INFORMATIČKI KLUB

FUTURA

python

PODATKOVNE ZBIRKE U PYTHONU

PREDAVANJE / RADIONICA

Tomo Sjekavica, Informatički klub FUTURA

Dubrovnik, 24. studenog 2017.

Creative Commons

- **slobodno možete:**
  - **Dijelite dalje** — možete umnažati i redistribuirati materijal u bilo kojem mediju ili formatu
  - **Stvarajte prerade** — možete remiksirati, mijenjati i prerađivati djelo
- **pod slijedećim uvjetima:**
  - **Imenovanje** — Morate **adekvatno navesti autora**, uvrstiti link na licencu i **naznačiti eventualne izmjene**. Možete to učiniti na bilo koji razuman način, ali ne smijete sugerirati da davatelj licence izravno podupire Vas ili Vaše korištenje djela.
  - **Nekomercijalno** — Ne smijete koristiti materijal u **komercijalne svrhe**.
  - **Dijeli pod istim uvjetima** — Ako remiksirate, mijenjate ili prerađujete materijal, Vaše prerade morate distribuirati pod **istom licencom** pod kojom je bio izvornik.

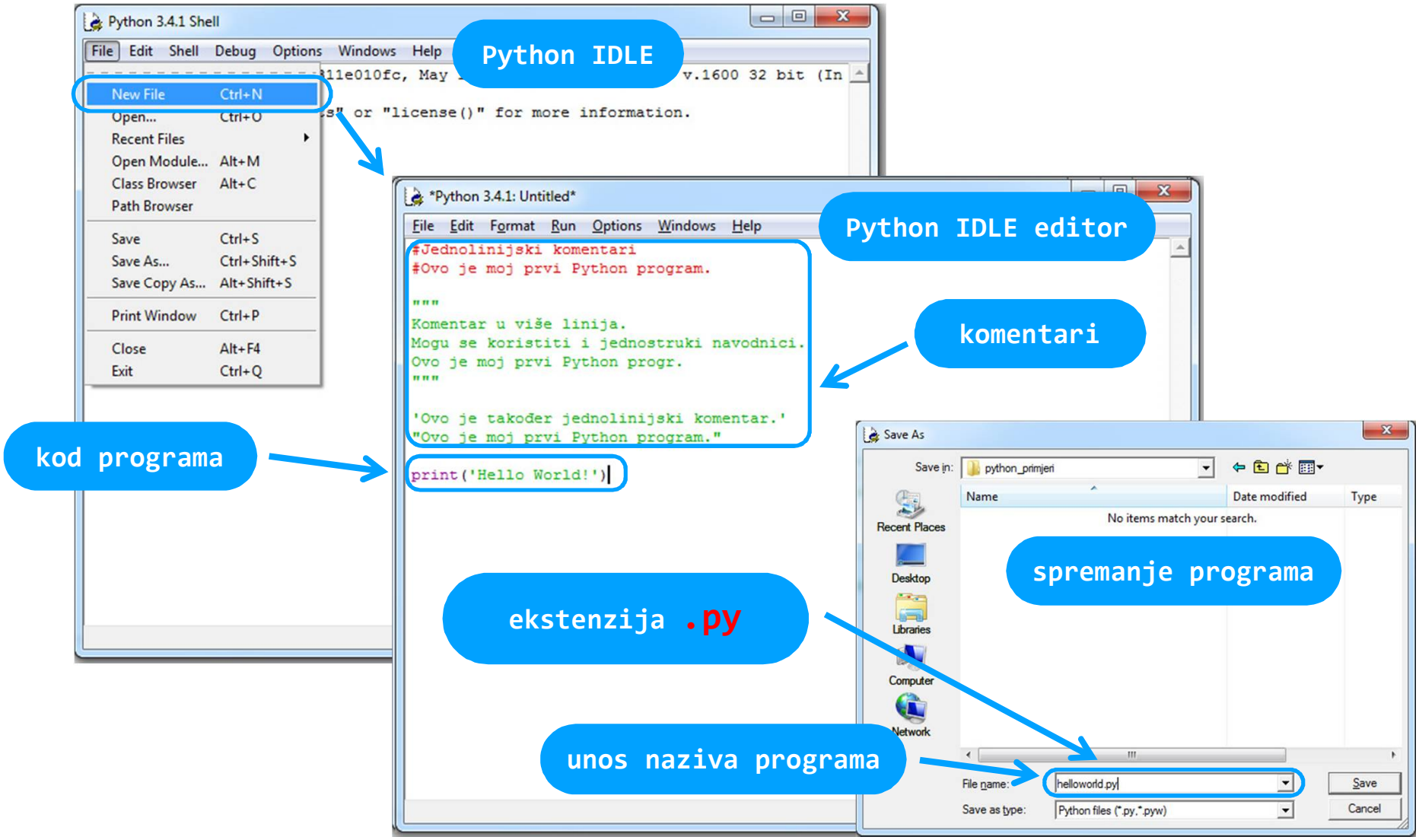
U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

