Praktikum softverski alati 1

PSA1 - Python

Prof. dr Mirjana Simić-Pejović

Katedra za Telekomunikacije

Kabinet: 108

3218-400

mira@etf.rs

Sadržaj

- Opšte informacije o Python programskom jeziku
- Python okruženja
- Tipovi podataka
- Operacije nad podacima
- Python I/O
- Strukture i petlje
- Funkcije
- Datoteke
- Moduli (biblioteke)

Opšte informacije o Python programskom jeziku

Python, opšte karakteristike

Programski jezik

Wikipedia:

- *"Python is a **general-purpose**, **high-level programming language** whose design philosophy emphasizes **code readability**. Python claims to "[combine] **remarkable power** with very **clear syntax**", and its standard library is large and comprehensive. Its use of **indentation for block delimiters** is unique among popular programming languages*."

- *"The reference implementation of Python (CPython) is **free and open source software** and has a community-based development model, as do all or nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation*."

Python, opšte karakteristike

- Guido van Rossum, December 1989. god.
- Prva vezija objavljena 1991. god.
- Naziv je dobio po engleskoj komičarskoj trupi Monty Python
- Bez obzira na poreklo naziva, zvanična maskota ovog programskog jezika postala je zmija piton.
- Masovno se uči kao prvi programski jezik: MIT, CU Boulder, ... (od skoro i na ETF)

Python, opšte karakteristike

- Interpreter (interpretator): direktno izvršava instrukcije (prevođenje programa i izvršavanje programa vremenski spojeni procesi), bez prethodne kompilacije na mašinski jezik i linkovanja (kod kompajliranja su prevođenje programa i izvršavanje programa vremenski odvojeni procesi)

- Vrlo brze probe, po tome nalik na BASIC, Octave, Matlab

- Ipak, kako se postupak prevođenja i izvršavanja izvodi pri svakom startovanju programa, Python je sporiji od programa koji koriste kompajler, kao npr. C programskog jezika (ali se dobro povezuje sa C-om)

- Radi pod raznovrsnim platformama

- Vrlo objektno orijentisan, mada ne mora da se koristi

Python, opšte karakteristike

- Jako moćne i raznovrsne biblioteke (pySerial, numpy, matplotlib, sympy, . . .)
- Vrlo moćni tipovi podataka
- Lako se prave novi tipovi podataka
- Jednostavna sintaksa
- Opšta namena
- Free!!!
- Jako dobro podržan, razvija se, rasprostranjen

Python, opšte karakteristike

detaljnije

Python se lako koristi

• Cilj svakog programskog jezika je da optimizuje (olakša) komunikaciju programera i računara – u tom cilju većina popularnih programskih jezika (Java, C++, C#, Visual Basic, ...) su jezici visokog nivoa (*high-level programming languages*), što znači da su bliži govornim jezicima nego mašinskom jeziku.

• Python je sa svojim jasnim i jednostavnim pravilima još bliži engleskom govornom jeziku. Programiranje u Python-u često nazivaju i "programiranje brzinom razmišljanja". Posledica je da su programi pisani na jeziku Python kraći i pišu se za manje vremena nego programi na mnogim drugim programskim jezicima. Sa druge strane, ima svu moć koja se i očekuje od savremenog programskog jezika.

Python, opšte karakteristike

detaljnije

Python kao objektno orijentisan programski jezik

- Objektno orijentisano programiranje (OOP) je takav pristup rešavanju problema pomoću računara, čija je suština intuitivna metoda predstavljanja informacija i akcija u programu. Čest je izbor u slučaju pisanja složenijih i obimnijih programa.
- Primeri jezika koji su objektno orijentisani su npr. Java, C#, Python,... U jezicima kao što su Java, C#, primena OOP nije pitanje izbora. Zbog toga su kratki programi nepotrebno složeni, a programerima početnicima potrebno prilično opsežna objašnjenja pre nego što uspeju nešto da realizuju.
- Python primenjuje drugačiji pristup: upotreba OOP u Python-u nije obavezna! To znači da vam OPP u Python-u stoji na raspolaganju, a da li ćete ga iskoristiti zavisi od problema koji rešavate (za manje i jednostavne probleme ne, za složenije da). Na taj način Python pruža i moć ali i fleksibilnost.

Python, opšte karakteristike

detaljnije

Python kao "*glue language*"

- Python se često opisuje i kao *glue* (lepak) programski jezik, čime se sugeriše na bitnu osobinu da se može integrisati sa drugim programskim jezicima, kao pto su npr. C, C++, Java...
- To znači da kada radi u Python-u, programer može iskoristiti rezultate nečeg već napravljenog u nekom drugom programskom jeziku.
- To dalje znači da se na ovaj način mogu iskoristiti prednosti koje pružaju drugi programski jezici (npr. veća brzina izvršavanja koju omogućavaju C ili C++), a da se pri tom zadrži lakoća pisanja programa koja je zaštitni znak programiranja u Python-u.

Python kao "*modul-based*"

- Osim jednostavnosti , jedna od ideja vodilja autora ovog programskog jezika je i da se izbegnu ponovna i nepotrebna računanja nekih poznatih i čestih programerskih problema. Rešenje primenjeno u Pythonu je primena gotovih modula (biblioteka) gde su rešenja ovih problema već realizovana (mnoge od njih ne dolaze uz osnovnu instalaciju već ih je potrebno naknadno dovući).

Python, opšte karakteristike

detaljnije

Python radi pod raznovrsnim platformama

- Programi napisani u Python-u ne zavise od platforme. Dakle, bez obzira na operativni sistem u kojem se realizuje program u Python-u, on će raditi i na bilo kojem drugom operativnom sistemu u kojem postoji Python (Linux, Windows, Mac, ...).

Python je besplatan i otvorenog koda

- Python je besplatan programski jezik! (besplatno se dovuče sa sajta i instalira).
- Štaviše, Python-ova licenca omogućava i da ga možete kopirati, deliti, menjati, ...
- Prihvatanje ideala otvorenog koda sastavni su deo onoga što Python čini tako popularnim i uspešnim.

Python 2 i 3

- Postoje dve verzije Python programskog jezika: Python 2.x i Python 3.x.
- Verzija 2 više „nije podržana“
- Verzija 3 nema *backward compatibility*
- Između verzija Pythona 2 i 3 nisu prevelike razlike (print, input, range,...)
- Ipak, problem može da se desi sa već napisanim programima
- Takođe, problem je ako se oslanjate na već postojeće programe
- Verzija 3 je aktuelna
- Izbor na kursu će biti verzija **Python 3**

Python okruženja

Python IDLE

Python IDE (*Integrated Development Environment***)**

- Danas postoji više integrisnih razvojnih okruženja za Python. U svojoj osnovnoj verziji, integrisano razvojno okruženje za Python je **IDLE** (termin je nastao u čast glumca Eric Idle, člana Monty Python grupe).

Python GNU/Linux:

- Python je već instaliran, tačnije, Python interpreter
- Python IDLE se mora dovući (ne dobija se uz Python)
- Biblioteke (moduli) se moraju dovući

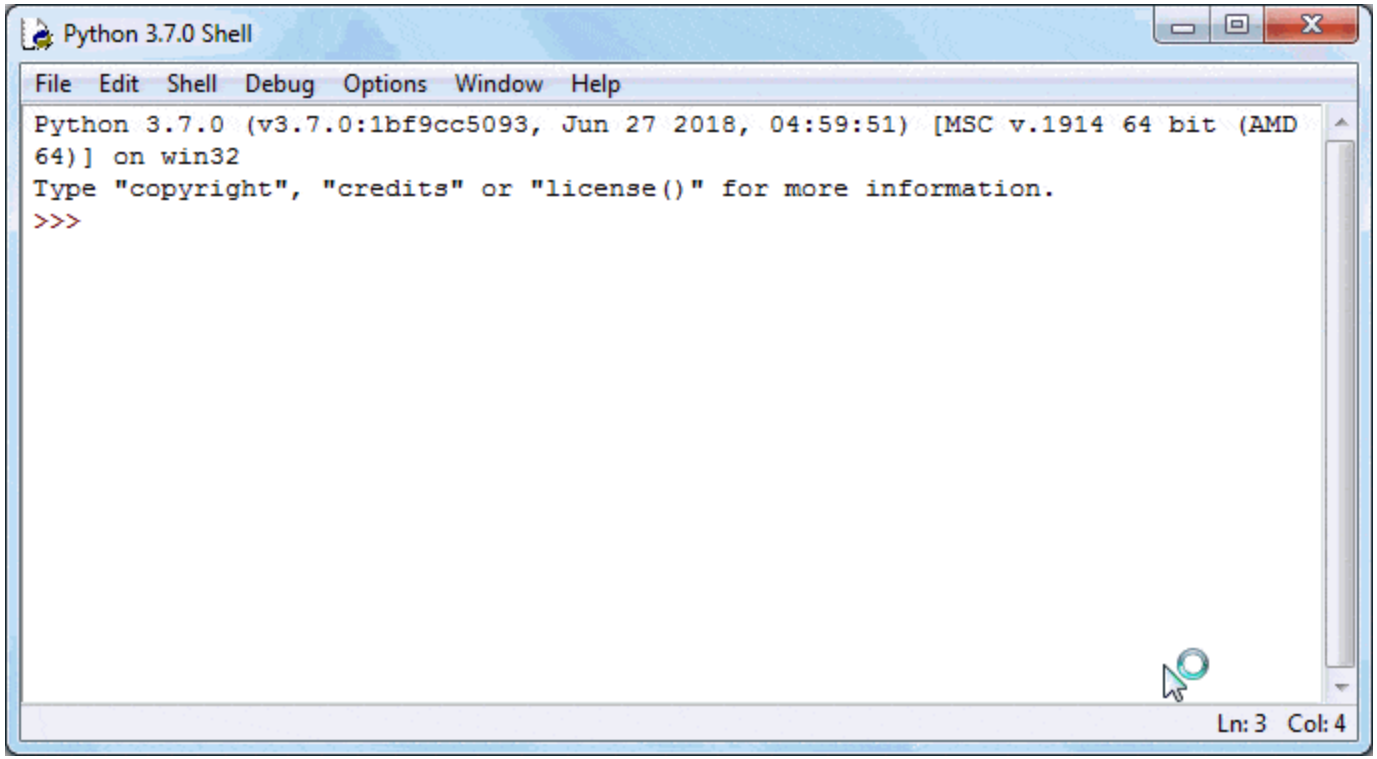
Python Windows:

