

Programska potpora komunikacijskim sustavima

• Izv. prof. dr. sc. Goran Delač

Uvod u programski jezik Python



Sadržaj predavanja

- Uvod
- Osnove jezika
- Tipovi podataka

Uvod



O programskom jeziku Python

- Programski jezik opće namjene
- Brz i jednostavan razvoj primjenskih programa
 - **Skriptni** jezik
 - **Interpretirani** jezik
 - Visoka razina apstrakcije
 - Automatsko upravljanje memorijom
 - **Dinamički tipovi** podataka

O programskom jeziku Python

- Podržano više programskih paradigmi
 - Objektno orijentirano programiranje
 - Imperativno programiranje
 - Funkcijsko programiranje
- Bogata programska knjižnica
 - **Standardna** knjižnica
 - Strojno učenje, obrada slika, ugrađeni sustavi, web aplikacije...

Popularnost Pythona (2022.)

■ GitHub

- 1. Javascript, 2. **Python**, 3. Java
- <https://octoverse.github.com/#top-languages-over-the-years>

■ Tiobe

- 1. **Python**, 2. C, 3. Java, 4. C++
- <https://www.tiobe.com/tiobe-index/>

■ PYPL

- 1. **Python** (29% udjela), 2. Java (18% udjela), 3. JavaScript
- <https://pypl.github.io/PYPL.html>

Povijest

- Začetak u Nizozemskoj, kasne 1980te
 - Guido van Rossum
 - Izvor imena je serija „Monty Python's Flying Circus”
- Prva verzija – veljača 1991
- Python 2.0 – 2000. godine
 - Python Software Foundation (2001. godine)
- Python 3.0 – 2008. godine
 - Veći broj poboljšanja
 - Nije kompatibilan s verzijama 2.x

Povijest

“Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered.”

- Guido van Rossum



Prednosti i nedostaci

■ Prednosti:

- Jednostavnost korištenja – potrebno je malo prethodnog znanja
 - Dinamički tipovi podataka, interaktivno izvođenje ...
- Bogata programska knjižnica
- Čitljivost programskog koda – sintaksa jezika podređena čitljivosti, de-facto standardi

■ Nedostaci:

- Python je spor! (izvođenje ciljnog programskog koda)
- Teže održavanje velikih projekata

