Predmet: "Vještačka inteligencija"
Laboratorijska vježba 1: Uvod u Python (1/2)
Odgovorna nastavnica: Vanr. prof. dr Amila Akagić

Sadržaj vježbe:

1 Cilj vježbe

Cilj vježbe je upoznavanje sa osnovama programskog jezika Python, kao i nekih često korištenih paketa u sklopu tog jezika. Pored toga, vježba daje kratki opis razvojnih okruženja za jezik Python, sa posebnim naglaskom na Jupyter Notebook okruženje. Preduslov za izradu ove vježbe je osnovno poznavanje programiranja, kao i rada u terminalu.

Treba napomenuti da je Python, kao i ekosistem oko njega, ogroman, te da nije moguće detaljno pokriti ovaj jezik. Studentima se preporučuje da konsultuju i online literaturu (tutorijale) za Python, kao i za prateće biblioteke.

2 Programski jezik Python

Python je popularni skripting jezik, koji je besplatno dostupan na stranici `www.python.org`. Danas je jedan od najpopularnijih programskih jezika, te se, u vrijeme pisanja ovih materijala, nalazi na trećem mjestu TIOBE indeksa, rang liste popularnost programskih jezika (prvo mjesto zauzima C, dok je Java na drugom mjestu). Python predstavlja jezik visokog nivoa, iznad jezika kao što su Java, C, C++, Assembly, itd. Program napisan u Python-u se naziva skripta, iz razloga što se Python kod ne kompajlira (kao npr. u slučaju jezika C), već se interpretira. To znači da se kod prevodi u mašinske instrukcije "u hodu", te se ne kreira izvršna datoteka (`.exe` na Windows sistemima). Umjesto toga, program prevodioc se pokreće, i prevodi Python skriptu. Najnovija verzija Python jezika je verzija 3.9.1.

Slika 1: Programski jezik Python.

2.1 Instalacija

Instalacija jezika Python je jako jednostavna. Potrebno je prije svega otići na stranicu na kojoj se može preuzeti Python:

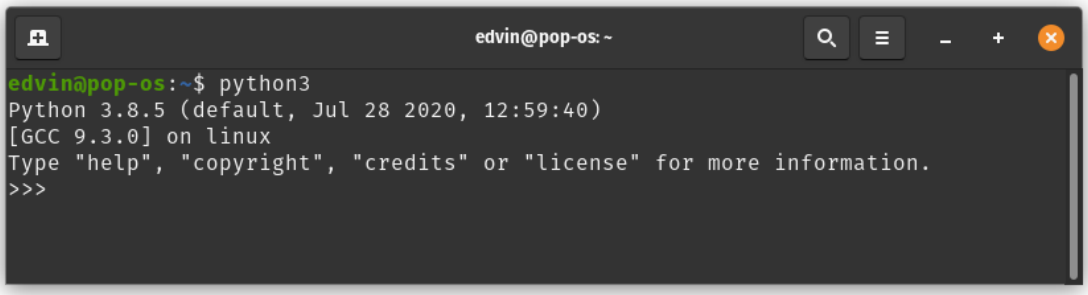
https://www.python.org/downloads/

Nakon toga, potrebno je odabrati odgovarajući operativni sistem (Microsoft Windows, GNU/Linux, Mac OS X, itd.) - dovoljno je kliknuti na dugme *Download*, te pratiti upute.

Nakon preuzimanja i instalacije, Python je spreman za korištenje. Može se pozvati interaktivna konzola direktno iz terminala (ili command prompt-a, u slučaju da koristite Windows operativni sistem), tako što se ukuca naredba:

```
python3
```

Rezultat toga je dat na slici 2:



Slika 2: Korištenje jezika Python direktno iz terminala.

