

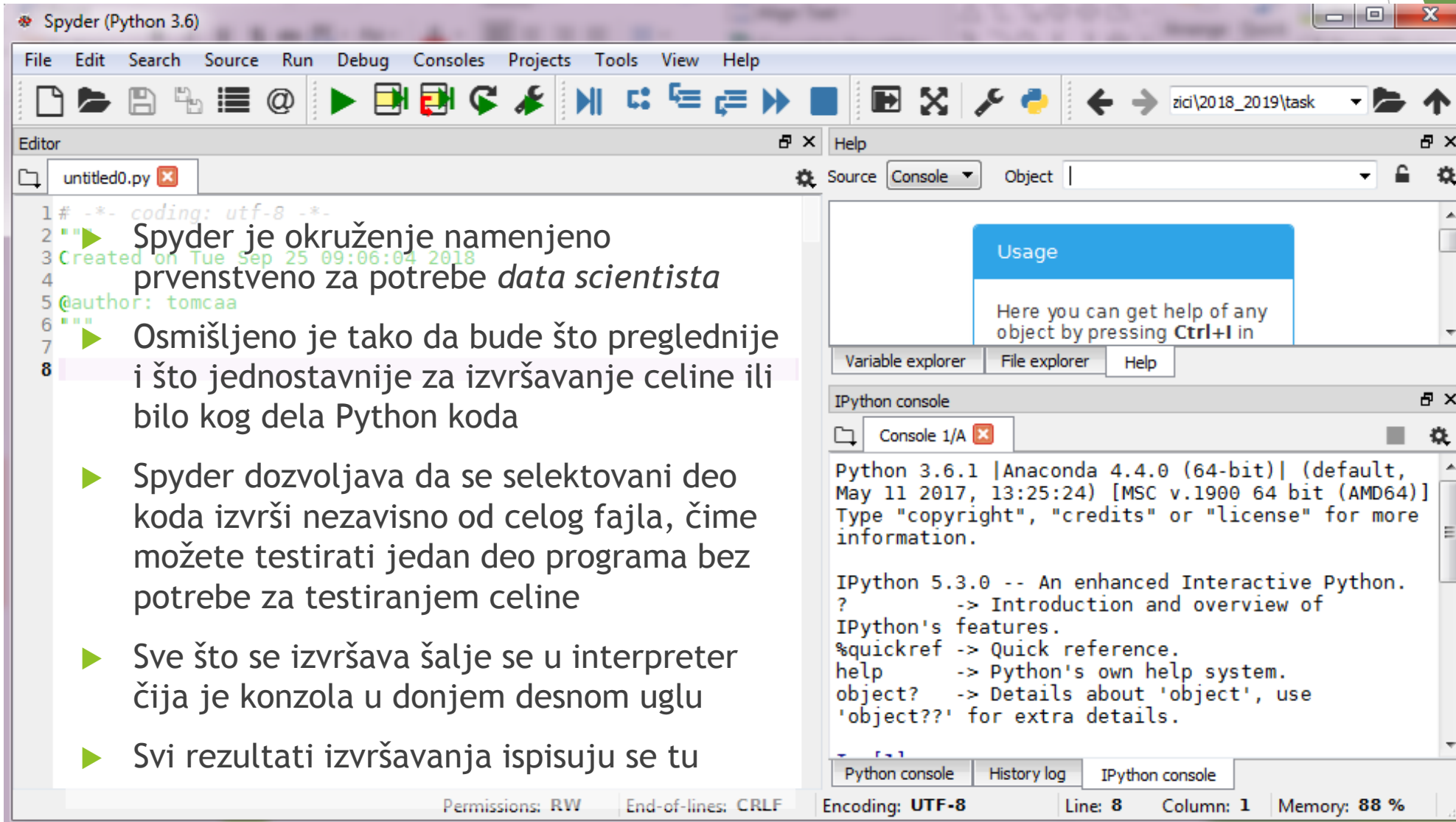
Skript jezici V01 – Uvod u Python

Milan Tomić

Python

- ▶ Osnovni programski konstrukti: komentari, promenljive, operatori, grananja i petlje, funkcije, moduli i paketi