Jednostavnost
korištenja



Brzina

Dinamički tipovi podataka

- Nije potrebno zadati tip podatka

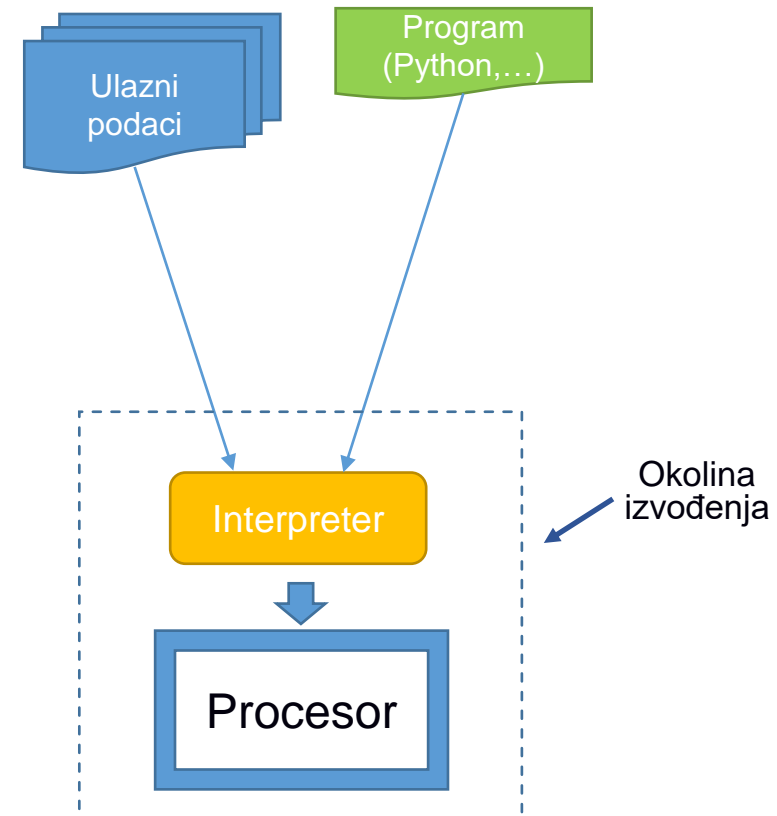
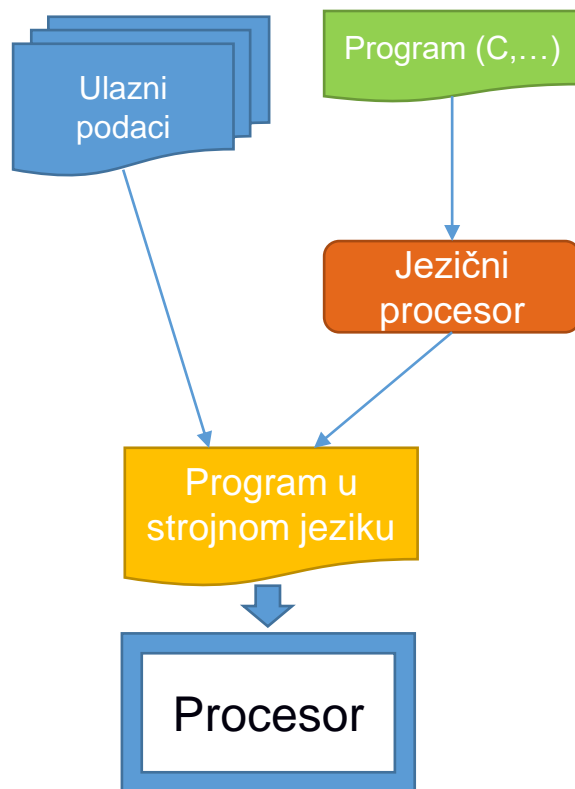
```
a = 3
b = [1, 2, 3, 5]
c = 'Niz znakova'
a = c
```

- Nedostatak:
 - Kod programskih jezika sa statički zadanim tipovima podataka jezični procesor (kompajler) odredi tipove prilikom prevođenja – nije ih potrebno naknadno provjeravati.
 - Programski jezici sa dinamički zadanim tipovima podataka provjere moraju obaviti za vrijeme izvođenja – obično su sporiji.

Interpreteri

- Poseban tip jezičnog procesora
 - Prevode naredbu po naredbu
 - Jezični procesori opće namjene prevode cijeli izvorni kod u strojni kod (potrebno je na ulazu postaviti kompletan kod koji sačinjava programsko ostvarenje)
 - Python je interpretirani jezik
 - Mogućnost interaktivnog izvođenja programa
 - Unos programa naredbu po naredbu
 - Praćenje izlaza nakon svake naredbe

Interpreteri



Interpreteri

- Nedostatak:

- Rezultantni strojni kod je sporiji jer prilikom prevođenja nije moguće provesti neke od postupaka optimizacije
 - Manji broj strojnih naredbi
 - Učinkovitije korištenje naredbenog cjevovoda
 - Učinkovitije korištenje računalne arhitekture (postavljanje podataka u registre)

a = 3

~~a = 3~~

c = 3 + 1 → c = 4

Skriptni jezik

- Viša razina od uobičajenih programskih jezika
- Komponiranje programa od komponenti
 - Programske knjižnice
 - Primjenski programi
 - Mogu biti napisani u drugom programskom jeziku!
- Primjer: NumPy
 - Rad s višedimenzionalnim poljima
 - Implementacija: C, Python

