"Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them."


Co-funded by the European Union

**Univerzitet u Bihaću**
**Tehnički fakultet**

# OSNOVE PROGRAMIRANJA PYTHON

Una Drakulić MA, Melisa Haurdić MA

**UNBI**





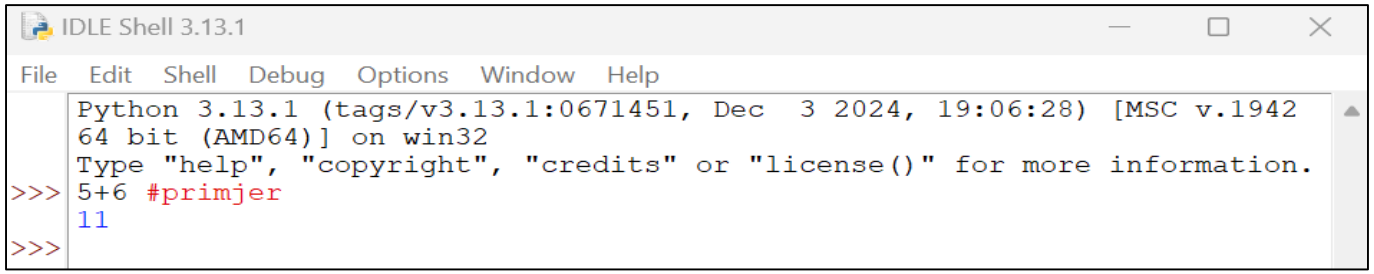
Co-funded by the European Union

# 1. Uvod u Python

- Varijable u Pythonu nemaju oznaku tipa iako pojam tipa podatka postoji. To znači da nekoj varijabli možemo pridružiti vrijednost bilo kojeg tipa.
- U Pythonu memorija se oslobađa automatski. Drugim riječima, ne postoji naredba kojom se eksplicitno uklanja ili dealocira objekt iz memorije. To je omogućeno sakupljačem smeća (engl. garbage collector) sistemom, sistemom za automatsko upravljanje memorijom koji je dio Pythonovog izvršnog sistema.
- Program pisan u Pythonu ne prevodi se na prirodni kôd (engl. native code) dotičnog računala nego se izvršava pomoću drugog programa koji se zove interpreter.

# 1. Uvod u Python

- Python se isporučuje sa standardnim razvojnim okruženjem koje se zove IDLE (Integrated Development and Learning Environment). Nakon pokretanja IDLE-a dobije se komandna linija u koju se može unositi programski kod. Nakon unosa, po pritisku tipke Enter ili Return ispod same linije, dobije se rezultat unesenog koda.

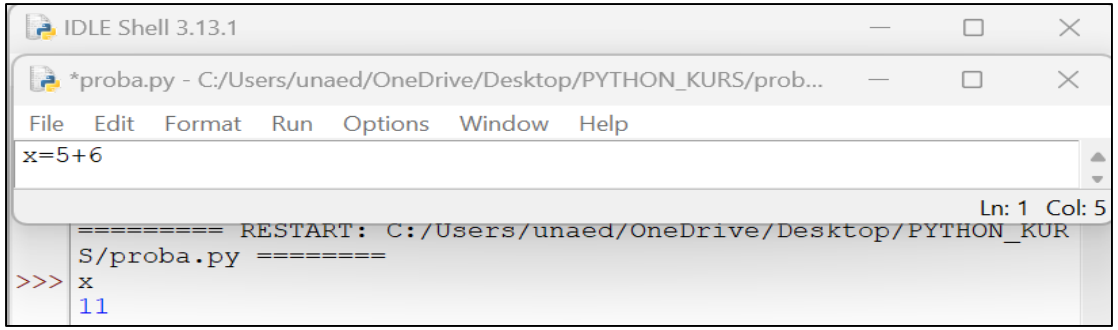



Slika 1. *Primjer sabiranja dva broja koristeći IDLE Shell*

- Tekst iza oznake # se smatra komentarom i ne izvršava prilikom pokretanja koda.