U interaktivnoj konzoli se mogu kucati naredbe koje će se direktno izvršavati. Međutim, ovo nije preferirani način korištenja jezika Python - u nastavku vježbe ćemo se upoznati sa pisanjem Python skripti, kao i sa najpopularnijim razvojnim okruženjima za ovaj jezik. Kako bi se napustila interaktivna konzola, dovoljno je ukucati naredbu

```
exit()
```

Kako bi se izbjeglo direktno kucanje naredbi kroz konzolu, moguće je kreirati gotove datoteke sa Python kodom, koji se onda mogu pokrenuti iz terminala. U Python-u, takve datoteke se nazivaju moduli, te imaju ekstenziju `.py`. Ovo je zapravo ekvivalent `.c` datoteci iz jezika C. Te datoteke je moguće napisati u proizvoljnom tekstualnom uređivaču, te se nakon što se sačuvaju mogu pozvati iz terminala, naredbom:

```
python3 [nazivDatoteke].py
```

2.2 Osnovne naredbe, operatori, i kontrola toka

Jezik Python je dobio na popularnosti zbog svoje jednostavne sintakse. U ovom jeziku, *Hello World* program[1], koji se obično koristi za upoznavanje sa nekim novim programskim jezikom, je izuzetno jednostavan, te se sastoji od samo jedne linije:

```
1 print("Hello World!")
```

Ukoliko se ovaj jednostavni *Hello World* program uporedi sa analognim programom napisan u, na primjer, jeziku Java, može se uočiti razlika u količini napisanog koda, kao i u nedostatku elemenata kao što su tačka-zarez (`;`), te vitičastih zagrada (`{` i `}`). Također možemo primijetiti da su skripte pisane u Pythonu izuzetno lagane za čitati, čak i nekome ko nikada prije nije programirao. Iz tog razloga je ovaj jezik popularan za prototipiranje raznih rješenja - dozvoljava da na brz i jednostavan način testiramo algoritam, prije nego se isti implementira u nekom bržem jeziku (npr. C ili C++).

Python je dinamički i interpretirani jezik. To znači da ne postoji deklaracija tipova, varijabli, funkcija, metoda, itd. Upravo ovo omogućuje da kod koji je napisan u Python-u bude jednostavan i fleksibilan. Interno, Python prati sve

[1] Strogo gledano, u kontekstu jezika Python se ne može govoriti o programima, već samo o skriptama, međutim, zbog jednostavnosti ćemo u nastavku koristiti termin program.

tipove unutar izvornog koda, te će prilikom interpretiranja javiti grešku ukoliko dođe do iste (dok će se, kod jezika kao što je C, to desiti odmah pri pokušaju kompajliranja). Jednostavan način za eksperimentisanje sa Python-om za početnike jeste kroz ranije opisanu interaktivnu konzolu. Unutar nje, moguće je ukucati bilo koju naredbu, te vidjeti šta će se desiti. Primjer u nastavku to ilustruje:

```
$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 4 # postavi vrijednost varijable x na 4
>>> print(x) # ispisi varijablu
4
>>> print(x+2) # uvecano za dva
6
>>> x = 'vjestacka inteligencija' # sada je x postalo string
>>> print(x)
vjestacka inteligencija
>>> print(len(x)) # duzina stringa
23
>>> print(x + len(x)) # nesto sto ne radi
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> print(x + str(len(x)) # nakon konverzije ce raditi
vjestacka inteligencija23
>>> exit() # da zavrsimo sa radom
```

Iz primjera se može vidjeti da je jednostavno eksperimentisati sa varijablama i operatorima. Pored toga, Python interpreter će dati grešku ukoliko neka naredba nema smisla (kao što je u primjeru iznad sabiranje string-a i int-a). Također, primjer koristi naredbu `len()`, koja daje dužinu stringa. Ova naredba se također koristi za dobijanje dužine listi.

Jezik Python razlikuje velika i mala slova, pa tako varijable `x` i `X` predstavljaju različite vrijednosti. Komentari u Python-u počinju sa znakom rešetke (#), koji je ekvivalentan `//` u jeziku C. Ukoliko se želi zakomentarisati više linija, onda se to može postići pomoću znakova `'''` na početku i kraju bloka koji se zakomentariše (ekvivalentno `/*` i `*/` u jeziku C).

2.2.1 Operatori

Python podržava iste operatore kao i drugi jezici, uz male razlike. Prvo, jezik Pyhton ne podržava operatore ++ i −−, već se umjesto njih koriste operatori + i −. Pored toga, Python podržava operator `**`, što predstavlja operator stepenovanja. Na primjer, ukoliko se želi izračunati vrijednost x^4 za neku varijablu `x`, pisalo bi se:

```
rezultat = x ** 4
```

Još jedan operator koji jezik Python podržava jeste operator `//`, odnosno cjelobrojno dijeljenje. Na primjer, ukoliko je varijabla `x` vrijednost 5, onda bi `x // 2` bilo jednako dva.