Okruženje - Spyder



The screenshot displays the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.6)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains icons for file operations, running, and debugging. The Editor pane shows a file named "untitled0.py" with the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Sep 25 09:06:04 2018
4 prvenstveno za potrebe data scientista
5 @author: tomcaa
6 """
7
8
```

Overlaid on the code editor is a text box with the following content:

► Spyder je okruženje namenjeno prvenstveno za potrebe *data scientista*

► Osmišljeno je tako da bude što preglednije i što jednostavnije za izvršavanje celine ili bilo kog dela Python koda

► Spyder dozvoljava da se selektovani deo koda izvrši nezavisno od celog fajla, čime možete testirati jedan deo programa bez potrebe za testiranjem celine

► Sve što se izvršava šalje se u interpreter čija je konzola u donjem desnom uglu

► Svi rezultati izvršavanja ispisuju se tu

The right-hand side of the interface contains the Variable explorer, File explorer, and Help tabs. The Python console is open, showing the following output:

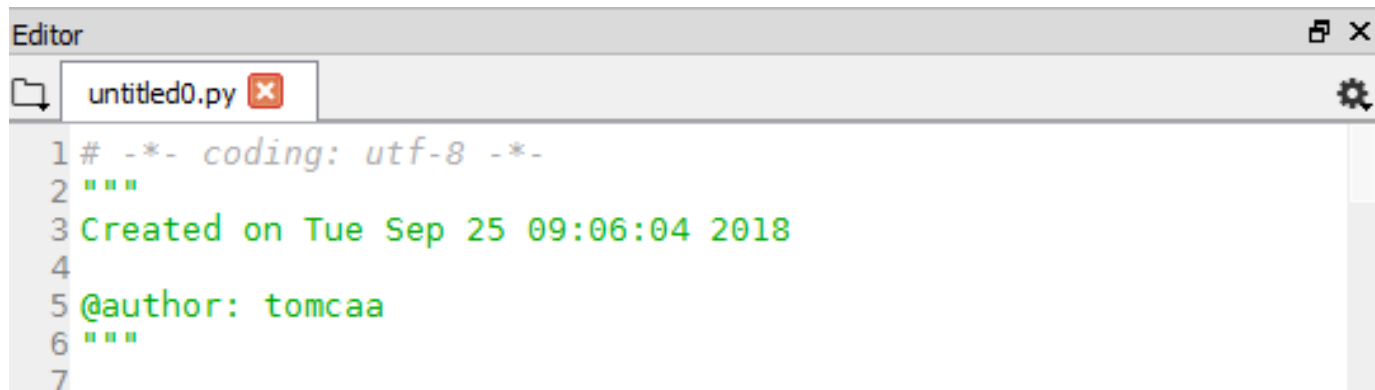
```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
```

The status bar at the bottom indicates: Permissions: RW, End-of-lines: CRLF, Encoding: UTF-8, Line: 8, Column: 1, Memory: 88 %.

Okruženje - Spyder

- ▶ Kada se Spyder pokrene, obično je u Editoru otvoren prvi fajl - template

A screenshot of the Spyder IDE's Editor window. The window title is 'Editor'. Below the title bar is a tab labeled 'untitled0.py' with a red close button. The editor area shows a Python template with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Sep 25 09:06:04 2018
4
5 @author: tomcaa
6 """
7
```

- ▶ Prva linija je hint interpreteru da je kod pisan sa enkodingom utf-8
- ▶ Ostale linije su višelinijski string - docstring (dokumentacija) - u kojoj stoji opis modula (docstring se piše na početku modula, klase ili funkcije)

Okruženje - Spyder

- ▶ Pored toga, u donjem desnom uglu je IPython konzola u kojoj će se izvršavati sve što se izvršava u editoru (mada se može koristiti i nezavisno od editora jer je u interaktivnom modu)

```
print('Skript jezici')
```