- Python je potrebno dovući i instalirati (http://python.org/)
- Python IDLE u Windows je uključen u instalaciju (nije ga potrebno naknadno dovlačiti)
- Biblioteke (moduli) se moraju dovući

Python IDLE

Interaktivni režim (Python Shell):

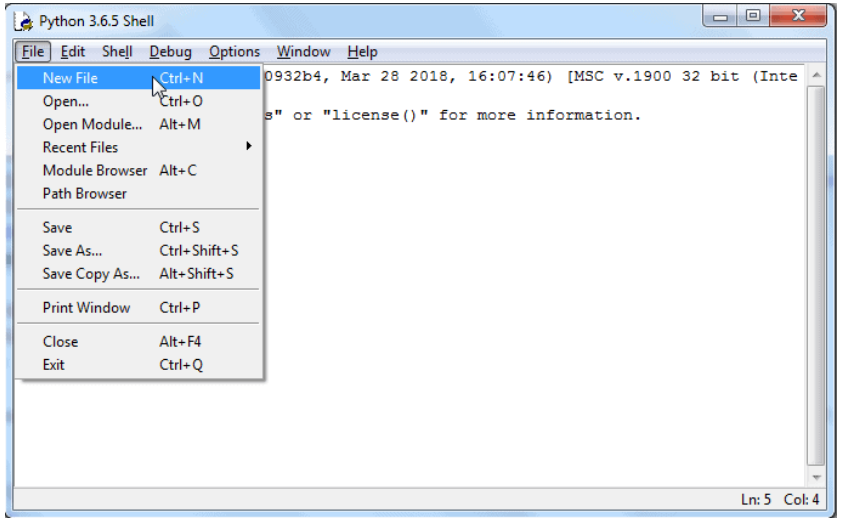
- izvršava se komanda po komanda: postavim pitanje – dobijem odgovor
- odziv je trenutan
- mogu se odmah videti rezultati

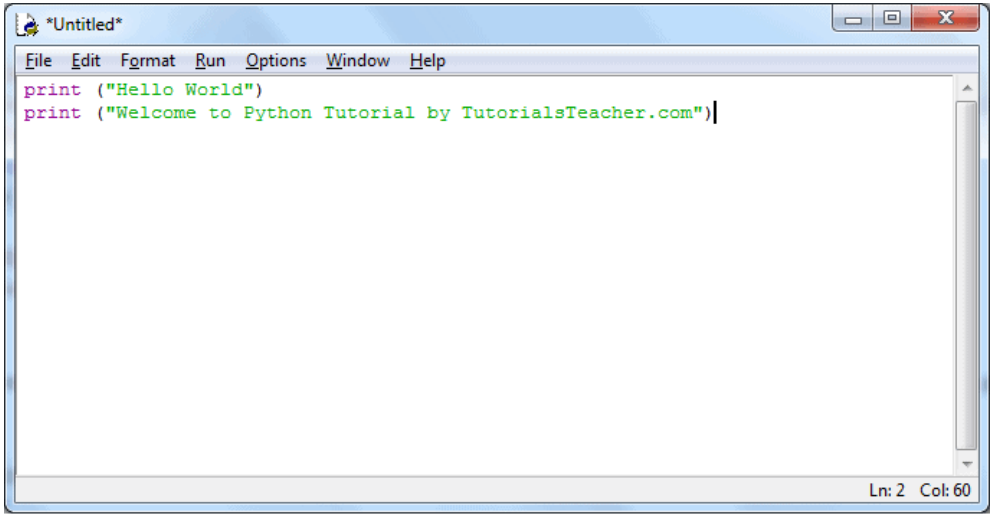


Python IDLE

Skript (*script*) režim

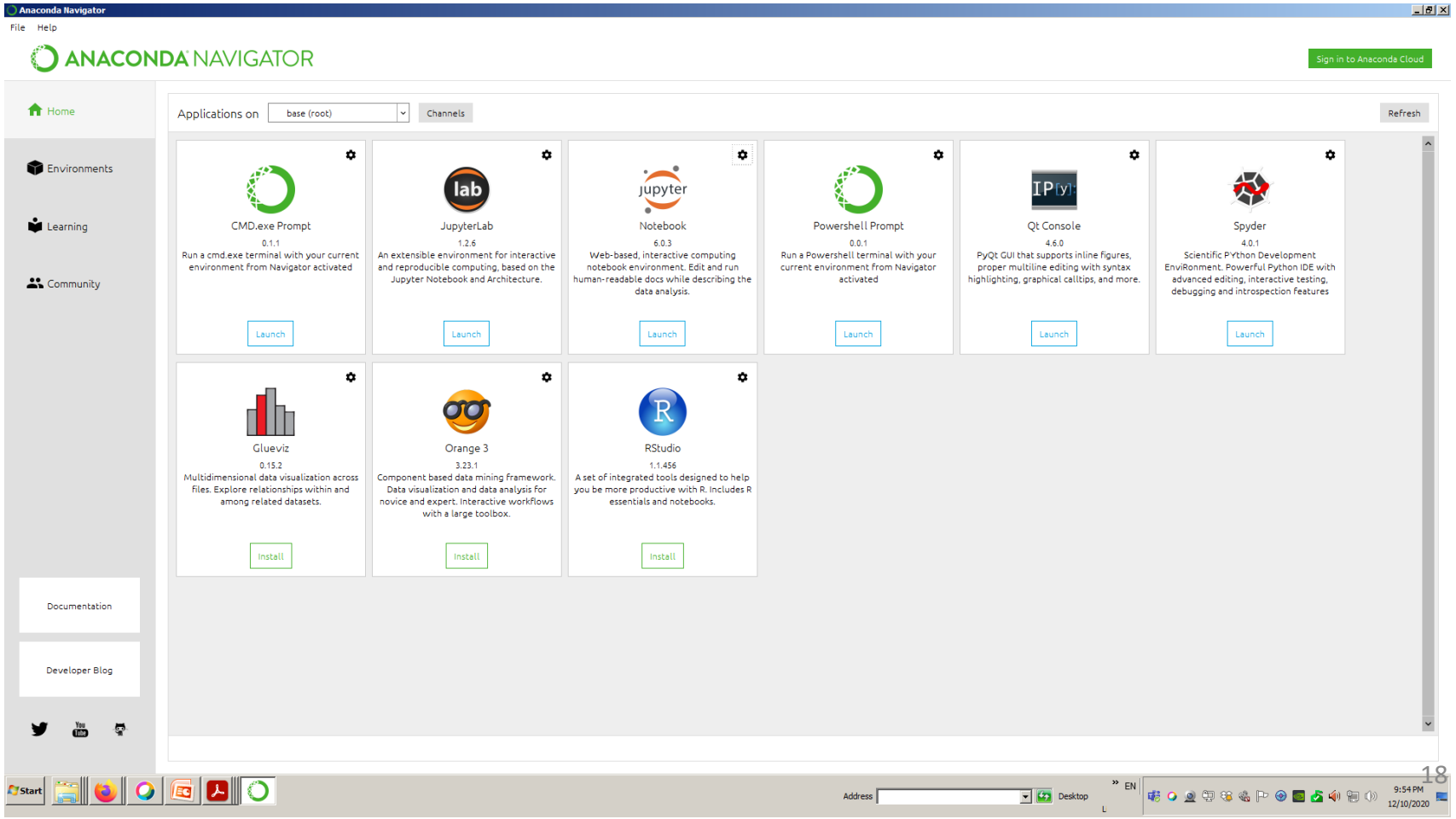
- Za pisanje programa koji će se čuvati, izvršavati kasnije, modifikovati,...