# 2. Osnovni pojmovi

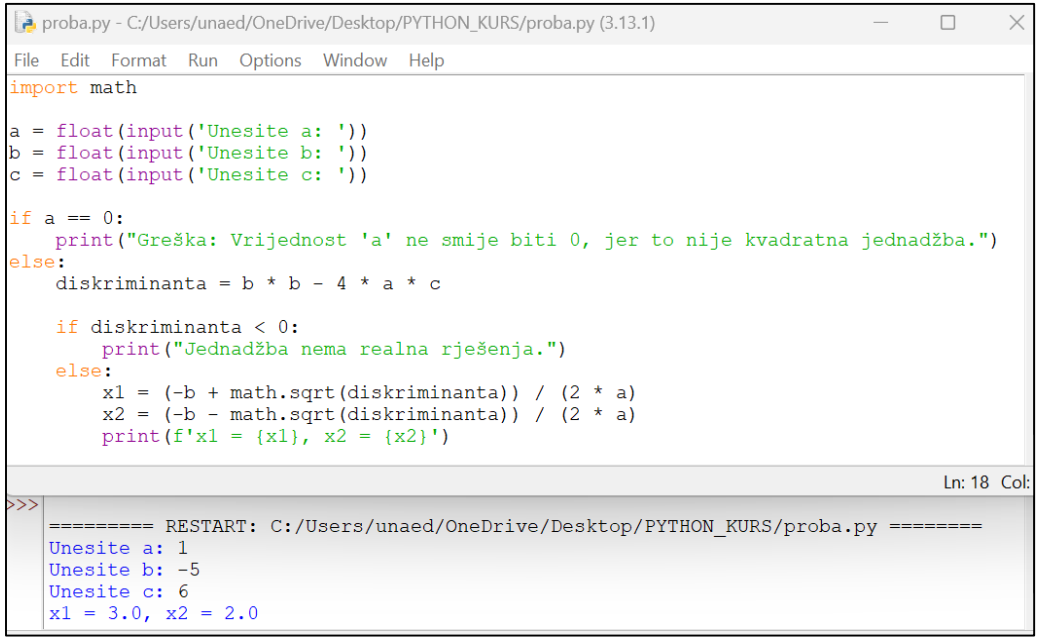
- Izraz je bilo koji konstrukt programskog jezika čije izvršavanje vraća neku vrijednost kao rezultat ($5 + 6$ je izraz, jer daje neki rezultat, tj. vrijednost $11$; $5 < 6$ je logički izraz, čiji rezultat u ovom slučaju bude $1$ ili True).
- Naredbe nemaju definisan rezultat ($x = 5 + 6$ predstavlja naredbu pridruživanja). Kada nekoj varijabli pridružimo određenu vrijednost, ona joj ostaje pridružena sve dok toj varijabli ne pridružimo neku drugu vrijednost.




Slika 2. *Pozivanje vrijednosti x iz modula proba.py*

# 2. Osnovni pojmovi

- Funkcija *print* služi za ispis rezultata svih izraza.

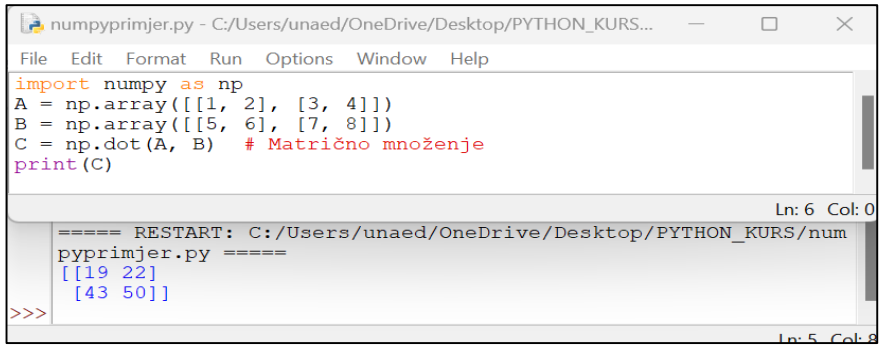



Slika 3. *Korištenje naredbe print za različite tipove podataka*

*Math* je ugrađeni Python paket koji pruža matematičke funkcije i konstante.
Najčešće korištene su:
- math.sqrt(x);
- math.pow(x,y);
- math.sin(x);
- math.cos(x);
- math.pi.