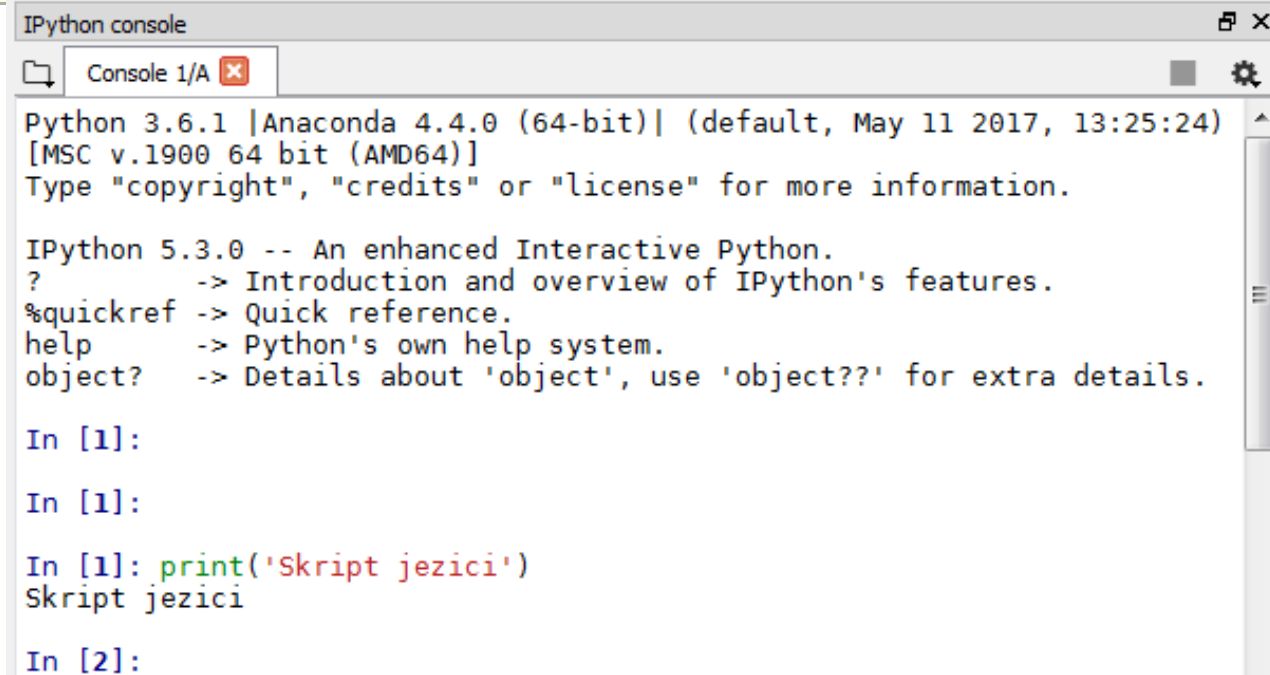
- ▶ Dodaćemo ovu liniju koda u editor
- ▶ Kod se može pokrenuti pritiskom na Run (F5) ili selektovanjem dela koda i pritiskom na Run current cell (Ctrl+Enter)
- ▶ „Ćelije” su podržane i ubacivanjem specijalnog komentara `#%` na početku linije i služe za izvršavanje delova koda iz editora korak po korak (inače nemaju smisla)
- ▶ Biće korišćene u nastavku kursa za pokazivanje mogućnosti Pythona

Okruženje - Spyder

► Pokretanje prve ćelije

```
#%% Print funkcija služi za ispis nečega na konzolu (u ovom slučaju string)  
print('Skript jezici')
```

- Na konzoli se ispisuje izvršena naredba kao ulaz (In) i ispod nje sve što je ispisano na standardni izlaz
- Ako izvršena naredba vraća neki rezultat, u konzoli će on biti posebno ispisan kao Out
- Broj u zagradama je redni broj naredbe koja se zadaje (služi i za povezivanje odgovarajućeg izlaza sa ulazom)



```
IPython console  
Console 1/A  
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24)  
[MSC v.1900 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.  
  
IPython 5.3.0 -- An enhanced Interactive Python.  
?      -> Introduction and overview of IPython's features.  
%quickref -> Quick reference.  
help    -> Python's own help system.  
object? -> Details about 'object', use 'object??' for extra details.  
  
In [1]:  
  
In [1]:  
  
In [1]: print('Skript jezici')  
Skript jezici  
  
In [2]:
```

Komentari

- ▶ Jednolinijski komentari u Pythonu se pišu iza znaka `#` i traju od tog znaka do kraja reda
- ▶ Dokumentacija modula, klase ili funkcije može se ostaviti na početku (ispod deklaracije) u višelinijskom stringu, koji počinje sa tri znaka navoda i završava se sa tri znaka navoda (kao u prvom primeru)
- ▶ Višelinijski string je moguće koristiti i kao komentar bilo gde drugo u kodu, što interpreter gleda kao naredbu definicije običnog stringa (bez ispisa), pa bi interpreter samo za tu naredbu mogao kao `Out` da ispiše taj string (ovo inače nema nikakav uticaj na izvršavanje ostatka koda)
- ▶ Pisanje docstringa će biti pojašnjeno dodatno u okviru funkcija i klasa