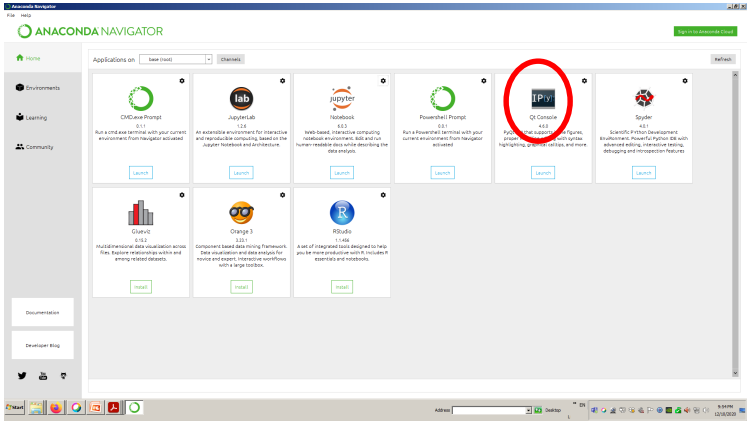
Anaconda

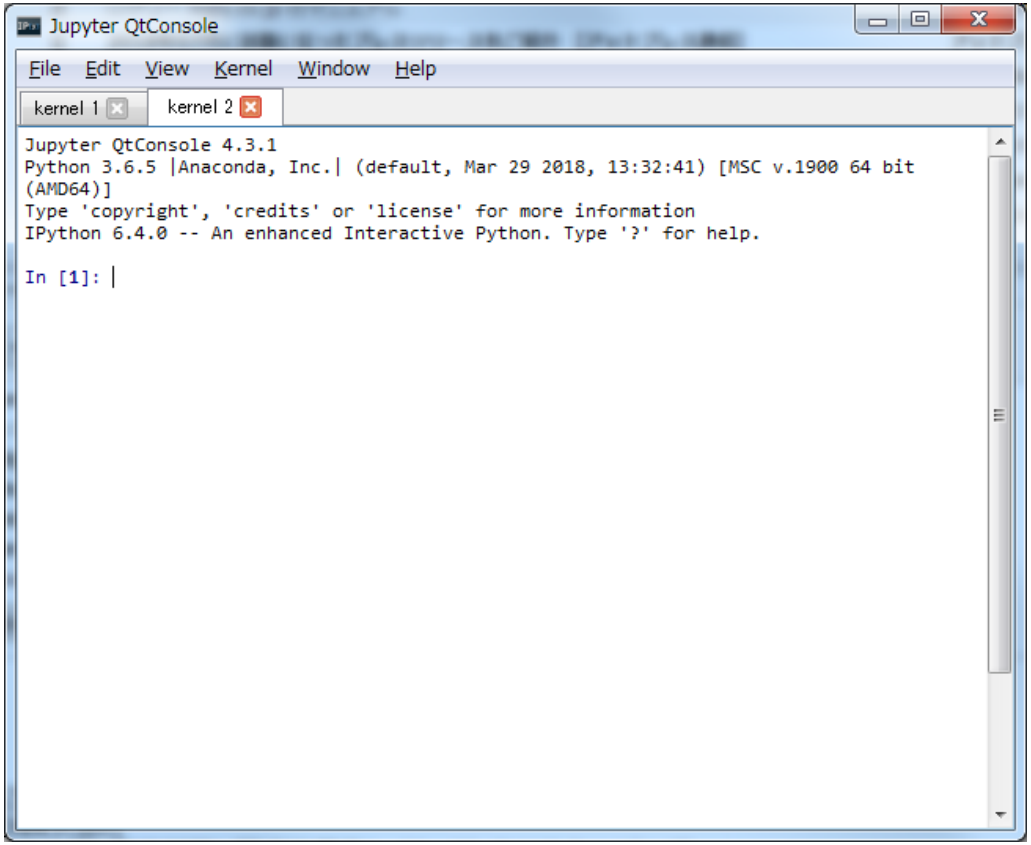
- **Anaconda Navigator** je desktop GUI koji dolazi sa distribucijom Anaconda Individual Edition
- Olakšava pokretanje aplikacija i upravljanje okruženjima bez korišćenja komandi preko komandne linije
- https://www.anaconda.com/products/individual



Python okruženja

- **IPython (**Interactive Python**)**

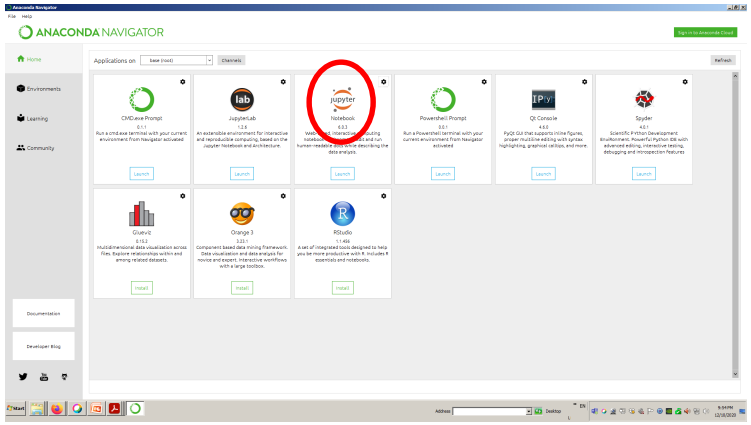


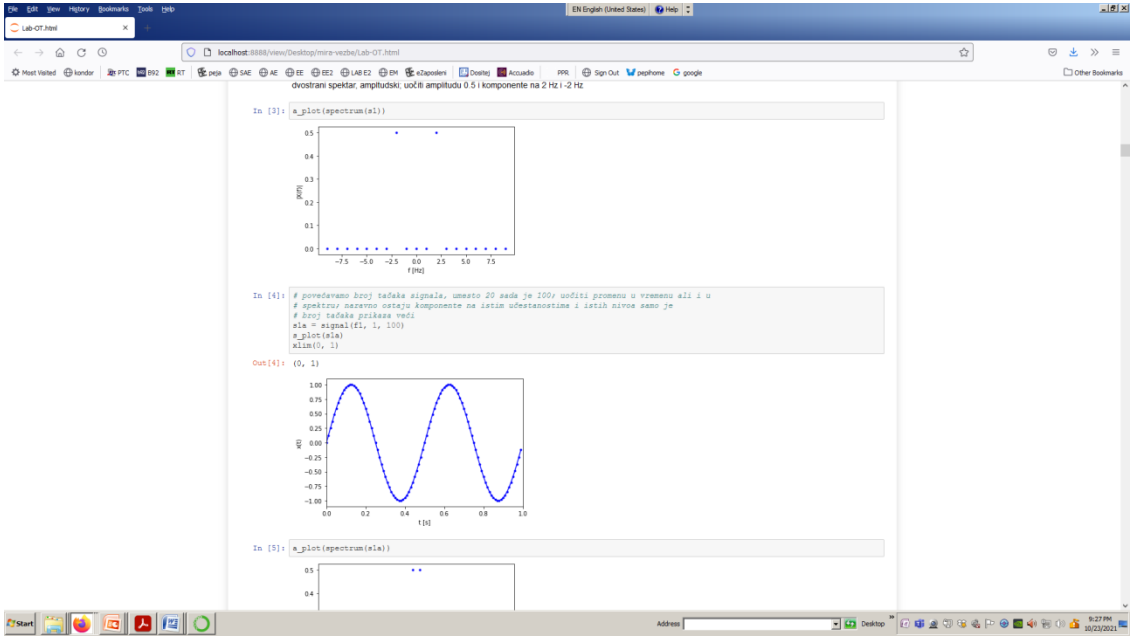


- Unapređeni interaktivni režim (Python Shell)
- Skripte se mogu pisati u bilo kojem editoru (npr. Notepad++)

Python okruženja

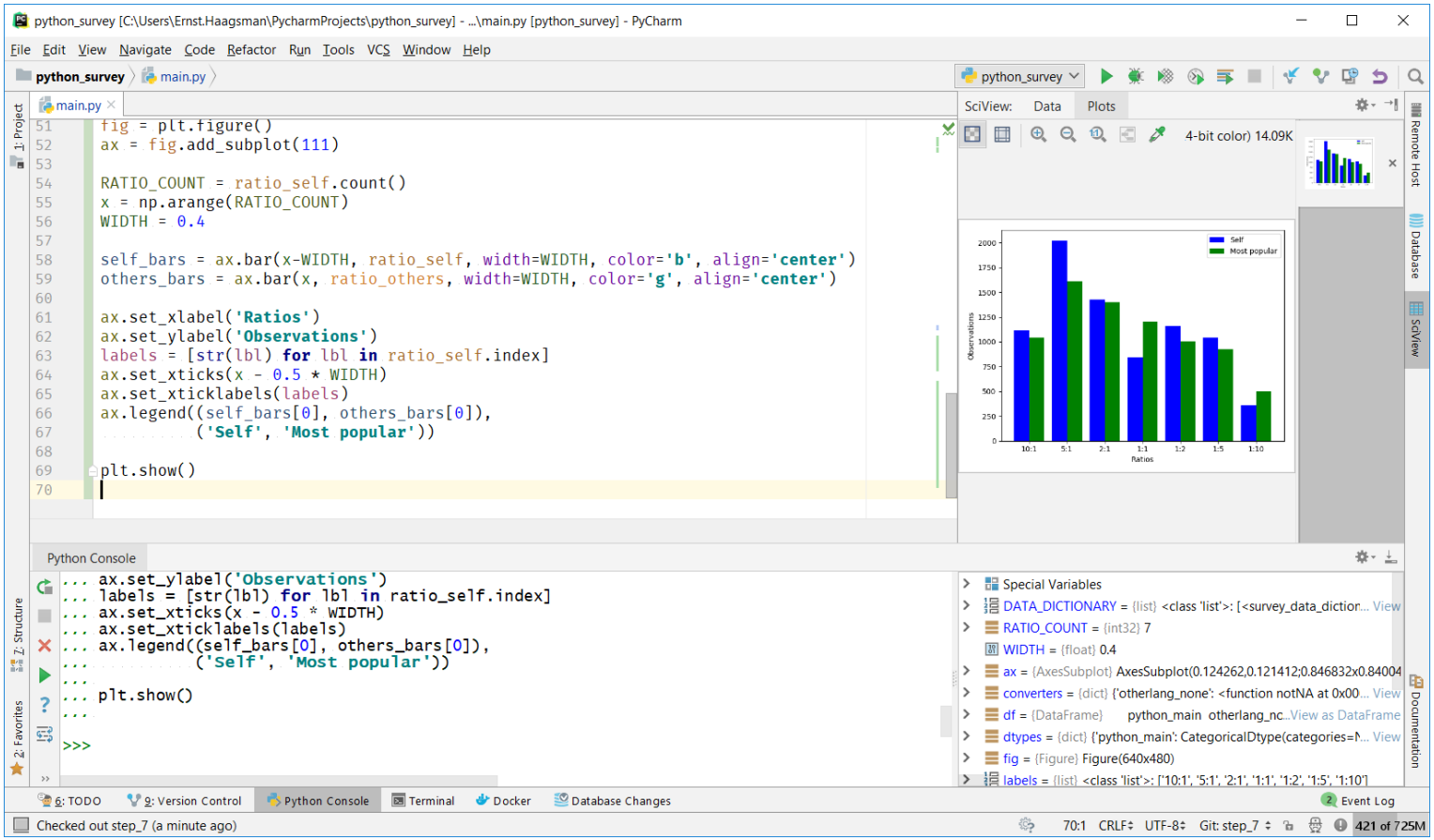
• **Jupyter :** web aplikacija koja omogućava kreiranje i distribuciju dokumenata koji sadrže delove koda, jednačine, vizuelizaciju, tekst (komentare)... Podržava preko 40 programskih jezika...