Pored aritmetičkih operatora, Python poznaje i logičke operatore, kao i operatore pripadnosti. Ovi operatori su opisani u tabelama 1 i 2, respektivno.

Operator	Opis	C ekvivalent	Primjer
`and`	Vraća `True` ukoliko su oba iskaza tačna	`&&`	`x < 4 and x > 0`
`or`	Vraća `True` ako je barem jedan iskaz tačan	`||`	`x < 0 or x > 11`
`not`	Negira rezultat	`!`	`not(x >= 10)`

Tabela 1: Python logički operatori.

Operator	Opis	C ekvivalent	Primjer
`in`	Vraća `True` ukoliko je $x \in y$	Ne postoji	`x in y`
`not in`	Vraća `True` ukoliko je $x \notin y$	Ne postoji	`x not in y`

Tabela 2: Python operatori pripadnosti.

2.2.2 Kontrola toka

Pod kontrolom toka podrazumijevaju se osnovne petlje, kao i *if-else* struktura. U nastavku će biti prikazan primjer *for*-petlje, *while*-petlje, kao i jednostavne *if-else* strukture.

```
def harm(n):
    suma = 0
    for i in range(1, n+1):
      suma += 1/i
    return suma

def main():
    n = -1
    while n <= 0 or n > 15:
        n = int(input('Unesi pozitivan broj: '))
        if n <= 0:
          print('Broj nije pozitivan!')
        elif n > 15:
          print('Broj mora biti manji ili jednak 15!')

    print('Harmonijska suma je:', harm(n))

main()
```

Ovaj primjer od korisnika traži unos broja n, te provjerava da li je taj broj između 1 i 15. Ukoliko jeste, računa se harmonijska suma dužine n po formuli

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i}$$

Ukoliko korisnik unese broj koji nije u traženom opsegu, unos broja se ponavlja.

Prvo se u programu, pomoću ključne riječi `def` definiše nova funkcija `harm`, koja kao ulazni argument prima neku varijablu `n`. U samoj funkciji se koristi for petlja, te se opseg for petlje definiše funkcijom `range`. Nakon što se izračuna harmonijska suma, ona se iz funkcije vrati pomoću komande `return`. Nakon toga je definisana `main` funkcija, koja predstavlja ulaznu tačku za skriptu. Unutar nje se od korisnika prvo traži unos nekog broja pomoću naredbe `input`. Unos je potrebno eksplicitno pretvoriti u broj, tako što se koristi naredba `int` (Python bilo kakav ulaz sa tastature u pozadini smatra stringom, tako da je potrebna ova eksplicitna konverzija). Nakon toga se, pomoću `if` konstrukcije provjerava da li je uneseni broj pozitivan. Ukoliko nije, ispisuje se odogavarajuća poruka. Nakon toga, provjerava se da li je broj veći od 15, i to pomoću `elif` (ekvivalent `else if` iz C). Ovaj proces se pomoću `while` petlje ponavlja sve dok se ne unese validna vrijednost za broj `n`. Nakon što se izađe iz `while` petlje, ispisuje se vrijednost harmonijske sume dužine `n`. Konačno se, iz najvišeg nivoa skripte (onog bez imalo indentacije), poziva `main` funkcija.

Treba naglasiti da jezik Python ne poznaje konstrukcije kao što su `{ i }` u jezicima kao što su C, već se pripadnost dijela koda određenoj funkciji, petlji, if-uslovu, itd. određuje na osnovu nivoa indentacije.

2.3 Liste

Python sadrži ugrađeni tip *list*, koji zapravo predstavlja osnovnu vrstu kolekcija (kao `std::vector` u jeziku C++). Rad sa listama je identičan kao i rad sa kolekcijama u drugim jezicima. Indeksacija počinje od nule, te se koriste uglaste zagrade.