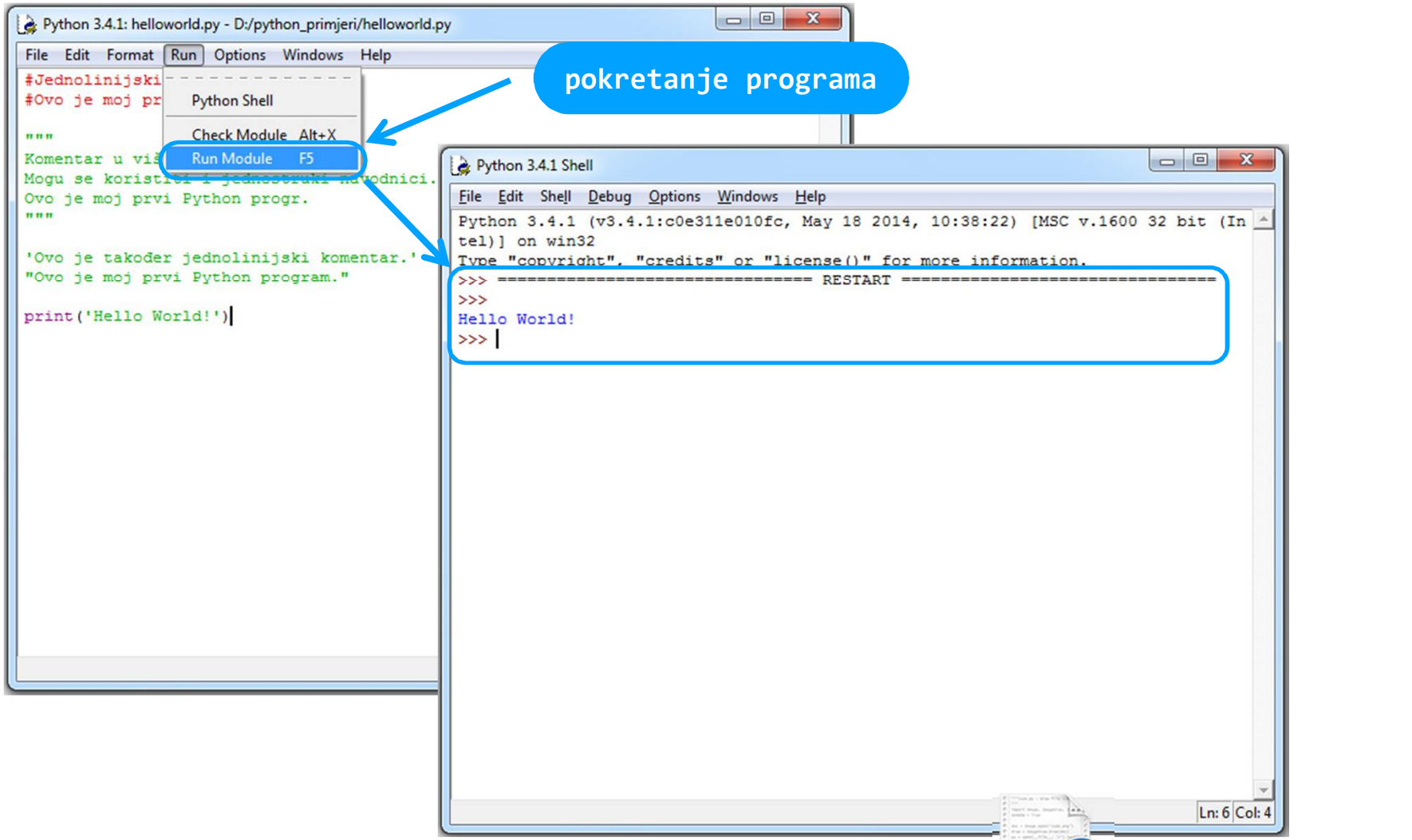
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

`Tekst licence preuzet je s http://creativecommons.org/`

Uvod: Prvi Python program



Uvod: Pokretanje programa



Uvod: Broj 10

VAŽNO: naredbe unutar if, elif i else moraju biti uvučene

funkcija input() se koristi za unos podataka s tipkovnice

funkcija input() vraća znakovni niz, pa se sa funkcijom int() pretvara u cijeli broj

```
broj = int(input('Unesite broj: '))
if broj == 10:
    print('Unijeli ste broj 10.')
elif broj < 10:
    print('Uneseni broj', 'je', 'manji od 10.')
else:
    print('Uneseni broj je veći od 10.')
print('Kraj programa.')
```

broj10.py

```
========== RESTART: C:/Users/Tomo/Desktop/broj10.py ==========
Unesite broj: 10
Unijeli ste broj 10.
Kraj programa.
>>>
========== RESTART: C:/Users/Tomo/Desktop/broj10.py ==========
Unesite broj: 5
Uneseni broj je manji od 10.
Kraj programa.
>>>
========== RESTART: C:/Users/Tomo/Desktop/broj10.py ==========
Unesite broj: 11
Uneseni broj je veći od 10.
Kraj programa.
>>>
```

1. pokretanje programa

2. pokretanje programa

3. pokretanje programa

Uvod: for petlja

```
>>> for i in range(4):
        print(i)

0
1
2
3
```

```
>>> for i in range(10):
        print(i, end = ' ')

0 1 2 3 4 5 6 7 8 9
```

```
>>> for i in range(3,12):
        print(i, end = ' ')

3 4 5 6 7 8 9 10 11
```

```
>>> for i in range(3,30,3):
        print(i, end = ' ')

3 6 9 12 15 18 21 24 27
```

```
>>> for i in range(5,1,-1):
        print(i, end = ' ')

5 4 3 2
```

```
>>> for i in range(1,5,-2):
        print(i, end = ' ')
```

Uvod: while petlja

```
>>> i = 0
>>> while i < 5:
        print(i, end = ' ')

0 0 0 0 0 0 0 0 0 0 0 Traceback (most recent call last):
File "<pyshell#37>", line 2, in <module>
  print(i, end = ' ')
File "C:\Python34\lib\idlelib\PyShell.py", line 1342, in write
  return self.shell.write(s, self.tags)
KeyboardInterrupt
```

beskonačna petlja:
ova petlja nema kraj, jer je i uvijek 0 i uvijek će biti zadovoljen uvjet petlje

Ctrl + C za prekid programa putem tipkovnice

```
>>> i = 0
>>> while i < 5:
        print(i, end = ' ')
        i += 1

0 1 2 3 4
```

inicijalizacija varijable i na vrijednost 0

uvećavanje vrijednosti varijable i za 1, isto kao da piše i = i + 1

Tipovi podataka u Pythonu

- Osnovni tipovi podataka u Pythonu su:
  - Cijeli brojevi – **`int`**
  - Brojevi s pomičnom točkom – **`float`**
  - Logički (Booleov) tip – **`bool`**
  - Znakovni nizovi (stringovi) – **`str`**
- Često se trebaju organizirati grupe srodnih podataka u **zbirke** (engl. *collections*).
- Zbirke se mogu podijeliti u dvije skupine:
  - zbirke sa slijednim smještanjem elemenata,
  - zbirke s raspršenim smještajem elemenata.

