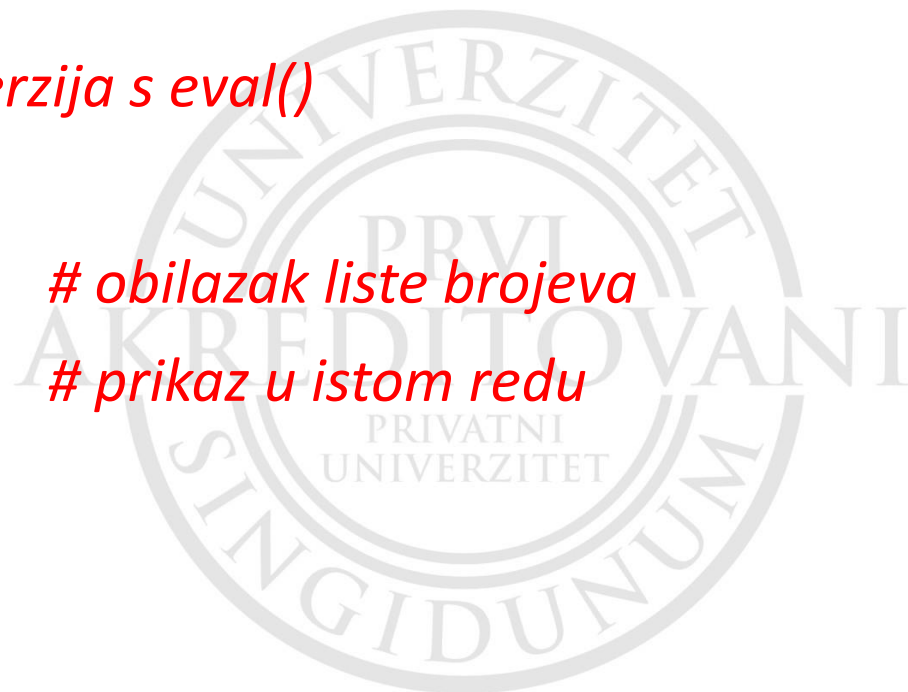
*broj in brojevi:*

*print(broj, end = " ")*

*f.close() # Zatvaranje fajla*

*# obilazak liste brojeva*

*# prikaz u istom redu*



- Funkcija vrši konverziju zadanog stringa u numeričku vrednost odgovarajućeg tipa:
  - `eval('100')` vraća celobrojnu vrednost 100, a
  - `eval('100.')` vraća decimalnu vrednost 100.0.
- Argument ove funkcije može biti i izraz, koji sadrži promenljive i zagrade. Funkcija raščlanjuje izraz i pokreće python kôd unutar programa

```
x=2  
y=eval('100+(x-1)')  
print(y)
```

101

```
x=2  
y=eval('100+(x-0.1)')  
print(y)
```

101.9

```
x=2  
y='100+(x-1)'  
print(y)
```

100+(x-1)

```
x=2
```

```
y=eval('x * x')
```

```
print(y)
```

4





## 10.5 Upis teksta na fajl

- **Upis brojeva na fajl naredbom write() vrši se nakon njihove konverzije u string**, koji se sastoji od niza cifara i eventualnih dodatnih oznaka, kao što su predznak, decimalni znak i eksponent.
- **Prilikom upisa, između ovih stringova cifara umeću se delimiteri**, za odvajanje brojeva prilikom čitanja.
- **U sledećem primeru otvara se fajl radi upisa i na njega se upisuje niz pseudoslučajnih brojeva, koji se pre upisa konvertuju u stringove pomoću funkcije str():**

```
import random
```

```
# Otvaranje fajla radi upisa podataka
```

```
f = open("brojevi.txt", "w")
```

```
# Zapis deset pseudoslučajnih brojeva na fajl
```

```
for i in range(10):
```

```
    broj = random.randint(0, 9)
```

```
    print(broj)
```

```
    f.write(str(broj)) # zapis stringa broj
```

```
f.close() # Zatvaranje fajla
```

8

8

5

4

9

9

5

5

9

2

Prikazati fajl: brojevi.txt u windows exploreru

Datoteka	Uredi	Prikazati
----------	-------	-----------

8854995592
------------

# 3.čas



Prikaz unetih brojeva:

```
s = open('brojevi.txt').read()  
print(s)
```

*Rezultat: 8854995592*



## 10.5.1 Operator formatiranja

- Formatiranje pomoću operatora formatiranja najstariji je način formatiranja (old-string-formatting), uveden još u verziji 2 jezika, po ugledu na jezik C.
- Za opis izlaza **koristi se operator formatiranja %** (modul), koji u jeziku Python ima dvostruku ulogu. Koju?
- $7\%5$
- Rezultat:?
- 2
- Kad su oba operanda brojevi, operator vraća ostatak od deljenja prvog operanda drugim.
- **Kada je prvi argument string, predstavlja operator formatiranja oblika:**