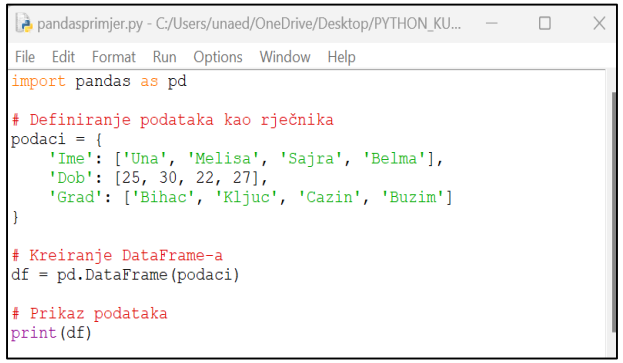
# 2. Osnovni pojmovi

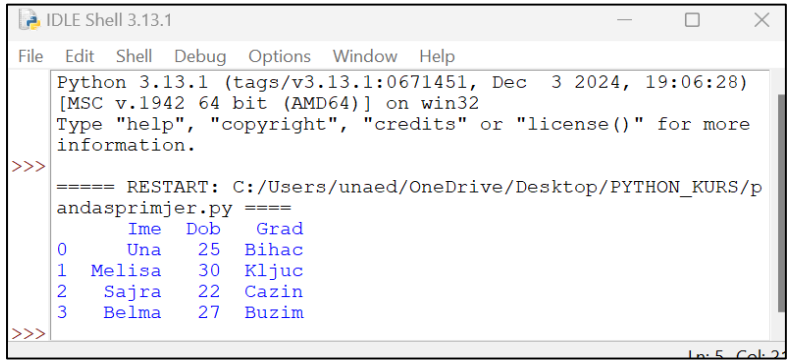
- *NumPy* (Numerical Python) je osnovni paket za numeričke proračune.




Slika 4. *Primjer upotrebe numpy paketa*

- *Pandas* je paket za obradu i analizu podataka.




Slika 5. *Primjer upotrebe pandas paketa*

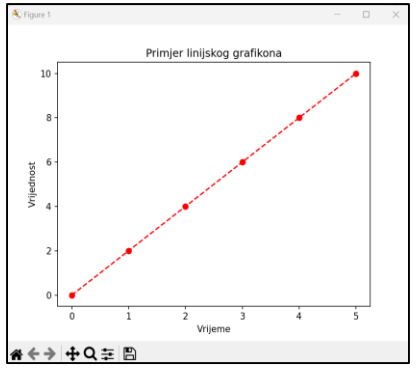
# 2. Osnovni pojmovi

- *Matplotlib* je standardni paket za crtanje grafova.

matplotlibprimjer.py - C:/Users/unaed/OneDrive/Desktop/PY...

File Edit Format Run Options Window Help

```
import matplotlib.pyplot as plt
x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
plt.plot(x, y, marker='o', linestyle='--', color='r')
plt.xlabel('Vrijeme')
plt.ylabel('Vrijednost')
plt.title('Primjer linijskog grafikona')
plt.show()
```




Slika 6. *Primjer upotrebe matplotlib paketa*

- *Scikit-learn* je paket za klasične algoritme mašinskog učenja.

scikitprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/scikit...

File Edit Format Run Options Window Help

```
from sklearn.linear_model import LinearRegression
import numpy as np

X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2, 4, 6, 8, 10])

model = LinearRegression()
model.fit(X, y)
print(model.predict([[5]]))  # Predviđanje za X=5
```

Ln: 9 Col: 0

```
>>>
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/sciki
tprimjer.py ====
[10.]
```

Ln: 4 Col: 66

Slika 7. *Primjer upotrebe scikit-learn paketa*

# 2. Osnovni pojmovi

- *TensorFlow/PyTorch* paket za duboko učenje. Način instalacije paketa je korištenje naredbe: pip install tensorflow, pip install torch, a primjer upotrebe je prikazan na slici 8.

tensorflowprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/tensorflowprimjer.py (3.10.11)

File Edit Format Run Options Window Help

```
import tensorflow as tf
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Učitavanje Iris skupa podataka
iris = load_iris()
X = iris.data
y = iris.target

# Podjela na trening i test skupove
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Preprocesiranje ciljne varijable (etikete)
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)

# Izgradnja modela
model = Sequential()
model.add(Dense(10, input_dim=4, activation='relu'))  # Ulazni sloj sa 4 ulaza i 10 neurona
model.add(Dense(3, activation='softmax'))  # Izlazni sloj sa 3 klase (iris vrste)

# Kompilacija modela
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Treniranje modela
model.fit(X_train, y_train, epochs=50, batch_size=10)

# Evaluacija modela
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test accuracy: {accuracy}")
```