Podatkovne zbirke

- Zbirke sa **slijednim** smještanjem elemenata:
  - **znakovni nizovi ili stringovi (engl. *string*),**
  - **liste (engl. *lists*),**
  - n-torke (engl. *tuples*),
  - nizovi bajtova (engl. *Bytestrings*).
- Zbirke s **raspršenim** smještanjem elemenata:
  - skupovi (engl. *sets*),
  - rječnici (engl. *dictionaries*).

N-torke

- N-torka može sadržavati više elemenata različitog tipa podatka.
- Često se koriste za prenošenje (vraćanje) više vrijednosti iz funkcija.

```
>>> nTorka = ('Futura', 2017)
>>> print(nTorka)
('Futura', 2017)
>>> print(nTorka[1])
2017
>>> nTorka[1] = 2018
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    nTorka[1] = 2018
TypeError: 'tuple' object does not support item assignment
```

elementi u n-torki su omeđeni okruglim zagradama, a odvojeni zarezom

elementima n-torke se može pristupati pomoću indeksa, indeks 0 – 1. element

elementima n-torke se ne može mijenjati vrijednost

Skupovi

- Skupovi odgovaraju matematičkom pojmu skupa i nad njima se primjenjuju iste zakonitosti kao i u matematici.
- Sastoje se od strogo različitih vrijednosti.

```
>>> string = 'Futura'
>>> skup = set(string)
>>> print(skup)
{'u', 'r', 'a', 't', 'F'}
>>> len(string)
6
>>> len(skup)
5
```

funkcija set() pretvara znakovni niz (string) ili listu u skup

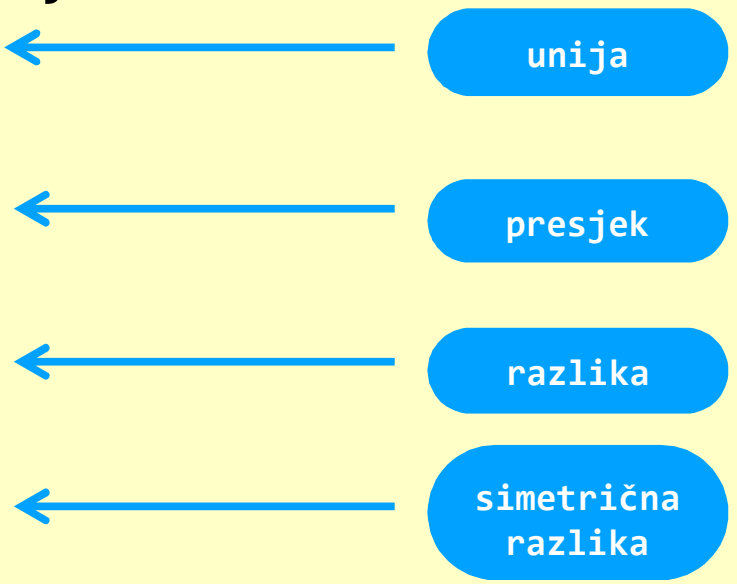
elementi u skupu su omeđeni vitičastim zagradama

redoslijed elemenata u skupu je proizvoljan i elementima se ne može pristupati pomoću indeksa

Skupovi

- Nad skupovima su definirane različite standardne matematičke operacije, kao što su unija, presjek, razlika, ...

```
>>> skup1 = {1, 2, 3, 4, 5}
>>> skup2 = {3, 4, 5, 6, 7}
>>> print(skup1 | skup2)
{1, 2, 3, 4, 5, 6, 7}
>>> print(skup1 & skup2)
{3, 4, 5}
>>> print(skup1 - skup2)
{1, 2}
>>> print(skup1 ^ skup2)
{1, 2, 6, 7}
```



Rječnici

- Rječnici sadrže parove vrijednosti: **ključ – pripradna vrijednost**. Vrijednosti se pohranjuju i dohvaćaju na temelju ključa, a ne preko indeksa.

nazivi ključeva su znakovni nizovi

```
>>> rjecnik = {'naziv':'Futura', 'prog':'Python', 'god':2017}
>>> print(rjecnik)
{'naziv': 'Futura', 'prog': 'Python', 'god': 2017}
>>> print(rjecnik.keys())
dict_keys(['naziv', 'prog', 'god'])
>>> print(rjecnik.values())
dict_values(['Futura', 'Python', 2017])
>>> print(len(rjecnik))
3
>>> print(rjecnik['naziv'])
Futura
```

dohvaćanje svih ključeva rječnika

dohvaćanje svih vrijednosti rječnika

dohvaćanje vrijednosti preko ključa

Znakovni nizovi (stringovi)

- Znakovni nizovi (stringovi) se tretiraju i kao osnovni tip podatka i kao zbirka podataka.
- Kada se znakovni niz koristi kao cjelina, tretira se kao osnovni tip podatka.
- Znakovni niz se može promatrati kao niz *Unicode* znakova, pa se elementima znakovnog niza može pristupiti preko njihovog indeksa, i tada se znakovni niz tretira kao zbirka podataka.

preporuka: korištenje jednostrukih navodnika

```
>>> 'Futura'
'Futura'
```

```
>>> "Futura"
'Futura'
```

```
>>> print('Futura')
Futura
```