<format> % <lista\_vrednosti>

- Prvi operand je string u kome su označena mesta umetanja vrednosti iz liste, a istovremeno i opisan njihov format, na primer:

```
print("Heksadecimalni zapis broja %d je %X" % (77, 77))
```

'Heksadecimalni zapis broja 77 je 4D'      Zašto je 4D?

- $4 \cdot 16^1 + 13 + 16^0$
- Decimalni prikaz celobrojnih vrednosti označen je specifikacijom '%d', a heksadecimalnih sa '%X'.

- Prikaz decimalnih vrednosti definiše pomoću specifikacije '%f', a način prikaza stringova sa '%s':

```
print("Stringovni zapis broja %f je %s" % (77,77))
```

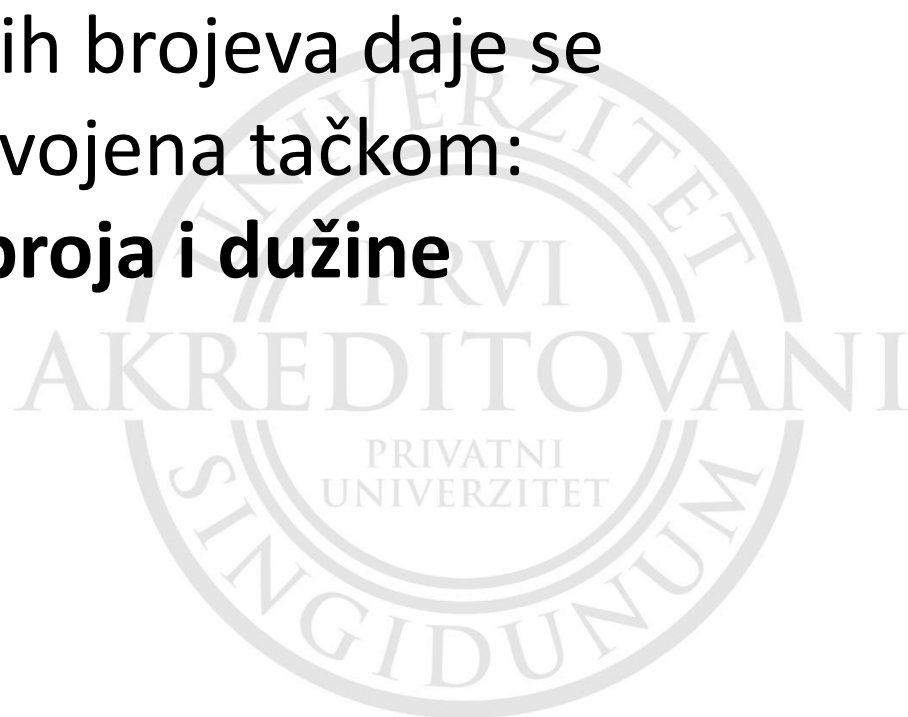
- Stringovni zapis broja 77.000000 je 77

- Osim tipa prikaza, **mogu se navesti i dužine pojedinih elemenata prikaza ispred oznaka:**

```
print("Vrednost %d/%d je %5.3f" % (1, 3, 1/3))
```

Vrednost 1/3 je 0.333

- Dužina prikaza decimalnih brojeva daje se pomoću dva podatka odvojena tačkom:  
**ukupne dužine prikaza broja i dužine decimalnog dela**



# Primeri:

```
str1 = "Ime studenta je {ime}, starost: {godine}".format(ime =  
    "Petar", godine = 21)
```

```
str2 = "Ime studenta je {0}, starost:{1}".format("Jovan","22")
```

```
str3 = "Ime studenta je {}, starost: {}".format("Marko",23)
```

```
str4 = "Ime studenta je {1}, starost:{0}".format("Jovan","22,,")
```

```
print(str1)
```

```
print(str2)
```

```
print(str3)
```

```
print(str4)
```

Ime studenta je Petar, starost: 21

Ime studenta je Jovan, starost:22

Ime studenta je Marko, starost: 23

Ime studenta je 22, starost:Jovan



**#Use "<" to left-align the value:      šta je ovo?**

```
str = "Imam {:<18} godine."
```

```
print(str.format(23))
```

Imam 23                      godine.

```
str = "Imam {:>18} godine."
```

```
print(str.format(23))
```

Imam                      23 godine

```
str = "Imam {:^18} godine."
```

```
print(str.format(23))
```

Imam            23            godine



```
str = "napolju je {:=18} temperatura. "      #broj mesta
```

```
print(str.format(-36))
```

napolju je - 36 temperatura.