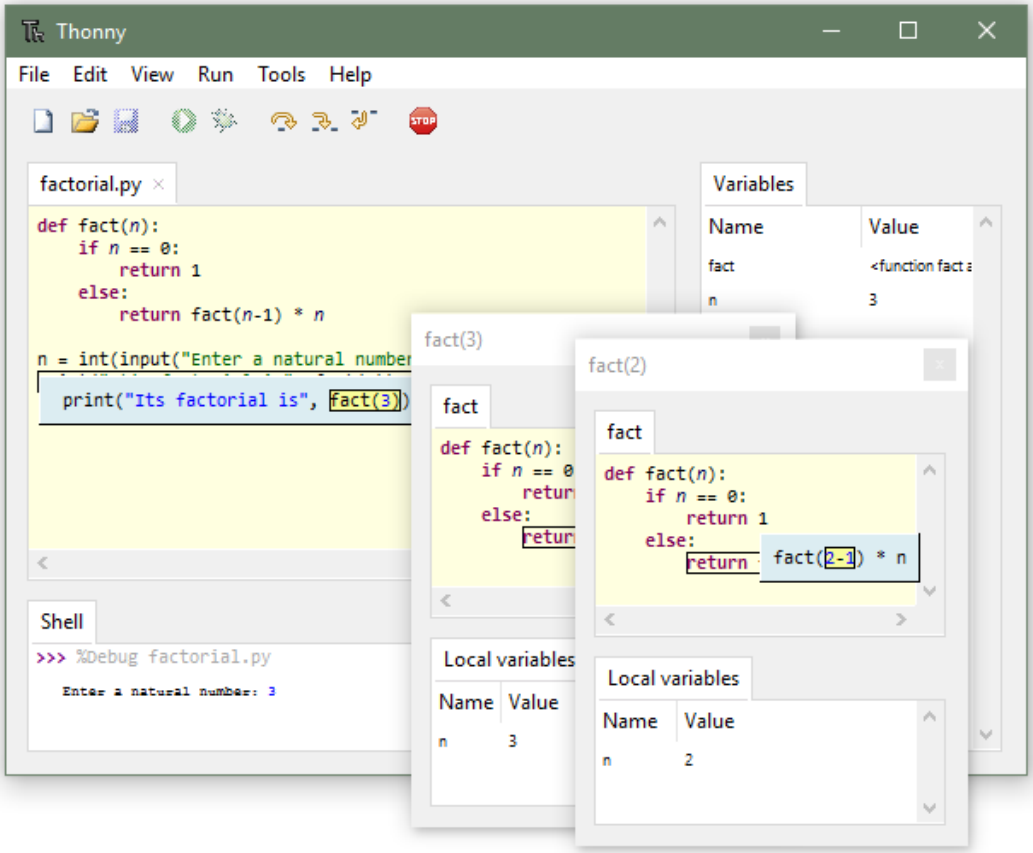
Python okruženja

- **PyCharm**
- Ima tri edicije, Professional, Community, Edu, od kojih su poslednje 2 free
- Može se integrisati u Anacondu



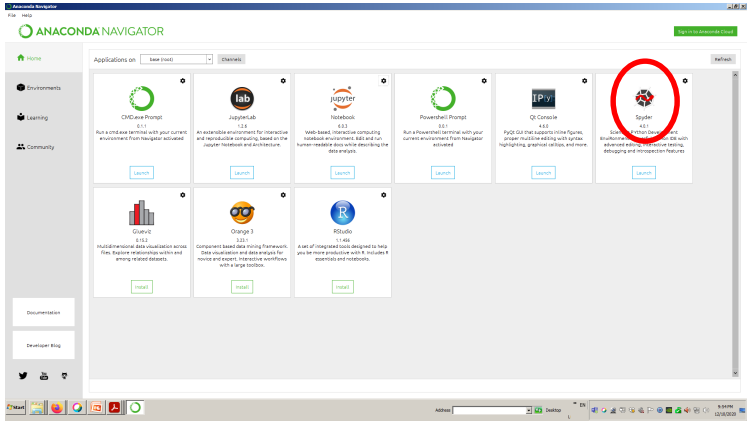
Python okruženja

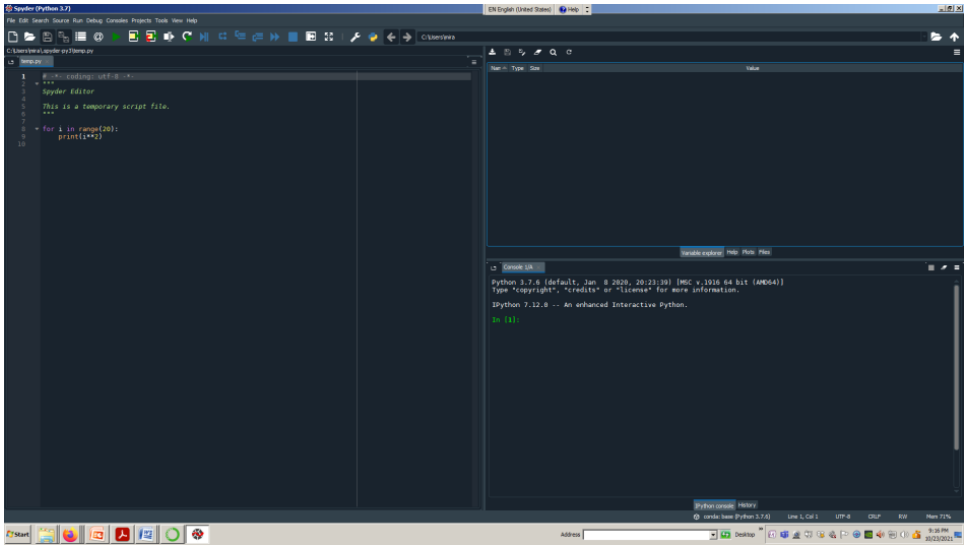
- **Thonny**
- Python IDE za početnike, free
- Jednostavno ispravljanje grešaka, označavanje sintaksnih grešaka, pomoć pri pisanju koda, evaluacija izraza korak-po-korak, ...