Operatori za znakovne nizove

- Postoje 4 binarna operatora za rad s znakovnim nizovima (stringovima):

operator	opis djelovanja
`+`	spajanje (nadovezivanje)
`*`	uvišestručenje – jedan operand je tipa `int`
`in`	prvi znakovni niz sadržan je u drugom znakovnom nizu
`not in`	prvi znakovni niz nije sadržan u drugom znakovnom nizu

- Uvišestručenje:

kod uvišestručenja vrijedi zakon komutativnosti

```
>>> string1 = 'Futura'
>>> string2 = string1 * 3
>>> print(string2)
FuturaFuturaFutura
>>> print(2 * string2)
FuturaFuturaFuturaFuturaFuturaFutura
```

Operatori za znakovne nizove

- Spajanje znakovnih nizova:

```
>>> niz1 = 'Informatički'
>>> niz2 = 'klub'
>>> niz3 = 'Futura'
>>>
>>> niz = niz1 + niz2 + niz3
>>> print(niz)
InformatičkiklubFutura
>>>
>>> niz = niz1 + ' ' + niz2 + ' ' + niz3
>>> print(niz)
Informatički klub Futura
```

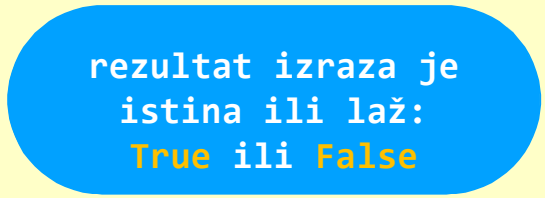
koristi se standardni operator za zbrajanje: +

svi operandi su znakovni nizovi (stringovi)

Operatori za znakovne nizove

- Operatori **in** i **not in** in ispituju da li je prvi string sadržan u drugom stringu:

```
>>> string1 = 'Futura'
>>> string2 = string1 * 3
>>> string1 in string2
True
>>> string1 not in string2
False
>>> string2 not in string1
True
>>> 'f' in string1
False
>>> 'F' in string1
True
>>> 'F' in string1 and 'ra' in string1
True
```



Ugrađene funckije za znakovne nizove

- Za rad s znakovnim nizovima postoji više ugrađenih funkcija:

funkcija	opis djelovanja
`len(niz)`	vraća duljinu znakovnog niza
`min(niz)`	vraća znak iz stringa koji ima najmanju kodnu vrijednost
`max(niz)`	vraća znak iz stringa koji ima najveću kodnu vrijednost
`ord(znak)`	vraća dekadski kod jednog znaka
`chr(kod)`	vraća znak određen dekaskim kodom **kod**
`str(broj)`	vraća znakovni prikaz broja **broj**

Ugrađene funckije za znakovne nizove

- Funkcija **`len`** - duljina znakovnog niza:

```
>>> futura = 'Informatički klub'
>>> len(futura)
17
>>> prazno = ''
>>> len(prazno)
0
>>> praznina = ' '
>>> len(praznina)
1
>>> string = futura + praznina + 'Futura'
>>> print(string, len(string))
Informatički klub Futura 24
```

Ugrađene funckije za znakovne nizove

- Funkcije **min**, **max**, **ord** i **chr**:

```
>>> niz1 = 'futura'
>>> print(min(niz1), max(niz1))
a u
>>> niz2 = 'Futura'
>>> print(min(niz2), max(niz2))
F u
>>> ord('f')
102
>>> ord('F')
70
>>> chr(69)
'E'
```

ispis znakova iz znakovnog niza sa najmanjom i najvećom kodnom vrijednosti

velika slova imaju manje kodne vrijednosti od malih slova

dekadski kod 69 odgovara velikom slovu 'E'

Ugrađene funckije za znakovne nizove

- Funkcija **str**: pretvaranje broja u string

```
>>> niz = 'Futura'
>>> broj = 2017
>>> print(niz + ' ' + broj)
Traceback (most recent call last):
  File "<pyshell#88>", line 1, in <module>
    print(niz + ' ' + broj)
TypeError: must be str, not int
>>>
>>> print(niz + ' ' + str(broj))
Futura 2017
>>>
```

varijabla broj je cjelobrojnog tipa - int

ne mogu se spajati nizovi znakova s cijelim brojem

pretvorba cijelog broj u znakovni niz s funckijom str()

Dohvaćanje znakova pomoću indeksa

- Svaki znak u znakovnom nizu ima svoj indeks, koji se može iskoristiti za dohvat jednog znaka:

```
>>> niz = 'Futura'
>>> niz[0]
'F'
>>> niz[1]
'u'
>>> niz[5]
'a'
>>> niz[len(niz)]
Traceback (most recent call last):
  File "<pyshell#96>", line 1, in <module>
    niz[len(niz)]
IndexError: string index out of range
>>> len(niz)
6
```

vrijednost indeksa se unosi unutar uglatih zagrada

prvi znak u nizu ima vrijednost 0

greška: indeks zadnjeg znaka u znakovnom nizu ima indeks len(niz)-1

Dohvaćanje znakova pomoću indeksa

- Pomoću indeksa se može dohvatiti i više znakova iz znakovnog niza:

```
>>> niz = 'Informatički klub Futura'
>>> niz[0:4]
'Info'
>>> niz[13:17]
'klub'
>>> niz[0:9] + niz[10] + niz[6]
'Informatika'
>>> niz[0:9] + niz[10] + niz[6] + niz[17:24]
'Informatika Futura'
>>>
```

dohvat prva četiri znaka u znakovnom nizu – s vrijednostima indeksa 0, 1, 2 i 3

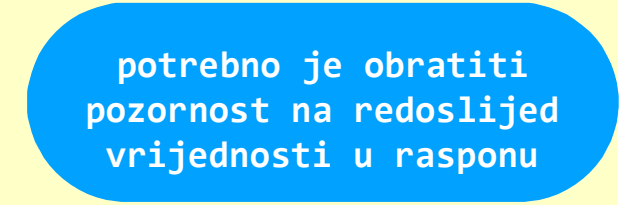
dohvat zadnjih šest znakova u znakovnom nizu – s vrijednostima indeksa 18, 19, 20, 21, 22 i 23