Promenljive

- ▶ Python podržava različite tipove podataka ali je dinamički tipiziran - promenljivama se dodeljuje tip u skladu sa trenutnim sadržajem
- ▶ Možete posmatrati promenljive kao pokazivače na neke objekte u memoriji (bez potrebe za dereferenciranjem), pri čemu se pokazivač u bilo kom trenutku može pomeriti na bilo koji objekat, dok je odgovarajući tip svojstvo samog objekta
- ▶ Python ne zahteva deklaraciju promenljivih, već se one uvode u prostor imena onog trenutka kada se prvi put iskoriste (tj. kada im se dodeli neka vrednost)

Promenljive

► Primeri:

Promenljive su dinamičke i nisu vezane za tip; vrednosti imaju tip

```
a = 4
b = 2
c = a + b
print(c)  # 6
```

Šta se dešava kada im promenimo vrednosti?

```
a = '4'
b = '2'
c = a + b
print(c)  # 42
```

Šta se dešava kada pomešamo tipove?

```
b = 2
c = a + b  # TypeError: must be str, not int
```

Kastovanje

```
c = int(a) + b
print(c)  # 6
```

Kastovanje

- ▶ U Pythonu je moguća jednostavna konverzija iz jednog primitivnog tipa u drugi pomoću kastovanja
 - ▶ Zapravo je koncept primitivnog tipa napušten i od početka Python-a 3 su sve objekti, a ono što nazivamo kastovanjem su zapravo parametrizovani konstruktori
 - ▶ To možda ne izgleda tako i teško ga je shvatiti u ovom trenutku
- ▶ Pozivom funkcija kao što su *int*, *float*, *str* možemo konvertovati vrednost neke promenljive u ceo broj, realni broj ili string
 - ▶ Postoje i drugi tipovi, sa kojima ćemo raditi kasnije, kod kojih postoji slična šema ali je nešto komplikovanije
 - ▶ Takođe, pri konverziji u celobrojni tip iz stringa može se koristiti druga osnova

Aritmetički operatori

- ▶ Python 3.6 podržava sledeće binarne aritmetičke operatore (između ostalog):
 - ▶ `+` (`__add__`) - sabiranje
 - ▶ `-` (`__sub__`) - oduzimanje
 - ▶ `*` (`__mul__`) - množenje
 - ▶ `/` (`__truediv__`) - deljenje (realno)
 - ▶ `//` (`__floordiv__`) - deljenje (celobrojno, zaokruženo na dole)
 - ▶ `%` (`__mod__`) - ostatak pri celobrojnem deljenju
 - ▶ `**` (`__pow__`) - stepenovanje

Aritmetički operatori

```
#%% Aritmetički operatori nad brojevima  
a = 5  
b = 3  
  
print('a + b = ', a + b) # 8  
print('a - b = ', a - b) # 2  
print('a * b = ', a * b) # 15  
print('a / b = ', a / b) # 1.6666666666666667  
print('a // b = ', a // b) # 1  
print('a % b = ', a % b) # 2  
print('a ** b = ', a ** b) # 125
```

Aritmetički operatori

- ▶ Isti operatori ne moraju da imaju isto značenje za različite tipove
 - ▶ Nadjačavanje operatora ćemo raditi kasnije u okviru OOP-a
- ▶ Primer - sabiranje je definisano za stringove, za rezultat ima nadovezane stringove (već viđeno)
- ▶ Još jedan neobičan primer je množenje stringa brojem

Operatori dodele i poređenja

- ▶ Operatori dodele su slični kao i u drugim jezicima
 - ▶ Osnovni operator dodele vrednosti je =
 - ▶ Operatori dodele uz uzimanje prethodnog rezultata kao prvog operanda binarne operacije su takođe podržani: +=, -=, *=, /=, %= itd.
- ▶ Operatori poređenja su takođe slični kao i u drugim jezicima
 - ▶ Poređenje za jednakost: == (`__eq__`)
 - ▶ Poređenje za nejednakost (različitost): != (`__ne__`)
 - ▶ Druge relacije poretka: < (`__lt__`), <= (`__le__`), > (`__gt__`), >= (`__ge__`)
- ▶ Poređenje po identitetu objekta
 - ▶ is, is not - proverava da li dve promenljive imaju isti objekat kao vrednost
 - ▶ Uporediti sa `.equals()` metodom u Javi, koja se razlikuje od ==

Python istinitosne vrednosti

- ▶ Python ima boolean tip (bool), čije su moguće vrednosti literali True i False (obratite pažnju na velika slova!)
- ▶ Za testiranje istinitosti važe sledeća pravila*:
 - ▶ Konstante None i False su netačne
 - ▶ Nule numeričkih tipova su netačne (0, 0.0 itd.)
 - ▶ Prazni stringovi, nizovi i kolekcije su netačni ("" itd.)