Jednostavan primjer korištenja liste je dat u nastavku (ovdje se također koristi i rangovska for-petlja):

```
kvadrati = [1, 4, 9, 16, 25]
suma = 0
for x in kvadrati:
  suma += x
print(suma)  # ispisuje 55
```

Pored standardne indeksacije, moguće je vršiti i indeksaciju unazad. Tako je `lista[-1]` zapravo posljednji element liste, `lista[-2]` pretposljednji, itd. Također je moguće indeksirati dio liste, pomoću operatora dvotačka (`:`). Na primjer, ukoliko je dat string `s = 'Hello'` (koji je zapravo lista karaktera), onda se može raditi indeksacija dijelova tog stringa (odnosno te liste):

- `s[-1]` - 'o', posljednji karakter;
- `s[-4]` - 'e', četvrti karakter s kraja;
- `s[1:4]` - 'ell' - počevši od indeksa 1, pa sve do indeksa 4 (ne uključujući i indeks 4);
- `s[1:]` - 'ello' - počevši od indeksa 1, pa do kraja;
- `s[:]` - 'Hello' - čitav string.

Ovdje se posljednja indeksacija koristi kako bi se napravila duboka kopija stringa ili liste.

2.4 Python moduli

Python okuplja veliki broj stručnjaka iz industrije i akademije, koji rade na razvoju novih biblioteka (modula), te tako znatno olakšavaju rad u Python-u, jer stavljaju na raspolaganje moćan skup alata, pomoću kojih možemo pisati komplikovane programe, bez da se brinemo o implementaciji samih algoritama koji su potrebni kako bi se ti programi izvršavali. Sa nekim od tih biblioteka ćemo se upoznati u nastavku ove vježbe. Neki predefinisani modul se može učitati pomoću ključne riječi `import` - ova naredba je ekvivalentna direktivi `#include` u jeziku C. Pored toga, moguće je učitati samo dijelove modula (kao što su određene funkcije, klase, konstante, itd.). U primjeru ispod, koristi se konstanta `pi` (π) iz biblioteke `math` kako bi se izračunala površina kruga:

```
from math import pi
r = 12
povrsina = pi * r ** 2
print("Povrsina kruga sa poluprecnikom", r, "je", povrsina)

# Alternativa:

import math
r = 12
povrsina = math.pi * r ** 2
print("Povrsina kruga sa poluprecnikom", r, "je", povrsina)
```

Ukoliko je potrebno preuzeti neku novu biblioteku, to se može lako uraditi iz terminala, koristeći komandu `pip3`, na primjer:

```
pip3 install numpy
```

Neki najpoznatiji Python moduli su:

- NumPy;
- Matplotlib;
- SciPy;
- Pandas;
- SkLearn;
- TensorFlow;

- Keras;
- Pickle.

Sa prve dvije biblioteke, NumPy i Matplotlib, ćemo se upoznati na današnjoj vježbi. Naredni put ćemo se upoznati sa ostalim Python bibliotekama.

Iz priloženog se može vidjeti da je Python izuzetno jednostavan za korištenje, te pruža brz način razvoja programa. Jednostavnost jezika omogućila mu je da postane popularan u raznim poljima primjene, od kojih su neka:

- Računarska Vizija;
- Robotika;
- Optimizacija;
- Bankarstvo;
- **Vještačka inteligencija.**

2.5 Python razvojna okruženja

Postoje mnoga okruženja (eng. *IDE - integrated development environment*) za rad u Python-u, od kojih su najpopularniji:

- PyCharm (https://www.jetbrains.com/pycharm/) - predstavlja profesionalni IDE za Python razvijen od strane Češke kompanije JetBrains. Pruža mogućnosti kao što su analiza koda, grafičko debagiranje, integracija sa kontrolom verzija (npr. git), unit tester, kao i podršku za razvoj web aplikacija sa Django kao i data science podršku uz Anakondu. Dostupan je za Windows, macOS, i GNU/Linux. Profesionalna verzija (sa dodatnim opcijama) je besplatna za studente;
- Anaconda (https://www.anaconda.com/) - distribucija jezika Python i R, namijenjena za grane kao što su data science, mašinsko učenje, procesiranje podataka, itd. Cilj Anakonde je pojednostaviti upravljanje paketima;
- Jupyter Notebook (https://jupyter.org/) - web-bazirano interaktivno okruženje za pravljenje Jupyter notebook dokumenata.

Na ovom predmetu, fokusirati ćemo se na Jupyter Notebook kao razvojno okruženje.