Dohvaćanje znakova pomoću indeksa

- Mogu se koristiti i negativni indeksi za dohvat znakova od kraja prema početku stringa:

0	1	2	3	4	5	6	7	8	9	10	11	12
F	**u**	**t**	**u**	**r**	**a**		**P**	**y**	**t**	**h**	**o**	**n**
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> niz = 'Futura Python'
>>> niz[-6]
'P'
>>> niz[-3:-5]
''
>>> niz[-5:-3]
'yt'
```



Zadatak: Palindrom

- Napisati program u kojem se unosi jedna riječ s tipkovnice. Ispisati da li je unesena riječ palindrom ili nije.

palindrom.py

Primjeri testnih podataka

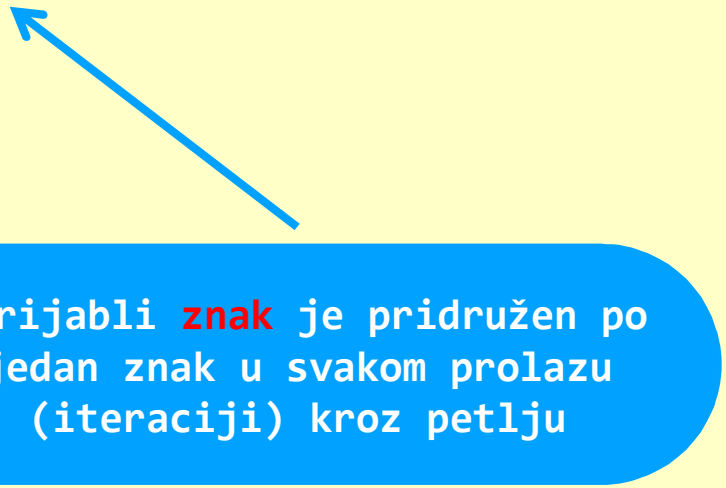
ULAZ	IZLAZ
ana	Palindrom!
ULAZ	IZLAZ
futura	Nije palindrom!
ULAZ	IZLAZ
radar	Palindrom!
ULAZ	IZLAZ
python	Nije palindrom!
ULAZ	IZLAZ
kajak	Palindrom!
ULAZ	IZLAZ
palindrom	Nije palindrom!

```
rijec = input('Unesite riječ: ')
obrnuta = ''

for i in range(len(rijec)):
    obrnuta = rijec[i] + obrnuta

if rijec == obrnuta:
    print('Palindrom!')
else:
    print('Nije palindrom!')
```

Iteriranje bez korištenja indeksa

- Kroz niz znakova je moće proći (iterirati) i bez korištenja indeksa:

```
>>> niz = 'Futura'
>>> for znak in niz:
      print(znak)


F
u
t
u
r
a
```



Ugrađene metode za znakovne nizove

- Za rad s znakovnim nizovima postoji i više ugrađenih metoda. Neke od njih su:

metoda	opis djelovanja
`s.upper()`	vraća kopiju stringa **s** sa svim velikim slovima
`s.lower()`	vraća kopiju stringa **s** sa svim malim slovima
`s.count(s1)`	vraća broj koliko se puta podniz **s1** pojavljuje u stringu **s**
`s.find(s1)`	vraća poziciju pojavljivanja podniza **s1** u stringu **s**, ili -1 ako podniz nije ponađen
`s.replace(s1, s2)`	vraća kopiju stringa **s** u kojem je svaki podniz **s1** zamijenjem podnizom **s2**
`s.split()`	podijelit će string **s** na manje podstringove i spremiti ih u listi,standardni separator je praznina

Ugrađene metode za znakovne nizove

- Metode **upper()**, **lower()** i **count(s1)**:

```
>>> futura = 'Informatički klub FUTURA'
>>>
>>> print(futura.upper())
INFORMATIČKI KLUB FUTURA
>>>
>>> print(futura.lower())
informatički klub futura
>>>
>>> print(futura.count('i'))
2
>>> print(futura.upper().count('I'))
3
```

vraća kopiju stringa sa svim velikim slovima

vraća kopiju stringa sa svim malim slovima

broji koliko puta se javlja zadani podniz

može se kombinirati više metoda – prvo se izvede upper(), pa onda count()

Ugrađene metode za znakovne nizove

- Metode **find(s1)** i **replace(s1, s2)**:

```
>>> futura = 'Informatički klub FUTURA'
>>>
>>> print(futura.find('mat'))
5
>>> print(futura.find('MAT'))
-1
>>>
>>> print(futura.replace('FUTURA','Python'))
Informatički klub Python
>>>
```

vraća poziciju prvog pojavljivanja zadanog podniza

podniz ne postoji u stringu, pa vraća -1 (to nije indeks)

vraća kopiju stringa sa zamijenjenim podnizom

Metoda split

- Metoda **`split`** vraća listu riječi iz zadanog niza znakova (standardni razdjelnik je praznina ‘ ‘).

```
>>> tekst = 'Podatkovne zbirke u Pythonu'
>>> tekst.split()
['Podatkovne', 'zbirke', 'u', 'Pythonu']
```

- Korisnik može kod poziva metode **`split`** postaviti razdjelnik po želji.