Python poređenja

- ▶ Objekti različitih tipova, osim različitih numeričkih tipova, nikada nisu jednaki
- ▶ Poređenja po poretku (<, >, <=, >=) mogu da uspeju samo ukoliko je poredak definisan, te ako se porede objekti različitih tipova može često da se javi `TypeError`
- ▶ Dve instance klase koje nisu identične obično su različite (tj. ne važi `==` osim ako se ne definiše drugačije)
- ▶ Poredak između objekata jedne klase može da postoji ako se odgovarajući operatori nadjačaju
- ▶ Način rada `is` i `is not` operatora ne može da se promeni i oni su uvek primenljivi

Bitovni operatori

- ▶ Python podpira bitovne operatore za celobrojne vrednosti
 - ▶ `<<` (`__lshift__`)
 - ▶ `>>` (`__rshift__`)
 - ▶ `&` (`__and__`)
 - ▶ `^` (`__xor__`)
 - ▶ `|` (`__or__`)

Kontrola toka

- ▶ Kompleksnije programske strukture teško je izvesti bez naredbi za kontrolu toka
- ▶ Python podržava standardne operacije grananja i petlji:
 - ▶ `if ... elif ... else ...`
 - ▶ `... if ... else ...`
 - ▶ `for`
 - ▶ `while`
- ▶ Zadati uslov se proverava za istinitost i ako je ispunjen, onda se prelazi na izvršenje bloka
- ▶ Blokovi koda za izvršavanje se zadaju poravnanjem (uvlačenjem redova)

Naredba if-elif-else

- ▶ Naredba if služi za izvršavanje koda ako je uslov ispunjen
- ▶ Dopuna može biti elif (skraćeno od else if, ali nije isto) - izvršavanje ako prvi uslov nije ispunjen a naredni jeste
- ▶ Dopuna može biti else - izvršavanje ako nijedan uslov (prvi if i svaki naredni elif) nije zadovoljen
- ▶ Blok if-elif-elif-elif-else može da se protumači nalik na malo kompleksniju verziju switch bloka u drugim jezicima

Primer if

```
### Operatori poredenja, if naredba
if 23 * 4 > 15:
    print('If je tacan') # If je tacan

if '':
    print('If je tacan')
else:
    print('If je netacan') # If je netacan

if 23 * 4 < 92:
    print('If je tacan')
elif 23 * 4 == 92:
    print('Elif je tacan')
else:
    print('If je netacan') # Elif je tacan
```

... if ... else ...

- ▶ Na poseban način je moguće koristiti if kod dodele vrednosti ili nekim specijalnim okolnostima za odlučivanje (nalik na ternarni operator u drugim jezicima $A ? B : C$)
- ▶ Prvo se izračunava uslov, zatim na osnovu njega dodeljuje vrednost
- ▶ Primer:

```
#%% Ternarni operator  
a = 5  
b = 3  
c = b if a > b else a  
print(c) # 3
```

Naredba for

- ▶ U Pythonu je svaka for petlja „for each” tipa, tj. dešava se za svaki element u nekom iterabilnom objektu
 - ▶ Objekt kroz koji se iterira može biti kolekcija drugih objekata (string, lista, rečnik, skup...), ili objekt koji generiše elemente do određenog trenutka (generator)

Primer naredbe for (iteracija kroz string)

```
#%% Naredba for  
c = ''  
for slovo in 'PRIMER':  
    c = c + slovo + ', '  
print(c)  # P, R, I, M, E, R,
```

Generisani brojevi iz opsega vrednosti - range

- ▶ Za potrebe generisanja niza brojeva od početne do krajnje vrednosti koristi se specijalni generator objekat - range
- ▶ Range objekat je objekat koji kada se kroz njega iterira generiše brojeve po zadatom kriterijumu
- ▶ To nam omogućuje da simuliramo klasičnu for petlju (for i=0 to n-1)
- ▶ Range objekat ima tri načina korišćenja:
 - ▶ `range(end)` generiše brojeve od 0 do end-1
 - ▶ `range(start, end)` generiše brojeve od start do end-1
 - ▶ `range(start, end, step)` generiše brojeve od start do end (ne uključujući end) sa inkrementiranjem step

Primer naredbe for sa range-om

```
### Naredba for in range  
c = ''  
for i in range(10):  
    c += str(i) + ', '  
print(c)  # 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,  
  
c = 0  
for i in range(1, 101):  
    c += i  
print(c)  # 5050  
  
c = ''  
for i in range(1, 11, 2):  
    c += str(i) + ', '  
print(c)  # 1, 3, 5, 7, 9, |
```