Radno okruženje

■ Postavljanje Python interpretera

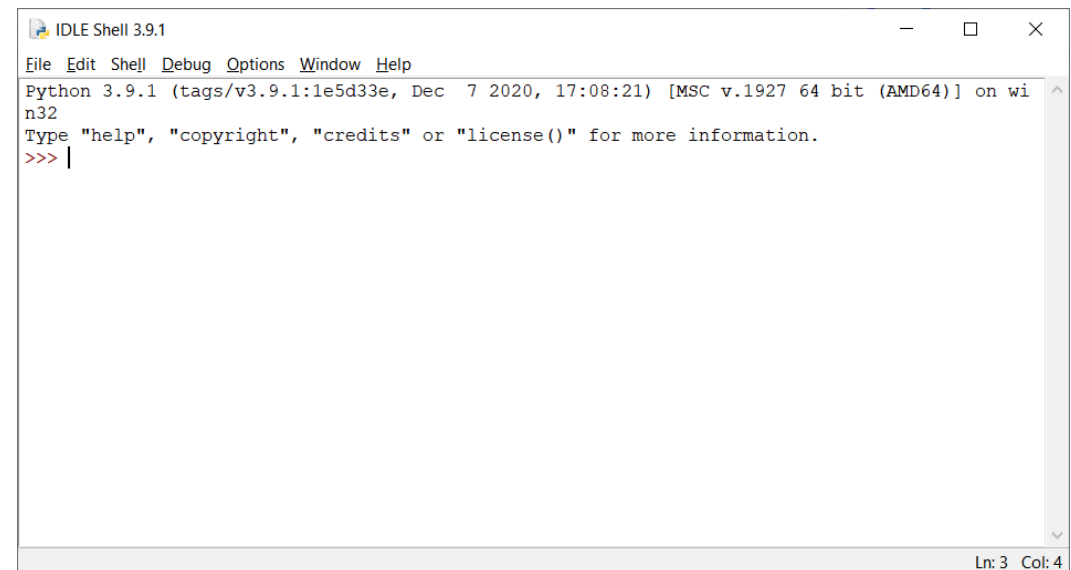
- <https://www.python.org/downloads/>
- Linux (Ubuntu): `sudo apt-get install python`

■ Pokretanje interpretera

- `python (.exe)`
- Zatvaranje interpretera: `quit()`

■ IDLE

- Integrirano razvojno okruženje



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Radno okruženje

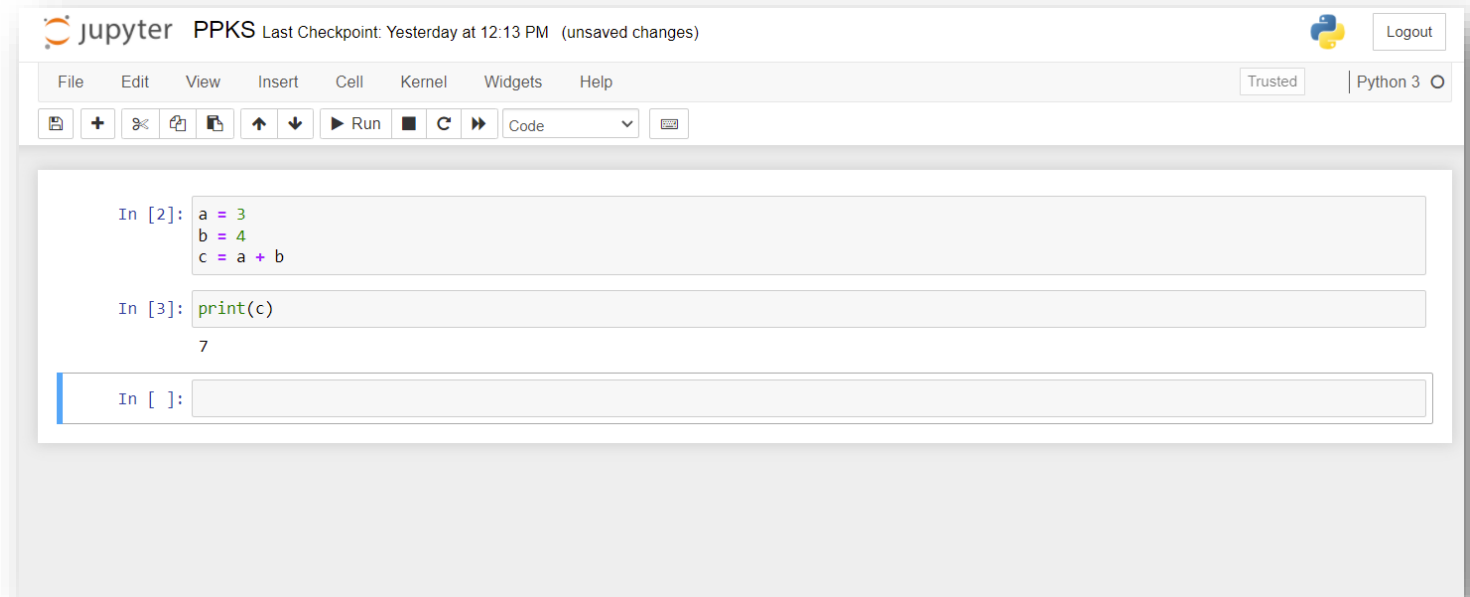
- Python programi (skripte)
 - Datoteke s ekstenzijom `.py`
- Pokretanje programa
 - `python imePrograma.py [arg1] [arg2] ...`
- Korišćenje prikladnog programa za uređivanje teksta
 - Sublime Text, Notepad++, ...
- Radna okruženja
 - PyCharm, Eclipse, ...