```
>>> tekst = 'Podatkovne zbirke u Pythonu'
>>> tekst.split(' ')
['Podatkovne', 'zbirke', 'u', 'Pythonu']
```

Metoda split

- Često se koristi kada se više podataka unosi u istom retku (s jednom input funkcijom).

```
>>> unos = input('4 broja odvojena zarezom: ')
4 broja odvojena zarezom: 2,5,3,6
>>> br1, br2, br3, br4 = unos.split(',')
>>> print(br1 + br2 + br3 + br4)
2536
>>> br1 = int(br1)
>>> br2 = int(br2)
>>> br3 = int(br3)
>>> br4 = int(br4)
>>> print(br1 + br2 + br3 + br4)
16
```

sve 4 varijable su stringovi, pa ih + spaja

da bi izračunali zbroj 4 broja, potrebno ih je prvo prebaciti u cijele brojeve s funckijom int()

Zadatak: Pravopis

- Iva stalno ratuje s pravopisom, pa često pomoćni glagol "će" napiše kao "če". Treba napisati program koji će Ivu ispraviti svaki put kad napiše pogrešno napiše "če".
- Ulazni podaci:
  - Ivina rečenica
- Izlazni podaci:
  - SVE OK – ako Iva nije pogriješila

  ... ili ...
  - broj grešaka
  - ispravna rečenica

Primjeri testnih podataka

ULAZ
Program če ispraviti česte greške
IZLAZ
Broj grešaka: 1 Program će ispraviti česte greške.
ULAZ
Prvi programi neće biti teški.
IZLAZ
SVE OK
ULAZ
Iva če prvo programirati, a kasnije če se igrati.
IZLAZ
Broj grešaka: 2 Iva će prvo programirati, a kasnije će se igrati.

Zadatak: Pravopis - rješenje

```
unos = input('Unesite Ivinu rečenicu: ')
broj = unos.count(' će ')

if broj == 0:
    print('SVE OK')
else:
    print('Broj grešaka: ', broj)
    print(unos.replace(' će ', ' ću '))
```

pravopis.py

```
================ RESTART: C:/Users/Tomo/Desktop/pravopis.py ============
Unesite Ivinu rečenicu: Program će ispraviti česte greške.
Broj grešaka:  1
Program će ispraviti česte greške.
>>>
================ RESTART: C:/Users/Tomo/Desktop/pravopis.py ============
Unesite Ivinu rečenicu: Prvi programi neće biti teški.
SVE OK
>>>
================ RESTART: C:/Users/Tomo/Desktop/pravopis.py ============
Unesite Ivinu rečenicu: Iva će prvo programirati, a kasnije će se igrati.
Broj grešaka:  2
Iva će prvo programirati, a kasnije će se igrati.
>>>
```

Liste

- Liste su najčešće korištena slijedna zbirka u Pythonu.
- Elementi liste mogu biti različiti tipovi podataka, pa i druge liste.

prazna lista

```
>>> lista_prazna = []
>>> lista_int = [2, 5, 3, 6, 4, 2]
>>> lista_str = ['Futura', 'Python', 'Program']
>>> lista_list = [[2, 5], [3, 6], [4, 2]]
>>> lista_razno = [2, 5, 'Futura', 4.55, 6]
>>> print(lista_razno)
[2, 5, 'Futura', 4.55, 6]
>>> print(lista_list)
[[2, 5], [3, 6], [4, 2]]
```

Liste

- Elementima liste se pristupa preko indeksa (na isti način kao i kod znakovnih nizova):

```
>>> lista_prazna = []
>>> lista_int = [2, 5, 3, 6, 4, 2]
>>> lista_str = ['Futura', 'Python', 'Program']
>>> lista_list = [[2, 5], [3, 6], [4, 2]]
>>> lista_razno = [2, 5, 'Futura', 4.55, 6]
>>> lista_int[1]
5
>>> lista_str[0]
'Futura'
>>> lista_list[2]
[4, 2]
>>> lista_int[1:4]
[5, 3, 6]
```

drugi element liste ima indeks 1

prvi element liste ima indeks 0

Unos elemenata liste s tipkovnice

```
lista = [0.0] * 5

for i in range(5):
    lista[i] = float(input('Unesite broj: '))

for i in range(5):
    print(i, 'element liste:', lista[i])
```

deklaracija liste koja sadrži 5 elemenata i inicijalizacija svih elemenata na vrijednost 0.

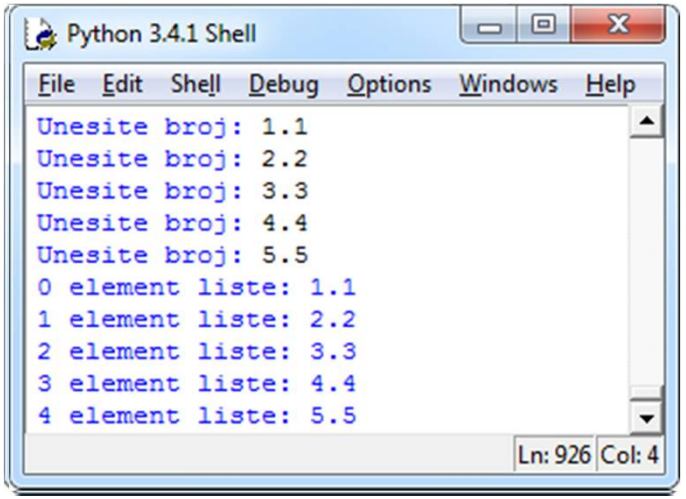
lista.py

unos brojeva s tipkovnice i spremanje u listu pomoću for petlje.

i = 5
kraj petlje

pristup elementima liste pomoću for petlje i ispis vrijednosti na ekran.

1.1	2.2	3.3	4.4	5.5
0	1	2	3	4



Liste i funkcija range()

- Liste se mogu generirati pomoću funkcije range():