IDLE Shell 3.10.11

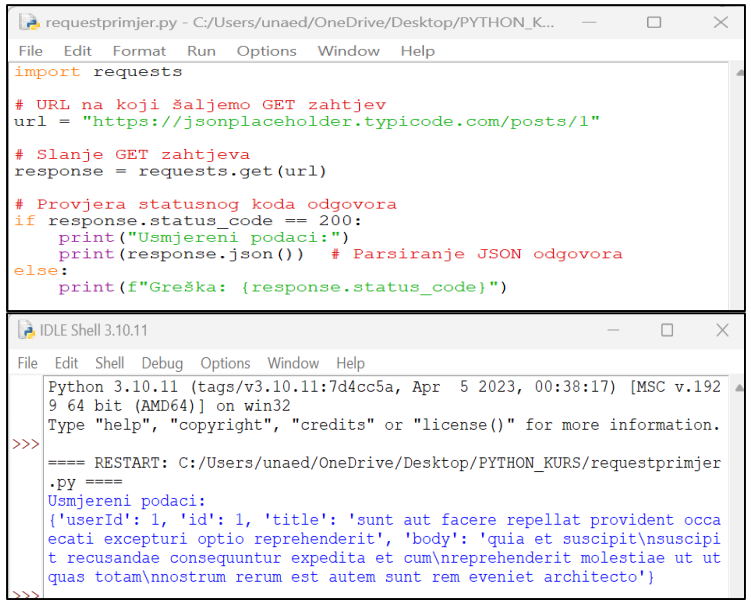
File Edit Shell Debug Options Window Help

```
[1m12/12[0m [32m
[0m[37m[0m [1m0s[0m 2ms/step - accuracy: 0.8986 - loss: 0.5300
Epoch 50/50
[1m 1/12[0m [32m  [0m[37m [0m [1m0s[0m [0m 1
9ms/step - accuracy: 0.9000 - loss: 0.5795
[1m12/12[0m [32m
[0m[37m[0m [1m0s[0m 2ms/step - accuracy: 0.8591 - loss: 0.5272
[1m1/1[0m [32m [0m[37m[0m [1m0s[0m [0m 80m
s/step - accuracy: 0.9333 - loss: 0.5408
[1m1/1[0m [32m
[0m[37m[0m [1m0s[0m 96ms/step - accuracy: 0.9333 - loss: 0.5408
Test accuracy: 0.9333333373069763
```

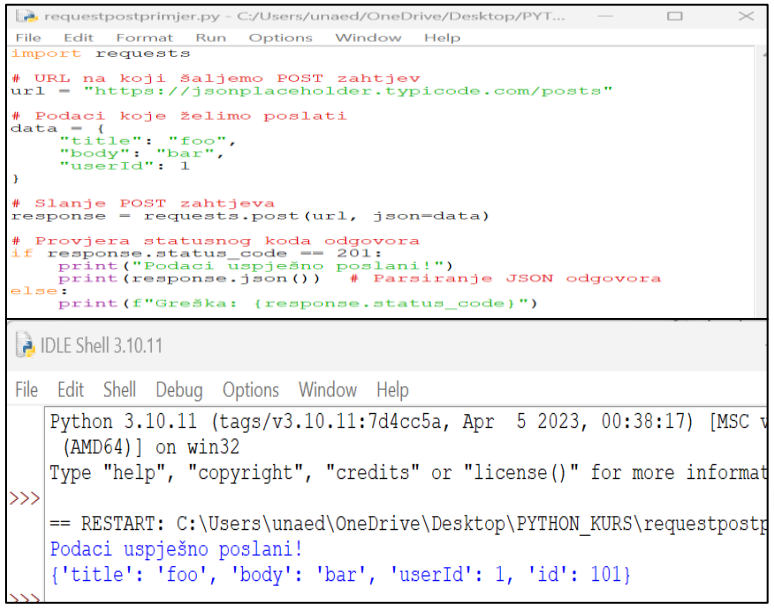
Slika 8. *Primjer upotrebe* Tensorflow *paketa*

# 2. Osnovni pojmovi

- *Requests* – *HTTP* paket za zahtjeve, koji omogućava slanje GET i POST zahtjeva, autentifikaciju i rad sa API-jima, preuzimanje podataka u JSON formatu i druge. Način instalacije paketa je korištenje naredbe: pip install requests, a primjer upotrebe je prikazan na slikama 9. i 10.