2.5.1 Jupyter Notebook

Jupyter Notebook je *open-source* web aplikacija koja omogućava kreiranje i dijeljenje dokumenata koji sadrže kod, tekst (LaTeX), jednačine, vizualizacije, itd. Jupyter podržava preko 40 programskih jezika - neki od njih su Python, R, Julia, i Scala.

Notebook predstavlja dokument koji sadrži kako programski kod, tako i tekstualne i grafičke elemente. Dokumenti pisani kao notebook su istovremeno čitljivi od strane čovjeka, jer sadrže analizu i prateći opis, kao i rezultate (grafove, tabele, slike, itd.), i izvršivi kao obični kod.

Svaki notebook se sastoji od ćelija. Ćelija može biti ili kodna (što znači da se u nju piše Python kod) ili tekstualna (npr. ćelije sa tekstom, jednačinama, itd. - ovdje postoji podrška za LaTeX i HTML sintaksu). Svaka ćelija se može posebno izvršavati. Na taj način je moguće odvojiti kod na dijelove, te izvršavati dio po dio. Na slici 3 je prikazan jedan jednostavan notebook fajl koji se sastoji od 4 ćelije. Kada se notebook sačuva, on je u `.ipynb` formatu.

1 Naslov - tekstualna ćelija

1.1 Podnaslov - primjer notebook fajla

[3]:
```
print('Jupyter Notebook - primjer')
x = 42
```

```
Jupyter Notebook - primjer
```

Kada se kodna ćelija iznad izvrši, rezultat se ispiše neposredno ispod ćelije. Ovo je tekstualna ćelija, može se koristiti Latex sintaksa, npr. za formule:

$$\sum_{i=0}^{n} i^2 \cdot \frac{0.4\xi}{\beta + \mu}$$

[4]:
```
# Kodne celije dijele isti prostor za varijable
print(x)
```

```
42
```

Slika 3: Jupyter Notebook.

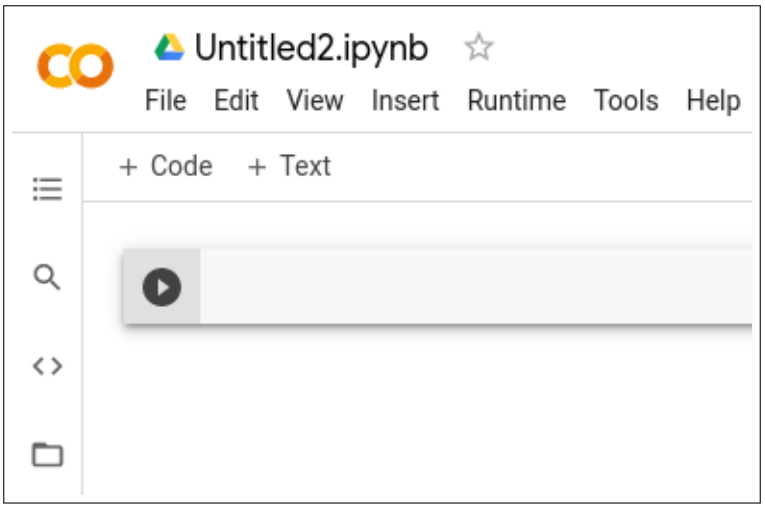
Jupyter Notebook je moguće preuzeti sa službene web stranice. Na taj način je moguće lokalno pokrenuti web server, te kroz browser praviti novi notebook dokument. Za potrebe vježbi na predmetu vještačka inteligencija, koristit će se Google Colab okruženje. Google Colab predstavlja online Jupyter Notebook server, koji se može direktno koristiti, bez da se rade ikakve dodatne instalacije. Za korištenje je potrebno otići na link

https://colab.research.google.com/

Za kreiranje novog notebook-a je potrebno odabrati

File → *New notebook*

Nakon toga, otvara se novi (prazan) notebook, kao što je prikazano na slici 4.



Slika 4: Google Colab okruženje - novi notebook.

Sada je moguće dodavti nove ćelije (tekstualne ili kodne), te raditi sa notebook-om. Kada se rad završi, moguće je notebook preuzeti, ili spasiti kako bi se rad nastavio poslije.

2.6 Rad sa n-dimenzionalnim nizovima - NumPy

Numpy (**Num**erical **Py**thon) je *open-source* Python biblioteka koja se koristi u gotovo svakom polju nauke i inžinjeringa. Predstavlja univerzalni standard za rad sa numeričkim podacima u Python-u.