Radno okruženje

- Python programi (skripte)
 - Datoteke s ekstenzijom `.py`
- Pokretanje programa
 - `python imePrograma.py [arg1] [arg2] ...`
- Korišćenje prikladnog programa za uređivanje teksta
 - Sublime Text, Notepad++, ...
- Radna okruženja
 - PyCharm, Eclipse, ...

Radno okruženje

- Jupyter Notebook
 - Interaktivna bilježnica za pokretanje isječaka programskog koda
 - Integracija dokumenta i programskog koda – dobar alat za prikazivanje i dijeljenje rezultata
 - Web aplikacija
- Postavljanje
 - `pip install notebook`
- Pokretanje
 - `jupyter notebook`
- JupyterLab (alternativno)
 - `pip install jupyterlab`



Osnove jezika



Primjer programa

```
SUFFIXES = { 1000: ['KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'],
             1024: ['KiB', 'MiB', 'GiB', 'TiB', 'PiB', 'EiB', 'ZiB', 'YiB']}

def approximate_size(size, kb_is_1024_bytes = True):
    '''Convert a file size to human-readable form.
    Description...
    '''

    if size < 0:
        raise ValueError('number must be non-negative')

    multiple = 1024 if kb_is_1024_bytes else 1000

    for suffix in SUFFIXES[multiple]:
        size /= multiple
        if size < multiple:
            return '{0:.1f} {1}'.format(size, suffix)


    raise ValueError('number too large')

if __name__ == '__main__':
    print(approximate_size(1000000000000, False))
    print(approximate_size(1000000000000))
```

Sintaksa (1)

- Naglasak na čitljivosti koda, jednostavnosti pisanja programa
- Nije nužno koristiti oznaku kraja naredbe ";"
 - Naredba se proteže do kraja retka
 - Prelamanje naredbe kroz više redova se postiže ključne riječi "\"
- Ne ključne riječi za označavanje početka i kraja bloka naredbi (npr. "{" i "}")
 - Blokove određuje broj praznih znakova na početku naredbe
 - Naredba koja započinje blok završava znakom ":"
 - Tabulator (Tab) ili prazan znak
 - 4 prazna znaka (de facto standard)

Sintaksa (2)

Uvjet: 
____ **Naredba1**
____ **Naredba2**
____ Uvjet:
____ **Naredba3**
____ Naredba4