```
>>> lista1 = [0 for i in range(11)]
>>> lista1
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
>>> lista2 = [i for i in range(11)]
>>> lista2
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> lista3 = ['a' for i in range(11)]
>>> lista3
['a', 'a', 'a', 'a', 'a', 'a', 'a', 'a', 'a', 'a', 'a']
>>> lista4 = [chr(i) for i in range(ord('a'), ord('z')+1)]
>>> lista4
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k',
'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
'w', 'x', 'y', 'z']
```

Liste i funckija list()

- Funkcija list() može od iterabilnog objekta (znakovni niz, n-torka, skup ili rječnik) napraviti listu:

```
>>> string = 'Futura'
>>> print(string)
Futura
>>> lista5 = list(string)
>>> print(lista5)
['F', 'u', 't', 'u', 'r', 'a']
>>>
>>> lista6 = list(range(1, 20, 2))
>>> lista6
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

Liste i znakovni nizovi

- I liste i znakovni nizovi koriste indekse.
- Razlika: Kod lista je dopušteno mijenjati vrijednosti pojedinih elemenata.

```
>>> string = 'Futura'
>>> string[2]
't'
>>> string[2] = 'T'
Traceback (most recent call last):
  File "<pyshell#34>", line 1, in <module>
    string[2] = 'T'
TypeError: 'str' object does not support item
assignment
```

Liste i znakovni nizovi

- Znakovni nizovi su nepromjenivi (engl. *immutable*) tip podataka.
- Liste su promjenivi (engl. *mutable*) tip podataka.

```
>>> lista_int = [2, 5, 3, 6, 4, 2]
>>> lista_int[2]
3
>>> lista_int[2] = 33
>>> lista_int
[2, 5, 33, 6, 4, 2]
>>>
```

promjena vrijednosti elementa liste

Operatori za liste

- Postoje 4 binarna operatora za rad s listama (isti kao za znakovne nizove):

operator	opis djelovanja
`+`	spajanje (nadovezivanje)
`*`	uvišestručenje – jedan operand je tipa `int`
`in`	element je sadržan u listi
`not in`	element nije sadržan u listi

- Uvišestručenje:

kod uvišestručenja vrijedi zakon komutativnosti

```
>>> lista1 = [1, 2, 3]
>>> lista2 = lista1 * 2
>>> lista2
[1, 2, 3, 1, 2, 3]
>>> print(2 * lista2)
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Operatori za liste

```
>>> lista1 = [1, 2, 3, 4]
>>> lista2 = [5, 6, 7]
>>> lista3 = lista1 + lista2
>>> lista3
[1, 2, 3, 4, 5, 6, 7]
>>> 3 in lista1
True
>>> 3 not in lista1
False
>>> 6 in lista2
True
>>> 6 not in lista2
False
```

spajanje lista: koristi se standardni operator za zbrajanje: +

svi operandi su liste

rezultat izraza je istina ili laž: True ili False

Ugrađene funckije za liste

- Za rad s listama postoji više ugrađenih funkcija:

funkcija	opis djelovanja
`len(L1)`	vraća duljinu liste **L1**
`min(L1)`	vraća najmanju vrijednost elementa liste **L1**
`max(L1)`	vraća najveću vrijednost elementa liste **L1**
`sum(L1)`	vraća sumu vrijednosti svih elemenata liste **L1**
`del(L1[i])`	iz liste **L1** uklanja element s indeksom **i**
`del(L1[i:j])`	iz liste **L1** uklanja isječak koji započinje s indeksom **i**, a završava s indeksom **j-1**

Ugrađene funckije za liste

```
>>> lista = [1, 2, 3, 4, 5, 6]
>>>
>>> len(lista)
6
>>>
>>> min(lista)
1
>>>
>>> max(lista)
6
>>>
>>> sum(lista)
21
```

```
>>> del(lista[2])
>>> lista
[1, 2, 4, 5, 6]
>>> len(lista)
5
>>>
>>> del(lista[1:3])
>>> print(lista)
[1, 5, 6]
>>> len(lista)
3
```

Zadatak: Natjecanje

- Unijeti broj učenika koji se natječu na školskom natjecanju iz programiranja. Za svakog učenika u istom redu unijeti ime i broj ostvarenih bodova (0-200).
- Unijeti broj bodova potreban za plasman na županijsko natjecanje.
- Maknuti s popisa sve učenike koji se nisu plasirali na županijsko natjecanje i ispisati:
  - koliko učenika se plasiralo na županijsko natjecanje
  - koliki su prosječan broj bodova ostvarili ti učenici

Primjeri testnih podataka

ULAZ	ULAZ
5 Ana 140 Marko 200 Pero 90 Ivana 150 Petra 85 100	9 Matej 190 Đive 20 Franko 90 Iva 160 Pero 120 Lovro 30 Mateo 50 Dora 160 Božo 180 90
IZLAZ	**IZLAZ**
3 163.333	6 150.0

Zadatak: Natjecanje - rješenje