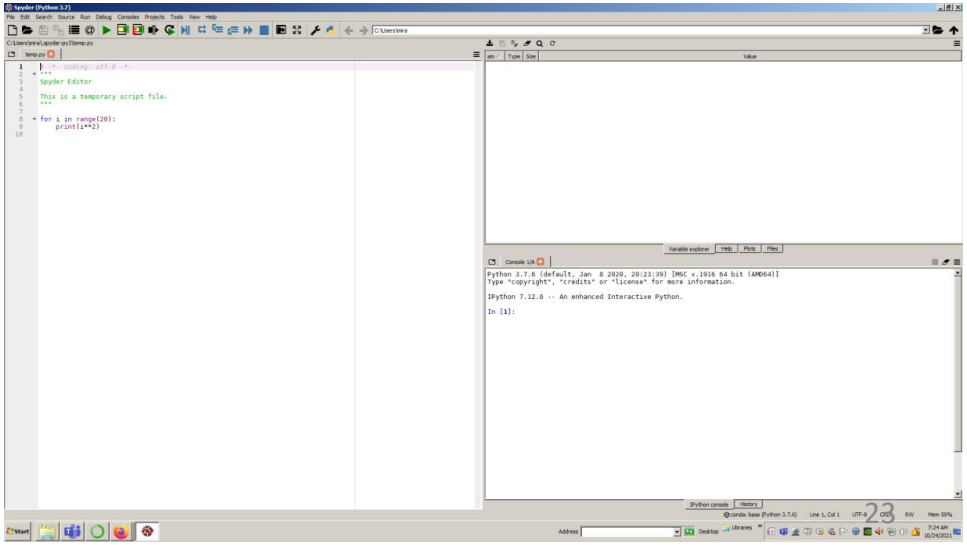


Python okruženja

- **Spyder**



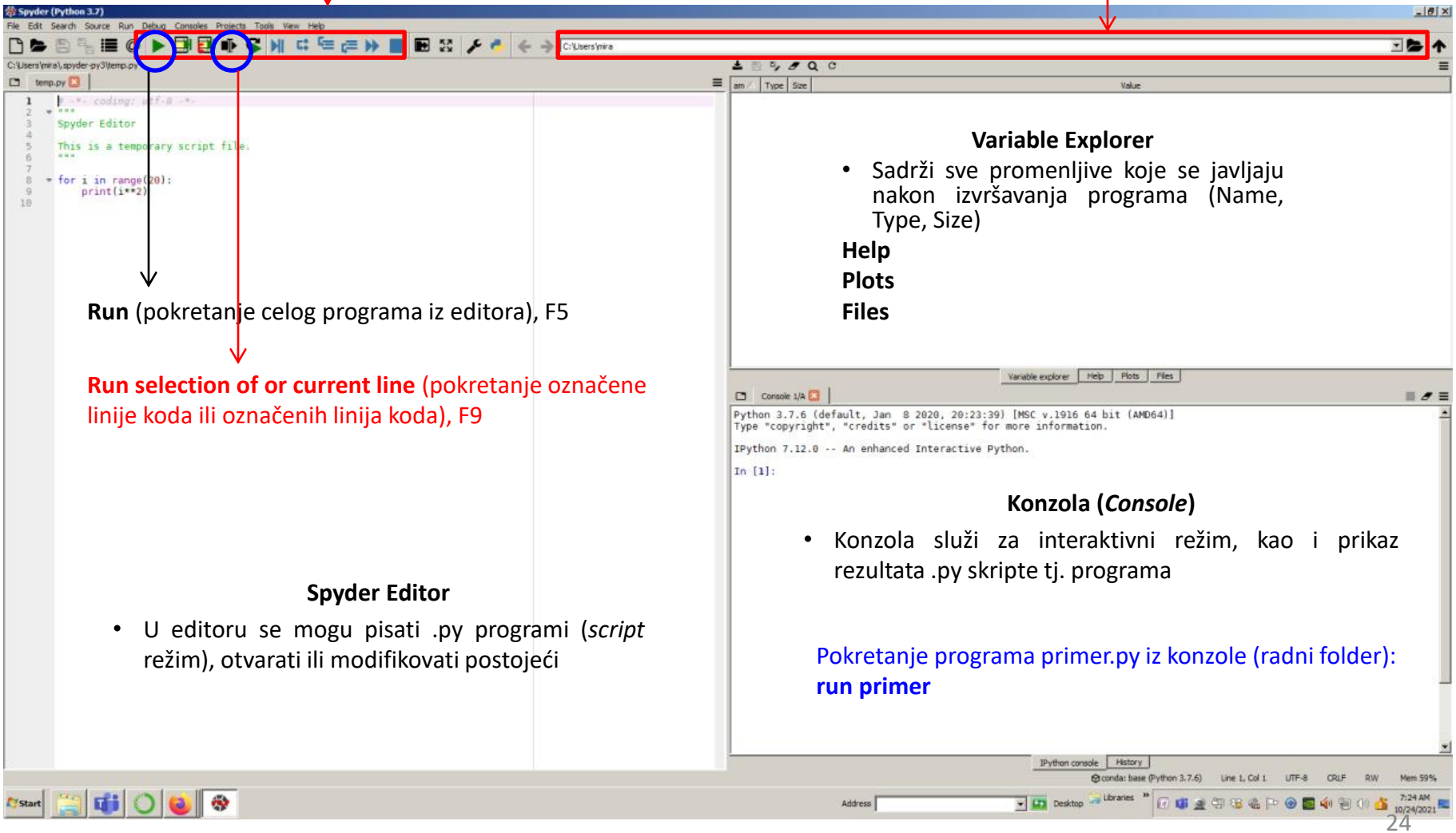




Spyder

Komande za pokretanje programa, delova programa (cells), sukcesivno pokretanje cells, pokretanje linija, debagovanje, ...

Radni (trenutni) folder. Ukoliko želimo da pozivamo neke spoljašnje datoteke u kodu, potrebno je da se putanja ka toj datoteci slaže sa putanjom ka trenutnom folderu. U suprotom će program prijaviti grešku kako ne može da nađe zadatu datoteku.



Run (pokretanje celog programa iz editora), F5

Run selection of or current line (pokretanje označene linije koda ili označenih linija koda), F9

Spyder Editor

- U editoru se mogu pisati .py programi (*script* režim), otvarati ili modifikovati postojeći

Variable Explorer

- Sadrži sve promenljive koje se javljaju nakon izvršavanja programa (Name, Type, Size)

Help

Plots

Files

Konzola (*Console*)

- Konzola služi za interaktivni režim, kao i prikaz rezultata .py skripte tj. programa

Pokretanje programa primer.py iz konzole (radni folder):
run primer

Tipovi podataka

Operacije nad podacima

Python I/O

Strukture i petlje

Funkcije

Datoteke

Moduli (biblioteke)

Python - Tipovi podataka

- **Logičke promenljive, Bulove** (*booleans*): dve vrednosti koje se predstavljaju kao False i True (s velikim početnim slovom), bool

- **Brojevi:**
  - Celi brojevi (integers), int
  - Realni brojevi (floating point numbers): svi brojevi koji imaju u sebi decimalnu tačku se smatraju realnim brojevima, float
  - Kompleksni brojevi (complex), complex

- **Sekvence (*sequences*)**: konačni uređeni skupovi čiji su elementi indeksirani. Ovde spadaju:
  - Liste , list
  - Tuples, tuple
  - Dictionary (asocijativni nizovi), dict
  - Sets, set
  - Stringovi, str

Python - operatori

- **Logičke operacije -** su definisane na operandima koji su bool tipa (*boolean* - mogu da imaju vrednost True ili False, 1 ili 0). Rezultat izvršavanja je takođe *boolean*.

Vrsta operacije	**Simbol**
Logičko i	and
Logičko ili	or
Logičko ne	not

Python - operatori

- **Operatori nad brojevima** – računske operacije, Python kao kalkulator

Vrsta operacije	Simbol
Sabiranje	+
Oduzimanje	-
Množenje	*
Deljenje	/
Celobrojno deljenje	//
Ostatak pri celobrojnom deljenju	%
Stepenovanje	**

- Python 2/Python 3 razlika:
  - U Pythonu 2 ukoliko su oba operanda celi brojevi prilikom deljenja dolazi do celobrojnog deljenja (operacija sa integerima se zatvara u integeru).
  - U Pythonu 3 za celobrojno deljenje postoji poseban operator (//).

- Ostale matematičke funkcije (sin, cos, sqrt…) ne pripadaju osnovnom paketu, već je potrebno učitati dodatnu biblioteku (modul).

Python - operatori

- **Operatori nad brojevima** – operatori poređenja.
- Izlaz je tipa bool, tj. True ili False

Vrsta operacije	**Simbol**	Py2
Jednako	= =	
Nije jednako	!=	< >
Veće od	>	
Manje od	<	
Veće ili jednako	>=	
Manje ili jednako	<=	

Python - operacije

- ***Bitwise* operatori**: operacije nad brojevima samo se broj predstavlja kao skup bita

x=10: 00001010
y=4: 00000100

Vrsta operacije	**Simbol**	**Primer**
Bitwise i	&	0 0000 0000
Bitwise ili	|	14 0000 1110
Bitwise ekskluzivno ili	^	14 0000 1110
Bitwise complement x (0 u 1 i 1 u 0) (isto je što i -x-1)	~x	-11 1111 0101
Pomera sve bite u x u levo za y mesta (x * (2**y))	x << y	160 1010 0000
Pomera sve bite u x u desno za y mesta (x // (2**y))	x >> y	0 0000 0000

Negativni broj, -x, se piše koristeći binarni zapis broja (x-1), pri čemu se u njemu svaki bit komplementira (0 u 1 ili 1 u 0). Npr.
-1 se dobija kao komplement od (1-1), tj. complement(0) = “11111111”
-10 se dobija kao komplement od (10-1), tj. complement(9) = complement(00001001)=11110110

Python – kompleksni brijevi

- **Kompleksni brojevi** u Python-u se mogu definisati na dva načina:
  1. Pomoću funkcije complex(realni_deo, imaginarni_deo) → a = complex(3, 4)
  2. Direktno pomoću imaginarne jedinice j → b = 3 + 4j

Parametri i metode nad kompleksnim brojevima:

	Vrsta operacije	Simbol
Parametri iz objekta	Realni deo	z.real
	Imaginarni deo	z.imag
Metode nad objektom (OOP)	Moduo	abs(z)
	Konjugovano-kompleksni broj	z.conjugate()

U primeru je pretpostavljeno da je kompleksni broj smesten u promenljivu z.
Modul: math

Python – operatori dodele

- Uobičajni zapis:
  - a = 10

- Sažeti zapis, u slučajevima kada se rezultat operacije dodeljuje jednom od operanada:
  - a+=2 (a = a + 2)
  - a*=2 (a = a * 2)
  - a/=2 (a = a / 2)
  - a-=2 (a = a - 2)
  - a**=2 (a = a ** 2)
  - …

Python zapisi brojeva

- Forma zapisa broja u nekom brojnom sistemu (**iz drugog u dekadni**):

  Vodeća nula, pa oznaka brojnog sistema, pa broj u tom brojnom sistemu

- Ulaz je broj u nekom brojnom sistemu, Izlaz je broj u dekadnom sistemu;
- Oznaka brojnog sistema i primeri:
  - **o: oktalni :** 0o12 10
  - **O: oktalni:** 0O12 10
  - **b: binarni:** 0b11 3
  - **B: binarni :** 0B11 3
  - **x: heksadecimalni :** 0x35 53
  - **X: heksadecimalni :** 0X35 53

- Obrnuto: Forma zapisa **iz dekadnog u neki drugi brojni sistem**:

  Skraćenica željenog brojnog sistema(dekadni broj koji se želi konvertovati)

- Ulaz je broj u dekadnom sistemu, Izlaz je broj u željenom brojnom sistemu.
- Skraćenica brojnog sistema i primeri:
  - **oct():** oct(10) 12
  - **bin():** bin(3) 11
  - **hex():** hex(53) 35

Python – Liste (Lists)

- **Liste** su tip podataka u Pythonu koje čine elementi uzajamno odvojeni zarezima, koji se nalaze unutar uglastih (srednjih) zagrada []. Najbliže su nizovima mada su specifične u odnosu na nizove u drugim programskim jezicima.

- Elementi liste mogu biti različitog tipa.

- Elementima liste moguće je pristupiti indeksiranjem (preko uglastih zagrada [])

- Indeksiranje u Python-u počinje od 0.

- Prvi element liste koja ima N elemenata ima indeks 0, drugi indeks 1, treći 2... Dok poslednji element ima indeks N-1.

- Za razliku od drugih programskih jezika, elementi liste se mogu indeksirati i unatrag, tako da poslednji ima indeks -1, pretposlednji -2... i prvi -N.

A = [2, 7, 12, 8, 26]

B = [True, 4, ‘PSA1’, 18.5]

Python – Liste

Indeksiranje elemenata liste:

Značenje	**Oznaka**
Prvi element liste A	A[0]
Drugi element liste A	A[1]
i-ti element liste A	A[i-1]
Poslednji element liste A	A[-1]
Pretposlednji element liste A	A[-2]
Elementi liste od indeksa *i* do indeksa *j-1*	A[i:j]
Elementi liste od prvog do trećeg	A[0:3]
Elementi liste od prvog do trećeg	A[:3]
Elementi liste od drugog do trećeg	A[1:3]
Elementi liste od trećeg do pretposlednjeg	A[2:-1]
Elementi liste od drugog do poslednjeg	A[1:]

Python – Liste

Operacije nad listama:

Opis operacije	**Operacija**	**Primer**
Dužina (broj elemenata) liste	len	len(A)
Brisanje liste ili elementa liste određenog indeksa	del	del A[indeks]
Brisanje određenog elementa liste	remove	A.remove(element)
Dodavanje jednog novog elementa na kraj liste	append	A.append(element)
Produžavanje liste novim elementom/elementima	extend	A.extend(element)
Dodamo element y na mestu x u listi	insert	A.insert(x, y)
Obrtanje redosleda elemenata liste	reverse	A.reverse()
Sortiranje elemenata liste	sort	A.sort()
Indeks prvog javljanja elementa u listi	index	A.index(element)
Broj ponavljanja elementa u listi	count	A.count(element)
Nalazi li se element u listi?	in	element in A

• Liste imaju tzv. **mutability** osobinu, što znači da joj se mogu menjati pojedini članovi.