Ispravno

Uvjet:
____ **Naredba1**
____ **Naredba2**
____ Uvjet:
____ **Naredba3**
____ Naredba4



Neispravno

Sintaksa (3)

- Prazan blok naredbi
 - Ključna riječ **pass**

```
Uvjet:  
____Naredba1  
____Naredba2  
____Uvjet:  
____pass  
____Naredba4
```

Sintaksa (4)

- Deklaracija varijabli:

```
a = 1000  
b = 3 + 1  
c = 'Niz znakova'
```

- Programski tipovi se **ne** deklariraju
 - Varijabloma se pridružuje vrijednost iz koje se određuje tip podatka
 - Dohvaćanje prethodno nedefinirane varijable izaziva grešku

Sintaksa (5)

- Deklaracija varijabli (primjer):

```
SUFFIXES = { 1000: ['KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'],  
             1024: ['KiB', 'MiB', 'GiB', 'TiB', 'PiB', 'EiB', 'ZiB', 'YiB']}
```

- Osnovne ugrađene strukture podataka

- Riječnik, mapa, asocijativno polje

```
ime_mape = { ključ1 : vrijednost1, ključ2 : vrijednost2 }  
vrijednost = ime_mape[ključ]
```

- Liste

```
ime_liste = [vrijednost1, vrijednost2]  
vrijednost = ime_liste[cijeli_broj]
```

Sintaksa (6)

- Definiranje funkcije
 - Programski isječci koji o odrađuju određeni zadatak
 - Ulazni podaci (parametri), rezultati

```
def approximate_size(size, kb_is_1024_bytes = True):
```

- Ključna riječ **def**
- Naziv funkcije `approximate_size`
- Argumenti `size`, `kb_is_1024_bytes`
- Dinamički tipovi podataka
 - Nije zadan tip argumenata
 - Nije zadan tip povratne vrijednosti i vraća li funkcija povratnu vrijednost

Sintaksa (7)

- Vraćanje vrijednosti iz funkcije

```
return neka_vrijednost  
return '{0:.1f} {1}'.format(size, suffix)
```

- Ako funkcija ne vraća povratnu vrijednost rezultat je tipa `NoneType` (**None**)
 - Slično kao `Null` u drugim programskim jezicima
 - Tip podataka koji označava praznu vrijednost

- Pozivanje funkcije

```
povratna_vrijednost = ime_funkcije(argumenti)  
approximate_size(1000000000000)
```

Sintaksa (8)

- Opcionalni argumenti

pretpostavljena vrijednost

```
def approximate_size(size, kb_is_1024_bytes = True):
```

- Opcionalni argument: kb_is_1024_bytes

- Nije ga potrebno zadati u pozivu funkcije – poprima pretpostavljenu vrijednost
- Nije zadan tip povratne vrijednosti i vraća li funkcija povratnu vrijednost
- Navode se uvijek na kraju liste argumenata

```
print(approximate_size(10000000000000, False))  
print(approximate_size(10000000000000))
```

Sintaksa (9)

■ Komentari

- Unutar trostrukih navodnika `'''`
 - protežu se kroz više linija
- Nakon znaka
 - Protežu se do kraja linije `#`

```
'''Convert a file size to human-readable form.  
    Description...  
'''
```

```
# komentar programskog bloka
```

Sintaksa (10)

- Uvjetno ganjanje

```
if size < 0:  
    raise ValueError('number must be non-negative')
```

- Uvjetni operator

```
multiple = 1024 if kb_is_1024_bytes else 1000
```

Sintaksa (11)

- Uvjetno ganjanje - općenito

```
if uvjet1:  
    Naredba1  
    Naredba2  
    ...  
elif uvjet2:  
    Naredba  
    ...  
else:  
    Naredba  
    ...
```

```
rezultat = vrati_ako_tocno if uvjet else vrati_ako_netocno
```

Sintaksa (12)