*#ako je broj pozitivan piše se znak +*

```
str = "Temperatura je izmedju {:+} and {:+} stepeni."
```

```
print(str.format(-3, 7))
```

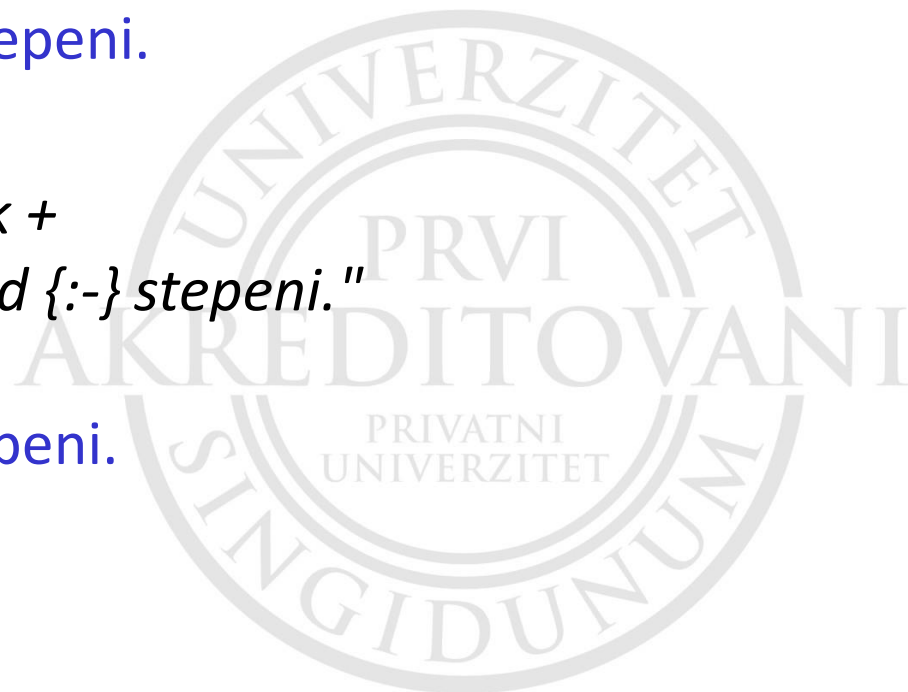
Temperatura je izmedju -3 and +7 stepeni.

*#ako je broj pozitivan ne piše se znak +*

```
str = "Temperatura je izmedju {: -} and {: -} stepeni."
```

```
print(str.format(-3, 7))
```

Temperatura je izmedju -3 and 7 stepeni.



*#broj praznih mesta*

*str = "Temperatura je izmedju {:5} and {:5} stepeni."*

*print(str.format(-3, 7))*

Temperatura je izmedju -3 and 7 stepeni.

*#Use "," to add a comma as a thousand separator:*

*txt = "Svet je star {:,} godina."*

*print(txt.format(13800000000))*

Svet je star 13,800,000,000 godina.

*#Use "b" to convert the number into binary format:*

*txt = "The binary version of {0} is {0:b}"*

*print(txt.format(5))*

The binary version of 5 is 101



```
txt = "Imam {:d} godina."  
print(txt.format(0b11101))  
Imam 29 godina.
```

```
#scientific format broja  
txt = "Imam {:e} godina."  
print(txt.format(0b11101))  
Imam 2.900000e+01 godina
```

```
.xf – fiksiranje formata broja  
txt = "Imam {:.6f} godina."  
print(txt.format(29))  
Imam 29.000000 godina.
```



*#Use "o" to convert the number into octal format:*

```
txt = "Oktalni broj broja {0} je {0:o}"
```

```
print(txt.format(13))
```

Oktalni broj broja 13 je 15

```
txt = "Heksa broj broja {0} je {0:x}"
```

```
print(txt.format(13))
```