NumPy biblioteka sadrži multidimenzionalne nizove i matrične strukture podataka. U samom centru je `ndarray`, homogeni n-dimenzionalni nizovni objekat, koji sadrži metode za efikasan rad nad istim. Ovdje riječ homogeni označava da se radi o nizovima u kojima su svi elementi istog tipa. NumPy se prvenstveno koristi za izvršavanje raznih matematskih operacija nad nizovima različitih dimenzija. Učitavanje ove biblioteke u neki notebook je veoma jednostavno, te se radi sa ranije spomenutom `import` naredbom:

```
import numpy as np
```

Ovdje smo biblioteci dali alias np, kako ne bismo morali uvijek pisati puni naziv biblioteke (na primjer, sada je moguće pisati `np.zeros` umjesto `numpy.zeros`).

Prednost NumPy nizova nad običnim Python listama leži u tome što su NumPy nizove memorijski kompaktniji, te pružaju više različitih opcija. Također, operacije nad NumPy nizovima su znatno brže od operacija nad listama. Ti nizovi zapravo predstavljaju tabelu elemenata (obično brojeva), te se mogu indeksirati kao i nizovi u jezicima kao što je C.

Klasa `ndarray` ima nekoliko bitnih metoda:

- `ndarray.ndim` - vraća broj dimenzija niza (npr. 2 za matrice);
- `ndarray.shape` - vraća dimenzije niza. Na primjer, za matricu sa n redova i m kolona, `shape` će vratiti `(n, m)`;
- `ndarray.size` - ukupni broj elemenata u nizu - ovaj broj je jednak proizvodu svih brojeva koje vrati metoda `shape`;
- `ndarray.itemsize` - veličina u bajtima svakog elementa niza (npr., za tip `float64` bi ova metoda vratila 8);
- `ndarray.dtype` - daje tip podataka koji se nalaze u nizu.

U nastavku je dat primjer koji koristi neke od nabrojanih metoda:

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype
dtype('int64')
>>> a.itemsize
8
>>> a.size
15
>>> b = np.array([42.0, 42.1, 42.2])
>>> b
array([42. , 42.1, 42.2])
>>> b.dtype
dtype('float64')
```

Iz primjera se vidi da postoji nekoliko načina da se kreira NumPy niz. U prvom slučaju koriste se funkcije `arange` i `reshape`. Prva funkcija vraća niz od prvih n brojeva, počevši od nula, gdje se n predaje kao parametar. Druga funkcija uzima postojeći NumPy niz, te ga oblikuje u skladu sa proslijeđenim parametrima. U ovom slučaju su proslijeđena dva parametra, 3 i 5, što znači da će se niz oblikovati u 2D niz (matricu) sa 3 reda i 5 kolona. Drugi način je kreiranje NumPy niza direktno pomoću Python listi. U tu svrhu, koristi se funkcija `array`, kojoj se proslijedi Python lista.

Pored opisanih funkcija, važne su i funkcije `np.zeros` i `np.ones`. Ove funkcije su analogne funkcijama istog imena u jezicima GNU Octave, Matlab, i Scilab. Funkcija `zeros` kreira niz odgovarajućih dimenzija, u kojem su svi elementi nula, dok funkcija `ones` radi isto to, samo što će sada svi elementi niza biti jednaki 1. Dodatno, funkcija

`empty` kreira niz zadanih dimenzija ispunjen nasumičnim vrijednostima. Primjer ovih funkcija je dat u isječku koda ispod:

```
>>> np.zeros((3,4))
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
>>> np.ones((2,3,4)) # niz matrica
array([[[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]],

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
>>> np.empty((3,3))
array([[6.91842574e-310, 6.91842574e-310, 6.91841419e-310],
       [6.91841602e-310, 6.91841419e-310, 6.91841419e-310],
       [6.91841419e-310, 6.91841419e-310, 6.91841419e-310]])
```

Nad NumPy nizovima je moguće vršiti razne operacije. Prije svega, tu su standardne operacije kao što su sabiranje, oduzimanje, množenje (operator `*`), matrično množenje (operator `@`), itd. Neke jednostavne operacije su prikazane u nastavku:

```
>>> a = np.array( [20, 30, 40, 50] )
>>> b = np.arange(4)
>>> b
array([0, 1, 2, 3])
>>> c = a - b
>>> c
array([20, 29, 38, 47])
>>> b ** 2
array([0, 1, 4, 9])
>>> 10 * np.sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])
>>> a < 35
array([ True,  True, False, False])
>>> A = np.array([[1,1], [0,1]])
>>> B = np.array([[2,0], [3,4]])
>>> A * B
array([[2, 0],
       [0, 4]])
>>> A @ B
array([[5, 4],
       [3, 4]])
```

Mnoge druge Python biblitoke ovise u NumPy i tipovima implementiranih u sklopu istog. Pored prikazanih primjera, postoji ogroman broj funkcija koje NumPy pruža, te se zainteresovanim studentima preporučuje dokumentacija ove biblioteke, koja se nalazi na linku

https://numpy.org/doc/stable/user/

2.7 2D grafika u jeziku Python - Matplotlib

Matplotlib je jedna od najpopularniji Python biblioteka za vizualizaciju podataka. Pomoću ove biblioteke, moguće je praviti 2D grafike na osnovu podataka sačuvanih u nizovima. Pruža objektno-orijentisani API koji u pozadini koristi NumPy. Biblioteka je inspirisana načinom na koji se kreiraju grafici u jeziku Matlab.

Osnovni način korištenja Matplotlib je kroz Pyplot. Pyplot je kolekcija funkcija koje omogućavaju da Matplotlib radi na istom principu kao i Matlab. Svaka Pyplot funkcija pravi neku izmjenu nad grafikom. Na primjer, funkcija

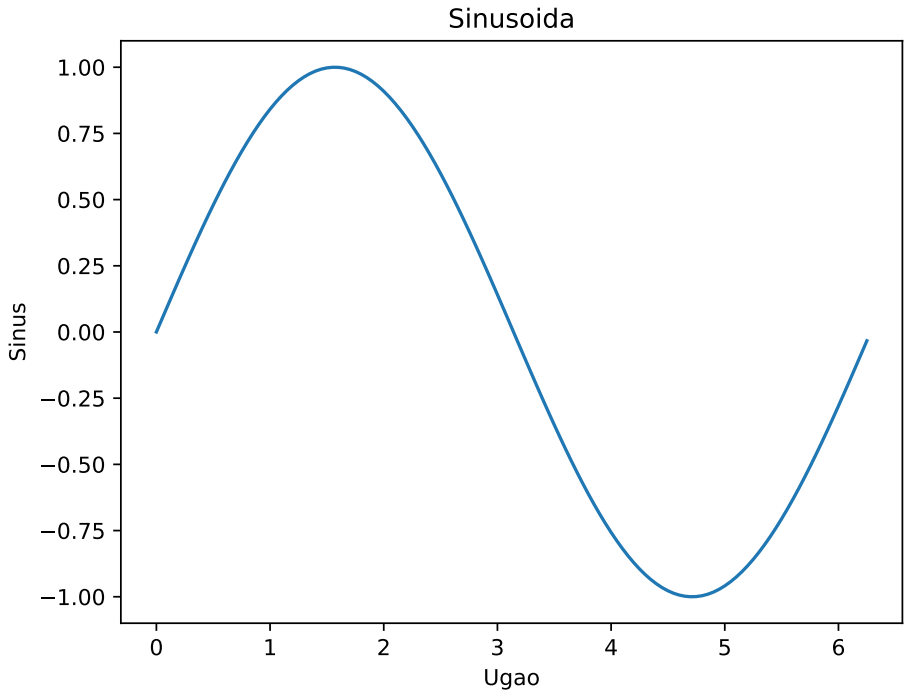
može kreirati grafik, kako bi zatim druga funkcija iscrtala liniju, treća dodala legendu, itd. Kako bi se učitao Pyplot u notebook, koristi se linija

```
import matplotlib.pyplot as plt
```

Sada je moguće na konkretnom primjeru napraviti jednostavan grafik sinusiode:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, np.pi*2, 0.05)

y = np.sin(x)

plt.plot(x, y)
plt.xlabel('Ugao')
plt.ylabel('Sinus')
plt.title('Sinusoida')

plt.show()
```