▪ Iznimke

- Programski konstrukti u višim programskim jezicima koji se koriste za dojavljivanje greške
- Iznimku je moguće izazvati (stvoriti)
- Iznimku je moguće obraditi (`try-except-finally` skup naredbi)

```
raise Objekt_Tipa_Exception
```

```
raise ValueError('number must be non-negative')
```


Sintaksa (13)

- Programske petlje

```
for x in iteratorski_objekt:  
    # x je i-ti element u iteratorskom objektu - lista, mapa...
```

Naredba

Naredba

```
for suffix in SUFFIXES[multiple]:
```

```
SUFFIXES = { 1000: ['KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'],  
             1024: ['KiB', 'MiB', 'GiB', 'TiB', 'PiB', 'EiB', 'ZiB', 'YiB']}
```

Sintaksa (14)

- Programske petlje

```
while uvjet:
```

```
    # iteracija regulirana uvjetom
```

```
    Naredba
```

```
    Naredba
```


Sintaksa (15)

■ Aritmetičke i logičke operacije

- Zbrajanje +, oduzimanje -, množenje *, dijeljenje /, potenciranje **
- Operacije nad bitovima : &, |, ^
- Logičke operacije: and, or, not
- Operacije usporedbe: >, >=, <, <=, ==, !=, <>

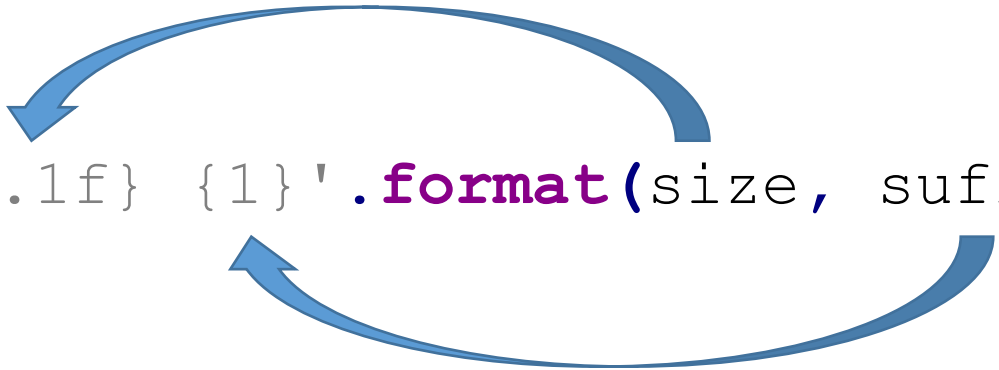
■ Konstante

- Znakovni niz: 'Primjer znakovnog niza'
- Cijeli broj: 10000000
- Brojevi s pomičnim zarezom: 425.46

size /= multiple  size = size / multiple

Sintaksa (16)

- Formatiranje znakovnog niza



The diagram illustrates the mapping between placeholders in a format string and arguments in a Python `.format()` call. The format string is `'{0:.1f} {1}'` and the arguments are `size` and `suffix`. A blue arrow points from the `{0:.1f}` placeholder to the `size` argument, and another blue arrow points from the `{1}` placeholder to the `suffix` argument.

```
'{0:.1f} {1}'.format(size, suffix)
```

Sintaksa (17)

■ Glavni program

- Svodi se na ispitivanje trenutnog programskog modula
- Ime programskog modula (knjižnice) koji se izvodi pohranjeno je u varijabli `__name__`
- Modul koji je interpreter pokrenuo izravno naziva se: `'__main__'`

```
if __name__ == '__main__':  
    print(approximate_size(10000000000000, False))  
    print(approximate_size(10000000000000))
```

Tipovi podataka



Tipovi podataka - Objekti

- Svaka vrijednost ima tip podatka
 - Varijabloma se određuje tip na temelju pridružene vrijednosti
- Svi tipovi podataka u Pythonu su objekti
 - Objektno orijentirani programski jezik
- Objekti
 - Programski konstrukti koji enkapsuliraju određenu funkcionalnost
 - Svaki objekt je građen od članskih atributa i funkcija (metoda)
 - Korištenje slično struct u programskom jeziku C
 - Operator .
 - `ime_objekta.clanska_var`
 - `ime_objekta.metoda`

```
'{0:.1f} {1}'.format(size, suffix)
```