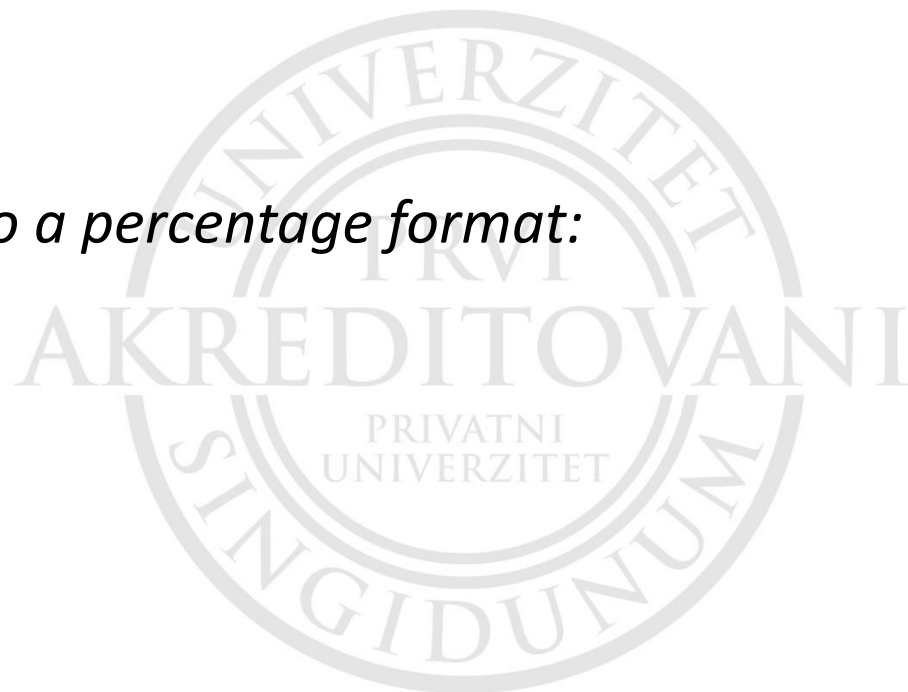
Heksa broj broja 13 je d

*#Use "%" to convert the number into a percentage format:*

```
txt = "Procenat je {:.2%}"
```

```
print(txt.format(0.25))
```

Procenat je 25.00%



## 10.5.3 Formatirani stringovi

- Najnovija verzija formatiranja, koja je uvedena u verziji 3.6 jezika koristi formatirane stringove, kraće f-stringove. Formatirani stringovi imaju prefix 'f' ili 'F' i osim teksta sadrže izraze zatvorene u velike zagrade, koji se evaluiraju u toku izvršavanja programa, npr.

*ime = "Milan"*

*print(f"Moje ime je {ime!r}.")*

Moje ime je 'Milan'.

- Oznaka '!' u izrazu označava da je potrebno izvršiti neku konverziju vrednosti, a 'r' označava konverziju u string koji se može prikazati (štampati).
- Bez !r bi bilo: Moje ime je Milan.

- Formatiranje numeričkih podataka na ovaj način vrši se takođe pomoću umetnutih izraza, koji se mogu gnezditi, kao npr. širina i broj cifara u opisu formata decimalnog broja:

*sirina= 10*

*Cifara=5*

*broj= float("12.34567")*

*print(f"rezultat: {broj:{sirina}.{Cifara}}")*

rezultat: 12.346

Ako se umesto 10 stavi 40 a 4 umesto 5:

rezultat: 12.35

Ako se stavi sirina=0 : rezultat: 12.346



## 10.6 Strukturirani podaci u formatu JSON

- Format zapisa podataka pomoću čitljivog teksta u formi rečnika razvijen je za jezik JavaScript kao otvoreni format JSON (JavaScript Object Notation), nezavisan od računara.
- Podrška za ovaj format zapisa i razmene podataka postoji u više programskih jezika, uključujući Python.
- Modul JSON deo je standardne biblioteke jezika Python.
- Format JSON veoma je sličan strukturama rečnika i liste u jeziku Python, tako da je konverzija dosta jednostavna.