Python – Tuple

- Po opisu slični listama: **tuple** je tip podataka u Pythonu koga čine elementi uzajamno odvojeni zarezima, koji se nalaze unutar oblih (malih) zagrada ().
- Elementi koji čine tuple mogu biti različitog tipa.
- Elementima koji čine tuple moguće je pristupiti indeksiranjem (preko uglastih zagrada [])
- Indeksiranje u Python-u počinje od 0, što važi i za tuple.
- U čemuje onda razlika u odnosu na liste?
- Tuple je **immutable**, što znači da mu se ne mogu menjati članovi. Dakle, tuple se posmatra kao celina.

A = (2, 7, 12, 8, 26) B = (True, 4, ‘PSA1’, 18.5)

Python – Nizovi i Matrice

- **Nizovi** se mogu shvatiti kao liste, dok **Matrice** u Pythonu nisu podržane na način kako smo to matematički navikli (ali zato postoji modul koji to podržava).

- Matrice se definišu kao liste listi.

Lista kao matrica → A = [[1, 2], [3, 4]]

B = [1, 2, 3, 4] ← Lista kao niz

- Na nama je da interpretiramo šta su vrste a šta kolone.

- Elementima matrice se pristupa indeksiranjem podliste gde je željeni element, a potom i indeksiranjem samog elementa u toj podlisti.

- Liste nisu optimalno rešenje za rad sa matricama, što se posebno odnosi na operacije nad matricama, obzirom da matrice sadrže objakte istog tipa a liste ne moraju.

Python – Stringovi

- **Stringovi** kao tip podataka predstavljaju niz simbola (karaktera) koji se predstavljaju između apostrofa (‘) ili navodnika (“), npr. ‘Pajton’. Modul za rad sa stringovima: string.
- Simboli unutar stringa se indeksiraju na isti način kao i elementi liste.

Operacije nad stringovima i operatori poređenja stringova:

Opis operacije	**Operacija**	**Primer**
Dužina (broj elemenata) stringa	len	len(s)
Konkatenacija	+	s + ‘abc’
Ponavljanje	*	s * 3
Indeksiranje	[ ]	s[5]
Izdvajanje opsega	s[:]	s[1 : 4]
Nalazi li se element u stringu?	in	‘a’ in s
Operator poređenja stringova a i b, jednako	= =	a == b
Operator poređenja stringova a i b, manje	<	a < b
Operator poređenja stringova a i b, veće	>	a > b

Python – Stringovi

Metode nad stringovima :

Opis operacije	**Primer**
String s sa svim velikim slovima	s.upper()
String s sa svim malim slovima	s.lower()
String s sa velikim početnim slovom (prvo slovo čitavog stringa)	s.capitalize()
String s sa velikim početnim slovima svakog podstringa	s.title()
Centriranje stringa s na polju koje ima x karaktera	s.center(x)
Da li se string s sastoji samo od slovnih karaktera?	s.isalpha()
Lista podstringova stringa s za koje je delimiter podstring ‘sep’ (npr. separator je zarez ‘,’ ili blanko ‘ ‘)	s.split(‘sep’)
Broj pojavljivanja podstringa ‘x’ u stringu s	s.count(‘x’)

Python – Dictionary

- **Dictionary** (rečnik) je poseban tip podataka u Pythonu (bulit-in) koji se sastoji iz zapisa unutar vitičastih zagrada { }.

- Zapisi su odvojeni zarezima i sastoje se od dva dela (razdvojena dvotačkom):
  - ključa (koji je tipa string): keys
  - vrednosti: values

- I ključevi i vrednosti su orderovane liste.

- Par 'key-value' napravljen je u cilju dodatne optimizacije ovog tipa podataka, imajući u vidu njegovu primenu (npr. telefonski imenik).

- Za razliku od sekvenci (liste, tuples) koje se indeksiraju brojevima, dictionary je tip podataka koji se indeksira pomoću ključne reči. Za to služi deo ključ tj. key. Zato ih često zovu asocijativni nizovi.

- Ključevi moraju biti jedinstveni unutar dictionary.

Dictionary → t = {'mira': 400, 'jelena': 348, 'milos': 361}

Python – Sets

- **Sets** je takođe poseban tip podataka u Pythonu, gde kao i liste, tuples, stringovi i dictionary pripada sekvencama.

- Set je neorderovan, neindeksiran skup različitih elemenata, koji se nalaze unutar vitičastih zagrada. Razdvojeni su zarezima.

- Za kreiranje seta koristi se funkcija set().

a = set('abracadabra‘)

a = {‘a’, ‘b’, ‘r’, ‘k’, ‘d’}

```
voce = {‘jabuka’, ‘banana’, ‘jagoda’, ‘banana’, ‘jabuka’}
print(voce)

{‘jabuka’, ‘banana’, ‘jagoda’}
```

Python – Sets

Operacije nad setovima:

Vrsta operacije	**Simbol**
Razlika dva skupa: iz skupa x oduzmemo elemente skupa y (elementi koji su u x ali nisu u y)	x - y
Razlika dva skupa: iz skupa y oduzmemo elemente skupa x (elementi koji su u y ali nisu u x)	y - x
Unija skupova x i y (elementi koji su bilo u x bilo u y, bilo u oba)	|
Presek dva skupa x i y (elementi koji su i u x i u y)	x & y
XOR (elementi koji su ili u x ili u y ali ne u oba)	x ^ y

Python – ulaz/izlaz (I/O)

- Za kontrolu ulaza/izlaza u Pythonu koriste se dve bulit-in funkcije:
  - **input()**
  - **print().**
- Ako vrednosti svih promenljivih nisu definisane u samom programskom kodu već je potrebno da neke unese sam korisnik, koristi se funkcija **input().**

input(‘string’**)**

- ‘String’ unutar input() je opcioni parametar i koristi se ako se želi i nešto ispisati na ekranu (osim očekivanog unosa nekih vrednosti od samog korisnika). Npr.

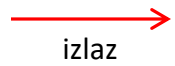
input(‘Unesite ceo broj: ’**)**

- Treba naglasiti da se svaki unos od strane korisnika u okviru funkcije input() u Pythonu tretira kao **string** (uključujući i brojeve). Dakle, ukoliko je vrednost koju je uneo korisnik potrebno tretirati npr. kao ceo broj, mora se izvršiti konverzija iz stringa u int.

Python – ulaz/izlaz (I/O)

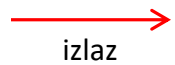
- Za prikaz rezultata na standardni izlaz (ekran), koristi se funkcija **print().**

```
print('Praktikum softverski alati 1')
```



Praktikum softverski alati 1

```
a = 3
print('Vrednost promenljive a iznosi:', a)
```



Vrednost promenljive a iznosi 3

- Ako želimo da naš izlaz izgleda atraktivnije, koristi se formatirani izlaz koji se realizuje koristeći **str.format()** metod:

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

```
x = 3
y  = 8
print('Vrednost promenljive x je {}, a promenljive y je {}' .format(x, y))
```

Vrednost promenljive x je 3, a promenljive y je 8

Python – ulaz/izlaz (I/O)

- Očigledno je da se vitičaste zagrade u formatiranom izlazu koriste za čuvanje mesta (*placeholder*).
- Formatirani izlaz se može koristiti i u slučaju više pojmova, pri čemu je u tom slučaju moguće kontrolisati i njihov redosled:

```
print('Na času su {0} i {1}'.format('Mika', 'Laza'))
```

↓

Na času su Mika i Laza

```
print('Na času su {1} i {0}'.format('Mika', 'Laza'))
```

↓

Na času su Laza i Mika

- Takođe, u formatiranom izlazu moguće je koristiti i ključne reči:

```
print('U toku je čas iz {predmet}'.format(predmet = 'PSA1'))
```

↓

U toku je čas iz PSA1

Python – Strukture i petlje

- U Python programskom jeziku, za kontrolu toka koriste se sledeće strukture i petlje:
  - **IF** struktura
  - **While** petlje
  - **For** petlje

- Za razliku od drugih programskih jezika koji programske blokove razdvajaju pomoću npr. vitičastih zagrada ili ključnih reči (begin, end, ...), Python koristi uvlačenje teksta.

- Ovaj način razdvajanja doprinosi većoj čitljivosti samog koda. Primenjuje se kod IF strukture, While petlje, For petlje, funkcija.

Python - IF struktura

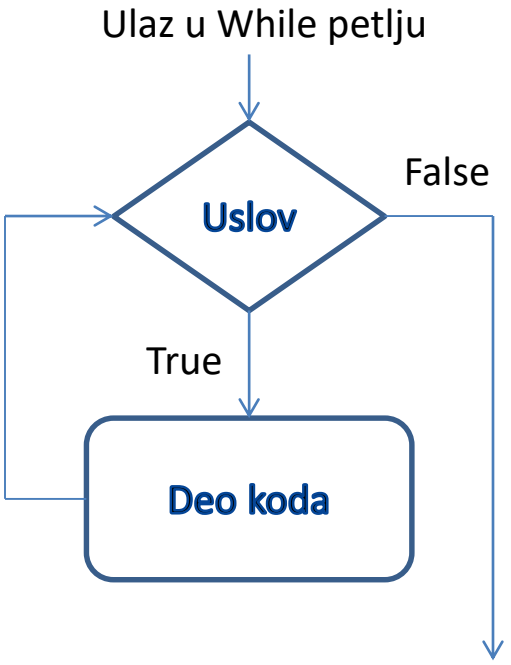
- Slično kao kod drugih programskih jezika, **IF struktura** se koristi kada od nekog uslova zavisi koji deo programskog koda će se izvršiti.
- Uslov je promenljiva bulovog tipa, *boolean*.

```
if    uslov 1:
      deo koda 1
elif  uslov 2:
      deo koda 2
elif  uslov 3:
      deo koda 3
.
.
.
else:
      deo koda n
```

- **elif** delova može biti nula ili više, dok je deo **else** opcioni.
- Ključna reč **elif** je skraćenica od **else if** i koristi se radi izbegavanja višestrukih uvlačenja teksta.