Tipovi podataka - Objekti

- Razredi
 - Nacrti na temelju kojih se grade objekti
 - Pomoću razreda se programski definira nacrt objekta
 - Moguće je u memoriji stvoriti više objekata istog razreda (više varijabli istog tipa postoji u memoriji)
 - Stvaranje objekta u memoriji na temelju razreda naziva se instanciranje objekta
 - Tip podatka = ime razreda
- Razredi za primitivne tipove podatka (brojeve, nizove znakova)
- Razredi za složene tipove podataka (liste, mape...)
- Sve je objekt!
 - Funkcije su objekti, i sami razredi su objekti

Određivanje tipa podatka

- Tip podatka je ime razreda iz kojeg je objekt instanciran
- Funkcija `type()` vraća tip objekta
- Funkcija `isinstance()` provjerava je li objekt nekog tipa

```
type(123)
```

```
isinstance(123, int)
```

Ugrađeni tipovi podataka

- Logički (boolean) – `True` ili `False`
- Brojevi
 - Cijeli brojevi: niz znamenaka `1234`
 - Brojevi s pomičnom točkom `1.124`
 - Razlomci
 - Kompleksni brojevi
- Znakovni nizovi
 - Korištenje jednostrukih ili dvostrukih navodnika
 - Svi nizovi u Pythonu3 su u formatu Unicode
- Liste
- N-torke
- Skupovi
- Mape

Logički tip podataka

- Tip `bool`
- Vrijednosti: `True` ili `False`
- Rezultat su provođenja logičkih operacija

`1 > 3`

`5 == 4`

`3 != 2`

- Automatska pretvorba u boolean
 - U `False` se pretvaraju: `0`, `' '`, `None`, prazni tipovi (`[]`, `{}`, ...)
 - U `True` sve ostalo

`bool(0)`

Brojevi

- Cijeli brojevi **int**
 - Proizvoljna veličina, ovisi o veličini radne memorije – bignum (Python 3)
 - Znamenke se pohranjuju zasebno
- Brojevi s pomičnom točkom **float**
 - Preciznost 16 znamenki (double)
- Razlika u definiranju konstante
 - Brojevi s pomičnim zarezom imaju decimalnu točku

Brojevi

- Automatska pretvorba

- Zbrajanje, oduzimanje, množenje, potenciranje argumenata od kojih je jedan **float**, uvijek rezultira tipom **float**.
- Obavlja se pretvorba **int** u **float** i onda se izvršava operacija
- Dijeljenje dva tipa **int** uvijek rezultira s tipom **float** (Python3)
 - Python 2
 - $1 / 3 = 0$
 - $1 / 2. = 0.5$

Brojevi

- Cjelobrojno dijeljenje (Python3)
 - Operator `//`
 - Zaokružuje na manji broj (`3 // 2 = 1`)
 - Ako je jedan operand tipa `float`, rezultat će biti tipa `float`, ali svejedno će se odraditi zaokruživanje na manje
 - Operator ostatka cjelobrojnog dijeljenja `%` (`2 % 3 = 2`)
- Potenciranje
 - Operator `**`
 - `2 ** 12` – 2 na 12. potenciju

Brojevi

- Razlomci
 - Potrebno koristiti biblioteku **fractions**
 - Moguće je provoditi standardne operacije
 - Automatsko skraćivanje

```
import fractions
a = fractions.Fraction(2, 4)
b = fractions.Fraction(8, 12)
a + b
```

Brojevi

- Kompleksni brojevi
 - Programska knjižnica standardno uključena
 - Zadaje se realni i imaginarni dio
 - Moguće je provoditi standardne operacije

```
c = complex(1, 3)
d = complex(5, 8)
c * d
```