Naredba while

- ▶ Naredba while je naredba koja izvršava petlju sve dok je neki uslov ispunjen
- ▶ Primer:

```
### Naredba while  
c = 5  
while c > 0:  
    c -= 1  
else:  
    print(c)  # 0|
```

- ▶ Naredbe while i for mogu imati završetak u obliku else uslova, koji se izvršava kada uslov više nije ispunjen (dodatak sintaksi koji uglavnom ne postoji u drugim jezicima)

Naredbe continue i break

- ▶ Kao i u drugim jezicima, continue i break mogu da služe za kontrolu toka petlje
- ▶ Naredba continue služi da nastavi sa iteracijom od sledećeg elementa
- ▶ Naredba break služi da u potpunosti prekine petlju
- ▶ Ako petlja ima else, on se neće izvršiti ukoliko dođe do prekida pomoću break naredbe

Naredbe continue i break - primer

```
#%% Naredbe continue i break
s = ''
for c in range(10, -10, -1):
    if c % 3 == 0:
        continue
    if c % 2 == 0:
        s += str(c) + ', '
    if c == 1:
        break
else:
    print(c)
print(s) # 10, 8, 4, 2,
```

Funkcije

- ▶ Funkcije kao i u drugim jezicima predstavljaju organizovane celine koda koje izvršavaju neku često korišćenu operaciju i mogu da vraćaju rezultat te operacije kao povratnu vrednost
- ▶ Ranije smo već videli kako se pozivaju funkcije (isto kao i u većini jezika - naziv funkcije i argumenti u zagradama)
 - ▶ Koristili smo `print(x)`, `int(x)`, `range(x)` itd.
- ▶ Definicija funkcije nije ništa komplikovanija
 - ▶ Zaglavlje funkcije počinje sa „`def nazivFunkcije(arg1, arg2, arg3):`“
 - ▶ Ispod ove linije može da se piše docstring
 - ▶ Nakon toga sledi (uvučeni) blok koda koji predstavlja telo funkcije
 - ▶ Ključna reč `return` vraća vrednost funkcije

Primer definicije funkcije sa pozivima

```
### Primer funkcije

def zbir_u_range(start, end):
    """ Vraca zbir svih brojeva od start do end-1 """
    s = 0
    for i in range(start, end):
        s += i
    return s

print(zbir_u_range(1, 101)) # 5050
print(zbir_u_range(1, 11)) # 55
```

Standardni ulaz (stdin)

- ▶ Od korisnika se u konzolnoj aplikaciji može očekivati da unese neki podatak
- ▶ Funkcija `input` služi za učitavanje jedne linije sa standardnog ulaza
 - ▶ Kao argument može da primi prompt - upit korisniku koji se ispisuje na ekranu pre učitavanja
 - ▶ Kao rezultat uvek vraća string
- ▶ Ako želimo da učitamo broj sa standardnog ulaza, moramo kastovati učitani string u broj
- ▶ Takođe, funkcija `eval` (koju inače treba izbegavati pri radu sa standardnim ulazom jer može da izvrši bilo koji kod) ima mogućnost da automatski evaluira uneti podatak i pretvori ga u odgovarajući tip

Primer učitavanja sa standardnog ulaza

```
### Standardni ulaz  
a = int(input('Unesite ceo broj: '))  # 20  
b = input()  # 4  
c = eval(input('Unesite izraz: '))  # 28 + a  
print('a =', a)  # a = 20  
print('b =', b)  # b = 4  
print('c =', c)  # c = 48
```


Tip string - str

- ▶ String predstavlja nisku simbola i zapisuje se između navodnika ili apostrofa
- ▶ Stringovi nisu mutabilni - jednom definisani string ne može da promeni ni dužinu ni sadržinu
- ▶ Dužina stringa može da se odredi pozivom funkcije len
 - ▶ Ista funkcija određuje i dužinu odnosno veličinu kolekcija
- ▶ Slovim stringa može da se pristupa operatorom indeksiranja - uglastim zagradama
 - ▶ Npr. `a[3]` predstavlja četvrti simbol u stringu (indeksi se broje od 0)
 - ▶ Negativni indeks predstavlja brojanje od kraja
 - ▶ Isti operator koristi se za pristup elementima kolekcija