Python – While petlja

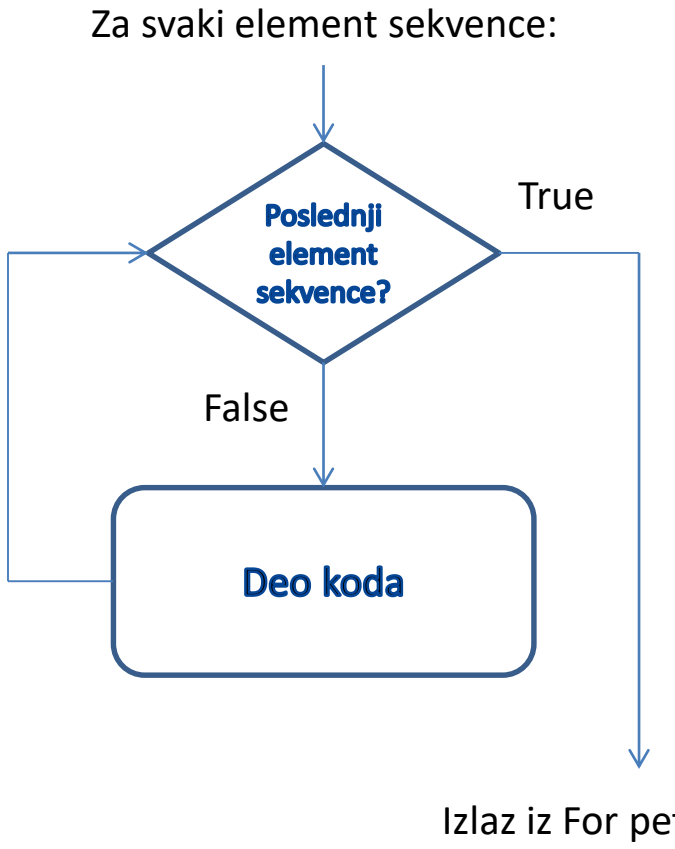
- Kao i kod drugih programskih jezila, **While** petlja se koristi za ponavljanje dela koda sve dok je ispunjen zadati uslov.
- Uslov je promenljiva bulovog tipa, *boolean*.
- Koristi se kada ne znamo unapred broj iteracija petlje (kategorija *indefinite iteration*).
- Donekle je slična IF strukturi jer se deo koda izvršava kad je ispunjen neki uslov (True). Ipak, za razliku od IF strukture, while petlja nastavlja sa izvršavanjem tog dela koda dokle god je taj uslov ispunjen (True).




```
while uslov:
    deo koda
```

Python – For petlja

- **For** petlja se koristi za ponavljanje delova koda kada je unapred poznat i eksplicitno definisan broj iteracija (kategorija *definite iteration*).
- For petlja u Pythonu se koristi za ponavljanje preko sekvence (lista, tuple, string, ...) ili bilo kog drugog iterabilnog objekta.
- Ovakav način ponavljanja prolazeći preko svakog elementa sekvence (iteriranje preko sekvence) zove se traversal.




```
for  n in sekvenca:
        deo koda
```

Python – For petlja

- Često se u slučaju For petlje za generisanje sekvence brojeva koristi funkcija **range().**

- **Range()** u izvesnom smislu spada u tzv. "lazy“ objekte jer ne generiše svaki broj koji sadrži kada ga kreiramo (ne čuva sve vrednosti u memoriji jer bilo bi neefikasno: pamti početak, zaustavljanje, veličinu koraka i generiše sledeći broj u pokretu).

range(start, stop, korak**)**

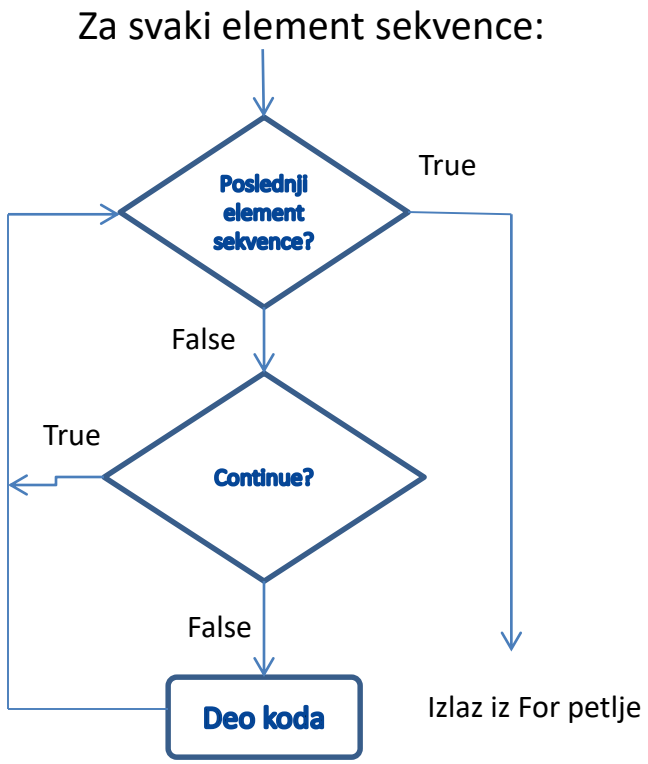
- Ako se ništa ne specificira, podrazumena se da je korak = 1. Sekvenca je od broja (start) do broja (stop-1). Ako postoji samo jedan broj, podrazumeva se da je to stop a da je start =0.

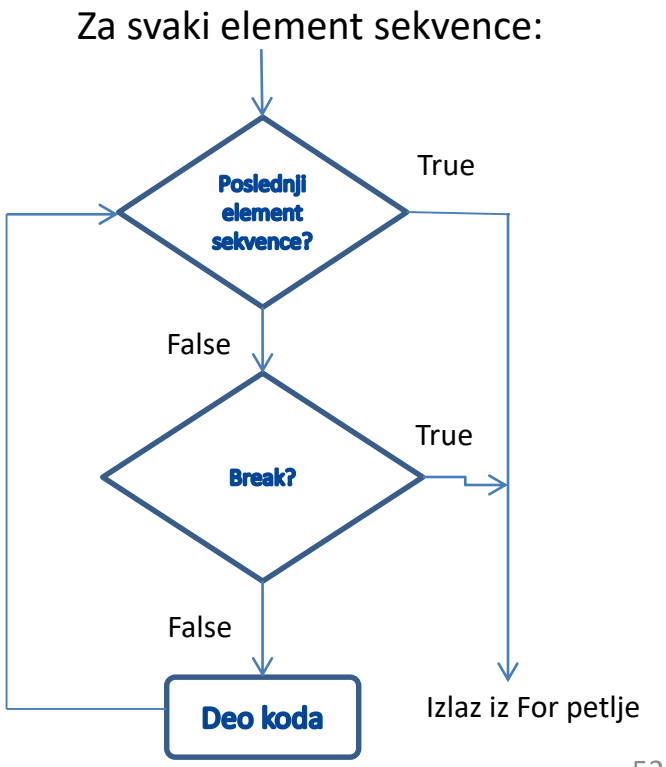
For petlja preko range() →

```
for n in range(10):
    deo koda
```

Python – For petlja, kontrolne opcije

- Kontrolne opcije menjaju menjaju uobičajni tok izvršenja petlji.
- Python podržava sledeće kontrole:
  - **Continue**: vraća kontrolu na početak petlje
  - **Break**: izvodi kontrolu van petlje (prekid petlje)





Python – funkcije

- **Funkcije** sadrže deo koda koji obavlja određeni zadatak i može se pozivati po potrebi i sa raznim ulaznim parametrima. Definise se kao:

ključna reč (def) ime funkcije(argumenti):

def pdv(x):

- def je ključna reč
- pdv() je ime funkcije
- x je argument

```
def pdv(x):
    opis funkcije (opciono)
    telo funkcije….
    y = x * 1.2
    .
    .
    .
    print(y)
    return y
```

Funkcija može vraćati i više vrednosti i tada se one grupišu pomoću malih zagrada, npr.

return(x, y, ...)

Python – funkcije

- Argumenti funkcije:
  - Obavezni: ovim argumentima se dodeljuju vrednosti prilikom poziva funkcije
  - Opcioni:
    - ovim argumentima se dodeljuju podrazumevane vrednosti prilikom definisanja funkcije; njima se prilikom poziva funkcije ne mora dodeljivati vrednost jer će se koristiti podrazumevana (osim ukoliko želimo drugu vrednost od podrazumevane, kada je to potrebno dodeliti prilikom poziva funkcije)
    - Opcioni argument se može pozvati i preko ključne reči

Python – funkcije

- Primer sa obaveznim argumentom x:

```
def pdv(x)
    y = x * 1.2
    return y
```

```
pdv(100)
```

Poziv funkcije **pdv**
sa jednim argumentom (obaveznim)

- Primer sa obaveznim argumentom x i opcionim argumentom stopa:

```
def pdv(x, stopa  = 20)
    y = x * (1 + stopa/100)
    return y
```

```
pdv(100)
```

Poziv funkcije **pdv**
sa obaveznim argumentom
(vrednost opcionog se
podrazumeva da je 20)

```
pdv(100, stopa=23)
```

Poziv funkcije **pdv**
sa obaveznim i opcionim
argumentom jer mu se vrednost
promenila od podrazumevane)

```
pdv(100, 23)
```

Poziv funkcije **pdv**
sa obaveznim i opcionim
argumentom jer mu se vrednost
promenila od podrazumevane,
bez ključne reči)

Python – datoteke (files)

- Kada se radi sa datotekama u Pythonu, datoteka prvo mora da se otvori i na kraju da se zatvori, dok se između manipuliše datotekom na željeni način.
- Bitno je naglasiti da se prilikom učitavanja datoteke trenutni folder u kome se *Spyder* nalazi mora podesiti da bude folder u kojem se nalazi datoteka.