Prvo se učitaju potrebne bibliteke - `matplotlib.pyplot` i `numpy`. Nakon toga je potrebno napraviti vrijednosti za x osu. U ovom slučaju, to će biti vrijednosti od 0 do 2π, u razmacima od 0.05. Nakon toga, kreiraju se vrijednosti po y osi - to će biti sinus vrijednosti sa x ose (primijetiti kako pomoću `np.sin` možemo uzeti sinus svakog elementa čitavog NumPy niza x). Nakon ovoga, slijedi iscrtavanje grafika pomoću pyplot. Komanda `plot` služi za iscrtavanje, dok komande `xlabel`, `ylabel`, i `title` postavljaju tekst po osama i naslov grafika. Kada je sve spremno, može se prikazati grafik, i to pomoću funkcije `show`. Rezultat izvršavanja ovog koda je dat na slici 5.



Slika 5: Dobijeni grafik.

Matplotlib sadrži mnogo opcija, koje se mogu pronaći u dokumentaciji na linku

https://matplotlib.org/3.3.3/tutorials/index.html

3 Zadaci za rad u laboratoriji

Predviđeno je da se svi zadaci u nastavku rade u sklopu Jupyter Notebook okruženja. Svaki podzadatak treba biti zasebna Jupyter ćelija.

Zadatak 1 - Osnove Pythona

Pomoću Python-a izračunati vrijednost sljedećih izraza:

a) $13 \cdot 2.5^{\frac{124}{9.3}} + \sqrt{3}$

b) $\cos\frac{\pi}{3} + \tan\pi \cdot e^{\pi}$

c) $(2 + 3^8) \cdot 10$

d) $\left\lfloor \frac{79}{18} \right\rfloor$

Zadatak 2 - Liste i kontrolne strukture

a) Napisati Python program koji prvo od korisnika traži unos 10 elemenata liste. Nakon toga, na ekranu ispisati drugi najveći element u listi;

b) Napisati Python program koji za početnu vrijednost n računa n-ti Fibonacci broj. Obavezno koristiti for petlju;

c) Napisati Python program koji za zadane vrijednosti a, b, i c računa vrijednosti rješenja kvadratne jednačine, tj.

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

U slučaju da je diskriminanta negativna, napisati "Rješenja su kompleksna!" i ne računati rješenja. U slučaju da je diskriminanta nula, ispisati "Rješenja su ista!" i ispisati rješenje. Obavezno koristiti if strutkuru.

d) Definisati funkciju `fact(n)` koja računa faktorijel broja n. Zatim definisati funkciju `choose(n, k)` koja računa $\binom{n}{k}$. Demonstrirati rad ovih funkcija na primjeru.

Zadatak 3 - Rad sa matricama

Pomoću NumPy, generisati sljedeće matrice:

a) Matricu dimenzija 4×3 čiji su svi elementi nule;

b) Matricu dimenzija 2×2 čiji su svi elementi jedinice;

c) Vektor-kolonu dužine 10 čiji su elementi brojevi od 1 do 10;

d) Matricu datu u nastavku:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ \frac{1}{3.6} & 5 & 23 \\ 2^{10.5} & 42 & \cos(80.841) \end{bmatrix}$$

e) Za prethodno kreiranu matricu, izračunati:

   (a) Transponovanu matricu;

   (b) Matricu dobivenu sabiranjem transponovane matrice i originalne matrice;

   (c) Proizvod matrice sa sumom njene prve kolone.

f) Za matricu M prikazati:

   (a) Treći red matrice;

   (b) Drugu kolonu matrice;

   (c) Element na lokaciji (0, 2).

g) Definisati funkciju `kvadratna(M)` koja prima NumPy 1D niz, te isti preoblikuje u kvadratnu matricu, ukoliko je to moguće. Ukoliko nije moguće, funkcija ne treba da uradi ništa.

Zadatak 4 - Rad sa grafikom

a) Neka je data funkcija $x = \sin(n)$ na 100 tačaka (n = 1:100). Iscrtati linijski dijagram ove funkcije. Postaviti odgovarajuće labele, te naslov. Zatim preko tog dijagrama (na isti graf) nacrtati funkciju $y = \cos(n)$, također na 100 uzoraka. Uključiti mrežu (*grid*);

b) Ponoviti prethodni zadatak, ali sada iscrtati u istom prozoru (ali na različitim graficima) signale x i y. Postaviti odgovarajuće labele i naslove za oba grafika;