Tip string - str - podstringovi i slice

- ▶ Podstringovi mogu da se dobiju operatorom isecanja - slice
 - ▶ Liči na indeksiranje, ali se zadaje opseg, npr. `a[start:end:step]` je podstring od indeksa start do indeksa end sa inkrementom step (ne uključujući end, kao u range)
 - ▶ Isti operator može da se koristi i u nekim kolekcijama
- ▶ Ako se u slice ne navede početak, uzima se početak stringa
 - ▶ Ako se ne navede kraj, uzima se kraj stringa
 - ▶ Podrazumevani step je 1

Tip string - primeri

```
### Tip string - primeri
s = 'Ana voli\nMilovana'
print(s)      # Ana voli
               # Milovana
s = 'Ana voli Milovana'
print(len(s)) # 17
print('Osmo slovo:', s[9]) # M
print('Podstring [9:16]:', s[9:16]) # Milovan
print('Svako drugo slovo:', s[::2]) # Aavl ioaa
print('Do petog slova otpozadi:', s[0:-5]) # Ana voli Mil
print('Unatraske:', s[::-1]) # anavoliM ilov anA
```

Metode stringova

- ▶ Kao što smo ranije rekli, sve u Pythonu su objekti, pa i stringovi
- ▶ str klasa ima dosta korisnih metoda (ovo nije konačna lista, `help(str)` za više)
- ▶ Među njima su i:
 - ▶ `s.count(a)` - broj pojavljivanja podstringa a u stringu s
 - ▶ `s.find(a)` - pozicija podstringa a u stringu s, -1 ako nije pronađen
 - ▶ `s.index(a)` - pozicija podstringa a u stringu s, greška `ValueError` ako nije pronađen
 - ▶ `s.lower()` - vraća kopiju s ispisanu malim slovima
 - ▶ `s.upper()` - vraća kopiju s ispisanu velikim slovima
 - ▶ `s.replace(x, y)` - vraća kopiju s u kojoj je svako pojavljivanje x zamenjeno sa y
 - ▶ `s.strip()` - uklanja razmake sa početka i kraja stringa (takođe vidi `lstrip`, `rstrip`)

Metode stringova - primer

```
### Tip string - metode
s = '    Ana voli Milovana '
print(len(s))    # 22
s = s.strip()
print(len(s))    # 17
print(s.count('ana'))    # 1
s = s.lower()
print(s.count('ana'))    # 2
print(s.index('ana'))    # 0
# Traži drugo pojavljivanje podreči 'ana':
print(s.index('ana', s.index('ana') + 1))    # 14
```

Datoteke

- ▶ Python daje jednostavan interfejs za rad sa datotekama, preko file streamova koji su objekti
- ▶ Za otvaranje datoteke koristi se funkcija `open()`
 - ▶ Kao argumenti prosleđuju se putanja do datoteke i mod (čitanje - `r`, pisanje - `w`, dodavanje - `a`)
 - ▶ Funkcija vraća stream otvorene datoteke

Datoteke - upisivanje

- ▶ Za upisivanje u datoteku (kada je u modu w ili a) koriste se metode write i writelines (druga metoda upisuje listu stringova u zasebnim linijama, više o tome kad budemo radili liste)
 - ▶ write(text) - ispisuje text u stream, vraća broj ispisanih znakova (dužinu stringa)
 - ▶ writelines(lines) - ispisuje listu stringova u zasebnim linijama
- ▶ Moguće je upisivati u datoteku i pomoću funkcije print
 - ▶ Kao argument file funkciji print treba proslediti stream (vidi primer)
- ▶ Za pražnjenje streama u datoteku koristi se metoda flush
- ▶ Za zatvaranje datoteke koristi se metoda close

Upisivanje u datoteku - primer

```
## Upisivanje u datoteku

# otvaranje datoteke
f = open('test.txt', 'w')

# upisivanje
f.write('Ovo je test\nVišelinijski string')

# upisivanje preko printa
print('\nOvo je drugi test', file=f)

# zatvaranje datoteke
f.close()
```


Datoteke - čitanje

- ▶ Čitanje iz datoteke je takođe relativno jednostavno, a izvodi se pomoću metoda `read`, `readline` i `readlines`
- ▶ `File` ima tri metode za čitanje:
 - ▶ `read(size)` - čita najviše `size` znakova iz streama; po defaultu čita do EOF (ako se prosledi `size < 0` ili se ne prosledi); vraća pročitani string
 - ▶ `readline(size)` - isto kao `read`, samo što može prekinuti čitanje i na `\n`; vraća pročitani string
 - ▶ `readlines(hint)` - vraća listu linija pročitanih iz streama; `hint` može da se upotrebi da se prekine čitanje ako se pročita više od `hint` bajtova sa streama (ako se prosledi `hint > 0`, inače ako se ne prosledi čita do EOF)
- ▶ Takođe ako je potrebno čitanje liniju po liniju u petlji, moguće je iterirati kroz fajl pomoću for petlje
- ▶ Primetite da svaka pročitana linija sadrži i `\n` na kraju (ako ga ima i u datoteci), te je potrebno pozvati `strip()` na stringu zbog suvišnih razmaka