Metode za rad sa datotekama:

Metoda	**Opis**
f = open('naziv_datoteke', mod)	Otvaranje datoteke. Komanda **open** ima dva argumenta: • prvi je obavezan i predstavlja naziv datoteke u vidu stringa • drugi je opcioni (mod) i pokazuje da li se datoteka koristi za čitanje ili za pisanje: •'r' (read), •'w' (write). Podrazumevana vrednost je 'r'. Može se u okviru mod specificirati da li otvara kao binarna ili tekstualna: • 'b' (binarna) • 't' (tekstualna) Podrazumevana vrednost je 't' tj. da se otvara kao tekstualna.

Python – datoteke (files)

Metoda	Opis
f.read()	Čitanje datoteke do kraja (čita se ceo sadržaj datoteke); pročitani sadržaj se vraća u vidu stringa. Dakle, ceo fajl je pretvoren u jedan string.
f.read(n)	Čitanje datoteke sa specificiranim brojem karaktera: iz datoteke se čita n karaktera (ili manje ukoliko se dođe do kraja), koji se vraćaju u vidu stringa.
f.readline()	Čitanje samo jednog reda datoteke, koji se vraća u vidu stringa. Datoteka se čita do znaka za novu liniju (\n), pri čemu se i sam znak \n uračunava u taj prvi string. Kada se opet pozove ova funkcija vraća se sledeći red opet do znaka \n, sve dok se ne pročitaju svi redovi.
f.readlines()	Čitanje datoteke do kraja. Rezultat koji vraća ova funkcija je lista redova (lista stringova).
f.write(s)	Upisivanje stringa s u datoteku. Ova funkcija vraća broj karaktera upisanih u datoteku.
f.close()	Zatvaranje datoteke. Nakon zatvaranja datoteke nije moguće vršiti nikakve manipulacije sa njom.

Treba naglasiti da se i datoteka može tretirati kao iterabilni objekat, tj. objekat koji ima članove koji su iterabilni; članovi su linije tj. delovi datoteke do \n. Dakle, može se koristiti u **for** petlji za traversing.

Python – biblioteke (moduli)

- Na početku je rečeno da je jedna od opštih karakteristika programskog jezika Python to što je on tzv. *modul based* programski jezik.
- To znači da se u samom programskom paketu (kada ga instaliramo), nalaze samo neke osnovne operacije, a da je sve ostalo izmešteno po modulima i importuje se po potrebi (uputstvo za programski jezik ima 147 strana a za module 2000 strana!!!).
- Postoji veliki broj biblioteka, a neke od najčešće korišćenih su:
  - **math:** modul za matematičke operacije
  - **string:** modula za manipulaciju stringovima
  - **numpy:** modul za rad sa nizovima i matricama
  - **matplotlib:** modul za vizuelizaciju rezultata (grafički prikaz)
- Često je za neke složenije probleme potrebno učitati više modula, pa postoje i tzv. okruženja koja integrišu više modula. Jedno od najčešće korišćenih je
  - **pylab (ima integrisane numpy i matplotlib)**

Python – biblioteke (moduli)

- Moduli **math** i **string** su deo standardnih biblioteka u Pythonu i nije ih potrebno posebno instalirati već samo importovati u željeni .py fajl.
- Ostali pomenuti moduli nisu deo standardnih biblioteka u Pythonu pa se moraju dodatno instalirati.
- Ukoliko se programski jezik Python koristi preko Anaconda distribucije, ona već sadrži ove module pa ih u tom slučaju nije potrebno dodatno instalirati, iako nisu deo standardnih biblioteka.
- Moduli se učitavaju (importuju) pomoću ključne reči *import*. Postoje različiti načini importovanja modula (primer je dat za modul **math**) :

Načini importovanja	**Primer korišćenja modula**	**Komentar**	**Način uklanjanja importovanog modula**
import math	math.sin(math.pi/2)	Pri korišćenju navodi se celo ime modula (**math**)	del math
import math as m	m.sin(m.pi/2)	Pri korišćenju navodi se skraćeno ime modula (**m**)	del m
from math import *	sin(pi/2)	Ceo modul **math** importovan u samo okruženje (postaje deo okruženja) i pri korišćenju ga nije potrebno navoditi	Ne može se ukloniti ceo modul već svako korišćenje pojedinačno (del sin, del pi, ...)

- Ukoliko se radi o modulu (numpy, matplotlib) koji je deo nekog okruženja koje integriše više modula (pylab), moguć je i način importovanja tako što se importuje celo okruženje umesto pojedinačnog modula:

```
from pylab import *
```

Python – biblioteke (moduli) **numpy**

from pylab import *
import numpy as np

- Modul **numpy** se koristi za rad sa nizovima i matricama (višedimenzionalnim nizovima). Iako podsećaju na liste, svi elementi niza moraju biti istog tipa i indeksiranje je moguće samo nenegativnim brojevima.
- Postoji veliki broj funkcija za manipulaciju nizovima i matricama, neke od njih su:

Vrsta operacije	**Simbol**
Kreiranje niza (1D) x	x = array([1, 3, 6])
Kreiranje matrice (2D niza) y	y = array([[1,2], [3,4]])
Pristup elementima niza x ili elementima matrice y	x[i] y[i, j]
Zbir elemenata niza x ili zbir elemenata matrice y	sum(x) sum(y)
Inicijalizacija niza dužine N ili matrice dimenzije N x M sa svim nulama	zeros(N) zeros((N, M))
Inicijalizacija niza dužine N ili matrice dimenzije N x M sa svim jedinicama	ones(N) ones((N, M))
Kreiranje jedinične matrice N x M (na dijagonali jedinice a na svim ostalim mestima nule); N je broj vrsta, M broj kolona; ako nema M smatra se da je M=N	eye(N, M)
Kreira niz brojeva u opsegu [a, b], sa korakom k	arange(a, b, k)
Daje novi oblik (K x L) nizu/matrici x bez izmene sadržaja	reshape(x, (K, L))

Python – biblioteke (moduli) **numpy**

Neki parametri matrice/niza a:

Vrsta operacije	**Simbol**
Broj elemenata u nizu/matrici a	a.size
Oblik niza/matrice (broj vrsta i kolona) a	a.shape
Dimenzija niza/matrice a	a.ndim
Tip elemenata niza/matrice	a.dtype
Veličina jednog elementa niza/matrice u bajtima	a.itemsize

Neki poznati parametri matrice a:

Vrsta operacije	**Simbol**
Transponovana matrica	transpose(a)
Determinanta	det(a)
Sopstvene vrednosti	eig(a)
Inverzna matrica	inv(a)

Python – biblioteke (moduli) **numpy**

- Sve matematičke operacije sa nizovima i matricama (sabiranje, množenje, deljenje, stepenovanje, ...) su *elementwise* što znači da se primenjuju na svaki član posebno.

- Npr. ako je u pitanju operacija množenja *, onda se ona sprovodi množenjem odgovarajućih elemenata niza/matrice (elemenata istih indeksa).

- Ali ukoliko želimo baš množenje matrica a i b (onako kako smo učili iz matematike, posmatramo ih kao objekat), onda se koristi funkcija **dot(a, b).**

- Isto važi i za nizove gde je operator množenja * *elementwise* operacija, dok se skalarni proizvod dva niza x i y dobija preko funkcije **dot(x, y)**

a * b - *elementwise,* množenje element po element matrica a i b

dot(a, b) **-** množenje matrica a i b

Python – biblioteke (moduli) **matplotlib**

from pylab import *
import matplotlib.pyplot as plt

- Matplotlib je modul koji se koristi za vizuelizaciju rezultata.
- Vlastito ime modula je matplotlib.pyplot pa otuda i alternativni import prikazan u naslovu.
- Neke od često korišćenih funkcija ovog modula su:

Vrsta operacije	**Simbol**
Crta y versus x kao linije (ako ne kažemo drugačije) (x i y su vektori istih dimenzija)	plot(x, y)
Crta y versus x kao markere željenog oblika i boje (u primeru je marker oblika + crvene boje (**r**ed))	plot(x, y, 'r+')
Označava ose	xlabel('x osa') xlabel('y osa')
Označava naziv slike (u primeru Signali)	title('Signali')
Ubacuje legendu (u primeru sin i cos)	legend(('sin', 'cos'))
Definiše granice a i b po osama u okviru kojih ce se prikazivati slika	xlim((a, b)) ylim((a, b))
Ekvidistantna linearna podela izmedju start i stop, u num tačaka	linspace(start, stop, num)
Specificiranje podelaka na osama	xticks(linspace(a, b, k)) yticks(linspace(a, b, k))
Omogućava promenu osa	axis()
Crta histogram za x tačaka u num binova histograma	hist(x, num)
Crtanje diskretnih signala (za svako x crta vertikalne linije duzine y i postavlja na tu tačku marker)	stem(x, y) Primer: stem(k, sin(k)) – slika sledeći slajd

Python – biblioteke (moduli) **matplotlib**

Vrsta operacije	**Simbol**
Generiše n slučajnih brojeva sa uniformnom raspodelom izmedju 0 i 1	rand(n)
Generiše n slučajnih brojeva u normalnoj raspodeli	randn(n)
Prikazuje više signala na posebnim graficima u okviru jedne slike (figure); generiše matricu grafika u jednoj figuri dimenzija m x n i smesta grafik na polje k (m je broj vrsta a n broj kolona)	subplot(m, n, k)
Snima sliku u .png formatu i smešta je u radni folder	savefig('slika')
Snima sliku u željenom formatu (pdf) i smešta je u radni folder	savefig('slika.pdf')