```
n = int(input('Unesite broj učenika: '))
ime = [''] * n
bod = [0] * n
for i in range(n):
    unos = input('Unesi ime i broj bodova: ')
    ime[i], bod[i] = unos.split()
    bod[i] = int(bod[i])
prag = int(input('Prag za županijsko: '))
j = 0
while j < len(bod):
    if bod[j] < prag:
        del(ime[j])
        del(bod[j])
    else:
        j += 1
print('Broj učenika:', len(ime))
print('Prosjek bodova:', sum(bod)/len(bod))
```

natjecanje.py

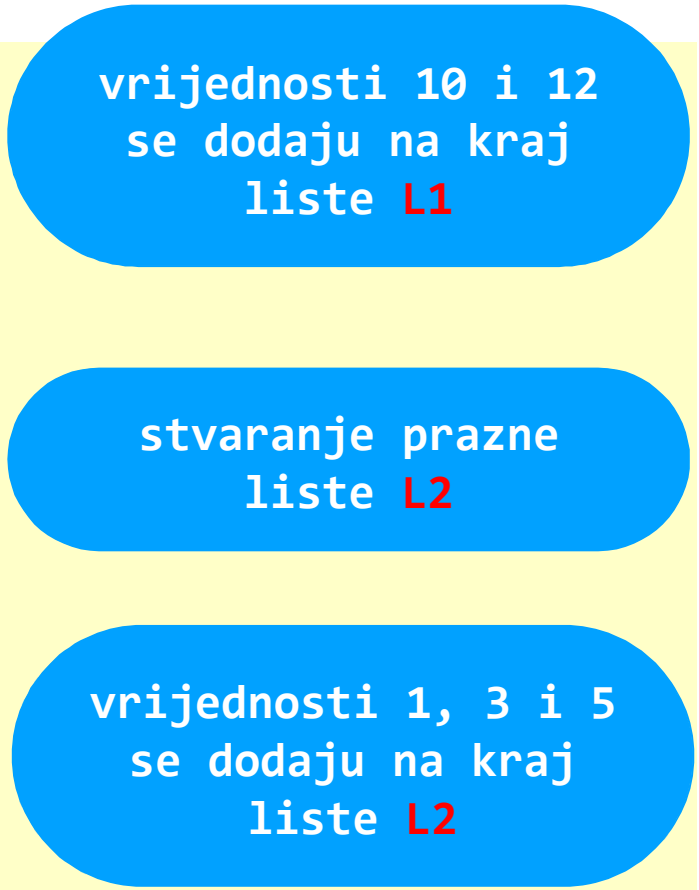
Ugrađene metode za znakovne nizove

- Postoji i više ugrađenih metoda za liste:

metoda	opis djelovanja
`L1.append(x)`	dodaje element **x** na kraju liste **L1**
`L1.insert(i, x)`	umeće element **x** prije **i-tog** elementa liste **L1**
`L1.remove(x)`	izbacuje element **x** (x je vrijednost elementa)
`L1.pop(i)`	vraća i izbacuje **i-ti** element iz liste **L1**, ako parametar **i** nije naveden vraća i izbacuje zadnji
`L1.reverse()`	okreće redoslijed elemenata liste **L1**
`L1.sort()`	sortira listu **L1** od najmanje do najveće vrijednosti elementa
`L1.count(x)`	vraća broj koliko se puta element **x** pojavljuje u listi **L1**

Ugrađene metode za liste

```
>>> L1 = [2, 4, 6, 8]
>>> L1.append(10)
>>> L1.append(12)
>>> print(L1)
[2, 4, 6, 8, 10, 12]
>>>
>>> L2 = []
>>> L2.append(1)
>>> L2.append(3)
>>> L2.append(5)
>>> print(L2)
[1, 3, 5]
>>>
```



Ugrađene metode za liste

```
>>> L1 = [2, 4, 6, 8]
>>> L1.insert(2, 5)
>>> print(L1)
[2, 4, 5, 6, 8]
>>> L1.insert(10, 1)
>>> print(L1)
[2, 4, 5, 6, 8, 1]
>>> L1.remove(5)
>>> print(L1)
[2, 4, 6, 8, 1]
>>> L1.pop(2)
6
>>> L1.pop()
1
>>> print(L1)
[2, 4, 8]
```

vrijednost 5 se dodaje prije 3. elementa (koji ima indeks 2)

ako je indeks veći od indeksa zadnjeg elementa, dodaje se na kraj liste

izbacuje se element koji ima vrijednost 5

vraća se i izbacuje element s indeksom 2

ako nije naznačen indeks vraća se i izbacuje zadnji element

Ugrađene metode za liste

```
>>> L3 = [4, 2, 8, 6]
>>> L4 = ['b', 'e', 'u', 'a', 'p']
>>> L3.reverse()
>>> print(L3)
[6, 8, 2, 4]
>>> L4.reverse()
>>> print(L4)
['p', 'a', 'u', 'e', 'b']
>>> L3.sort()
>>> print(L3)
[2, 4, 6, 8]
>>> L4.sort()
>>> print(L4)
['a', 'b', 'e', 'p', 'u']
```

okreće se redoslijed elemenata u listama L3 i L4

liste L3 i L4 se sortiraju (od manjih prema većim vrijednostima elemenata)

Zadatak: Pozitivni brojevi

- Ana se voli igrati s pozitivnim brojevima, pa joj je potreban program u kojem se unose brojevi sve dok se ne unese negativni broj.
- Unesene pozitivne brojeve treba sortirati od većeg prema manjem.
- Od unesenih brojeva stvoriti dvije nove liste.
- U jednu je potrebno spremiti sve parne brojeve, a u drugu sve neparne brojeve.
- Ispisati sve tri liste.

Primjer izvođenja programa

```
Unesi broj: 229
Unesi broj: 104
Unesi broj: 263
Unesi broj: 88
Unesi broj: 395
Unesi broj: 215
Unesi broj: 123
Unesi broj: -1
Lista: [395, 263, 229, 215, 123, 104, 88]
Parni brojevi: [104, 88]
Neparni brojevi: [395, 263, 229, 215, 123]
```

Zadatak: Pozitivni brojevi - rješenje

```
lista = []
broj = int(input('Unesi broj: '))
while broj > 0:
    lista.append(broj)
    broj = int(input('Unesi broj: '))
lista.sort()
lista.reverse()

parni, neparni = [], []
for br in lista:
    if br%2 == 0:
        parni.append(br)
    else:
        neparni.append(br)
print('Lista:', lista)
print('Parni brojevi:', parni)
print('Neparni brojevi:', neparni)
```

pozitivnibrojevi.py