Čitanje iz datoteke - primer

```
### Čitanje iz datoteke

f = open('test.txt', 'r')

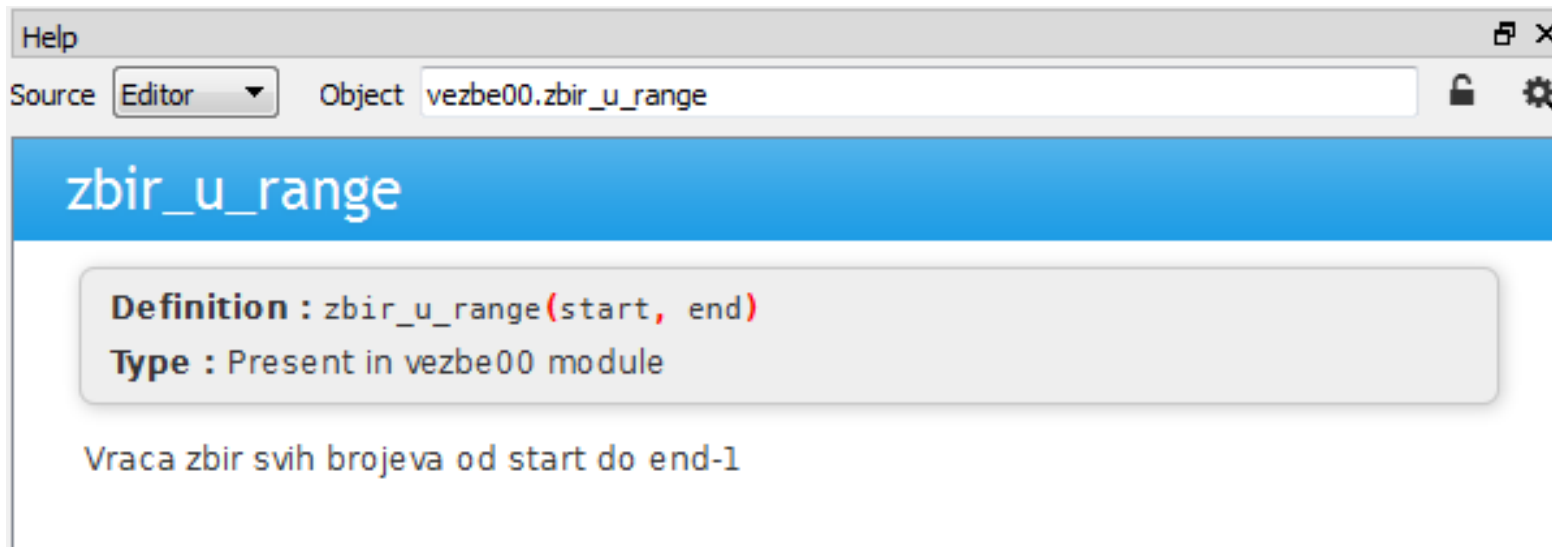
# prvi red
s = f.readline()
print(s) # Ovo je test\n
# ostali redovi
for s in f:
    print(s) # Višelinijski string\n
              # Ovo je drugi test\n
f.close()
|
```

Gledanje dokumentacije iz interaktivnog moda i iz Spydera

- ▶ Funkcije, moduli, klase idr. koje imaju pridružene docstringove mogu biti pregledani za pomoć
- ▶ Python ima ugrađenu interaktivnu funkciju `help`, koja služi da izlista docstring traženog objekta (tipa promenljive, funkcije, klase, metode itd.)
- ▶ U Spyder editoru postoji Help prozor koji ima istu svrhu (kada pozicionirate kursor na željeni objekat i pritisnete `Ctrl+I`, pomoć o tom objektu će vam se pojaviti u prozoru)

Primer traženja pomoći

U Spyderu



U IPython konzoli

```
In [43]: help(zbir_u_range)
Help on function zbir_u_range in module __main__:

zbir_u_range(start, end)
    Vraca zbir svih brojeva od start do end-1
```

Zadaci za samostalan rad

- ▶ Vezbe01_zadaci.pdf