Sledeći rečnik s ugnježdenim strukturama liste i rečnika veoma je sličan JSON strukturi podataka :

```
ime={'naziv':'Petar', 'prezime':'Petrović'}
```

```
rec = {'imeprezime':ime, 'polozaj':['razvoj', 'menadžer'],'starost':40.5}
```

```
print(rec)
```

```
{'imeprezime': {'naziv': 'Petar', 'prezime': 'Petrović'}, 'polozaj':  
['razvoj', 'menadžer'], 'starost': 40.5}
```



- Proces prevođenja objekata jezika Python u podatke u fajlovima koji su u formatu JSON, kao i njihovo čitanje vrši se odgovarajućim metodima iz modula json.
- Npr. zapis prethodno kreirane strukture rečnika **rec** u tekstualni fajl u formatu JSON može se izvršiti naredbom:

```
ime={'naziv':'Petar', 'prezime':'Petrović'}
```

```
rec = {'imeprezime':ime, 'polozaj':['razvoj', 'menadžer'],'starost':40.5}
```

```
import json
```

```
print(json.dump(rec, fp=open('podaci_json.txt', 'w'), indent=4))
```

Formira se fajl: *podaci\_json.txt*

```
{  
  "imeprezime": {  
    "naziv": "Petar",  
    "prezime": "Petrovi\u0107"  
  },  
  "polozaj": [  
    "razvoj",  
    "menad\u017eer"  
  ],  
  "starost": 40.5  
}
```



Sadržaj fajla se može učitati i prikazati naredbom:

```
ime={'naziv':'Petar', 'prezime':'Petrović'}
```

```
rec = {'imeprezime':ime, 'polozaj':['razvoj', 'menadžer'], 'starost':40.5}
```

```
import json
```

```
json.dump(rec, fp=open('podaci_json.txt', 'w'), indent=4)
```

```
print(open('podaci_json.txt').read())
```

```
{  
  "imeprezime": {  
    "naziv": "Petar",  
    "prezime": "Petrovi\u0107"  
  },  
  "polozaj": [  
    "razvoj",  
    "menad\u017eer"  
  ],  
  "starost": 40.5  
}
```



Potpuno tekstualni format nezavisan je i od sistema kodiranja znakova, tako da su znakovi koji ne pripadaju ASCII skupu označeni posebnim kodom.

- Učitana struktura se prevodi u strukturu rečnika jezika Python naredbom:

```
import json
```

```
P = json.load(open('podaci_json.txt'))
```

```
print(P)
```

Rezultat:

```
{'imeprezime': {'naziv': 'Petar', 'prezime': 'Petrović'}, 'polozaj':  
  ['razvoj', 'menadžer'], 'starost': 40.5}
```

Vidi se da je kreirana odgovarajuća struktura u memoriji, u kojoj je prikaz znakova po standardu Unicode.

Slična podrška u jeziku Python postoji i za rad s podacima koji su zapisani po standardu XML

## 10.7 Čitanje teksta s Interneta

- Jezik Python omogućava čitanje teksta fajlova koji se nalaze na serverima mreže Internet.
- Za pristup ovim fajlovima potrebno je koristiti metod **urlopen** iz modula **urllib.request**.

Urllib je Python paket koji omogućava pristup veb lokacijama i interakciju sa njima koristeći njihove URL adrese

Šta je URL?

(Uniform Resource Locator).

Ima nekoliko modula za rad sa URL-ovima:

1. `Urllib.request`
2. `Urllib.parse`
3. `Urllib.error`

Primeri:



## Urllib.request:

Korišćenje urllib.request, sa urlopen, omogućava da **otvorite navedeni URL**.

Kada se URL otvori, **funkcija read() se koristi za dobijanje celog HTML koda za veb stranicu**

Primer:

```
from urllib.request import urlopen
```

```
myURL = urlopen("http://www.google.com/")  
print(myURL.read())
```

Rezultat: b'<!doctype html><html itemscope=""  
itemtype="http://schema.org/WebPage" lang="sr"><head><meta  
content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta  
content="/images/branding/googleg/1x/googleg\_standard\_colo itd itd



1. Proveriti, tako što se na Microsoft edge, prikaže source kod stranice : google.com

Rešenje : CTRL + u

A kako na Firefox ili Chrome?

2. Umesto read staviti readline

3. from urllib.request import urlopen

```
myURL = urlopen("http://www.google.com/")
```

```
print(myURL.read()[0:123])
```

Rezultat: b'<!doctype html><html itemscope=""  
itemtype="http://schema.org/WebPage"  
lang="sr"><head><meta content="text/html; charset=UTF'

Probat: print(myURL.read()[0:12])

## Urllibparse:

Isečak koda.

URL je podeljen na svoje komponente kao što su :

- šema protokola koja se koristi,
- mrežna lokacija i
- putanja do veb stranice.

### Primer:

```
from urllib.parse import urlparse  
parsedUrl = urlparse('https://www.google.com')  
print(parsedUrl)
```

### Rezultat:

```
ParseResult(scheme='https', netloc='www.google.com',  
path='', params='', query='', fragment='')
```