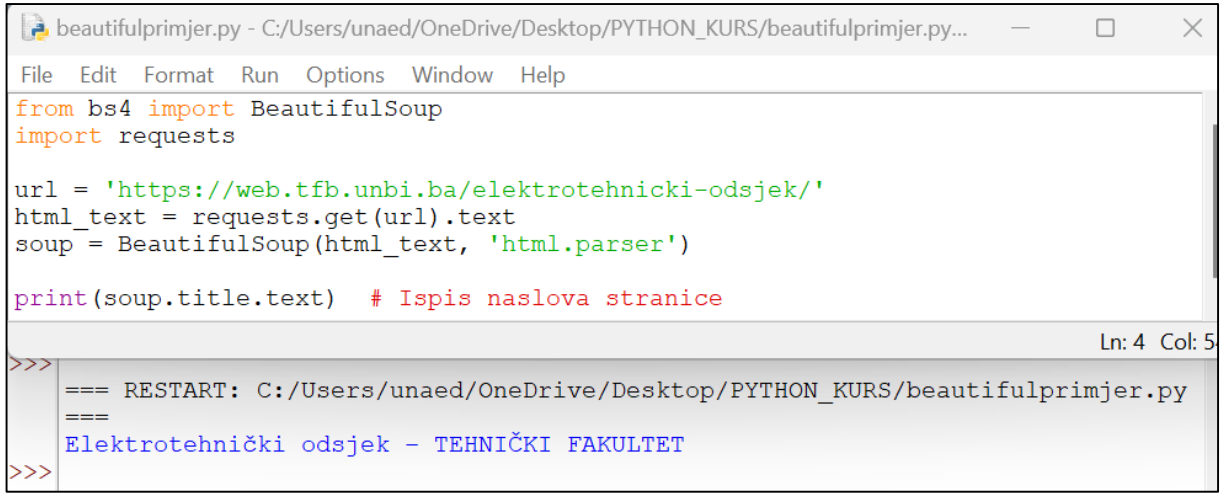

Slika 9. *Primjer upotrebe* request *paketa – GET zahtjev*




Slika 10. *Primjer upotrebe* request *paketa – POST zahtjev*

# 2. Osnovni pojmovi

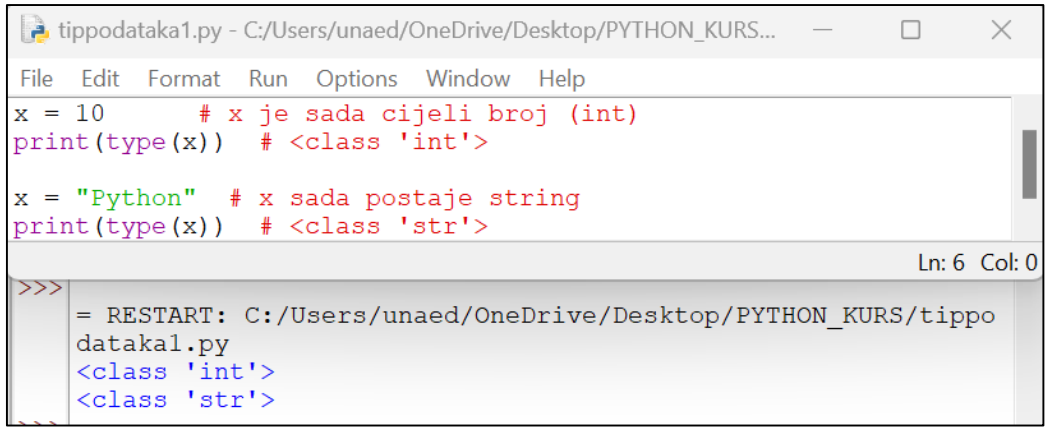
- *BeautifulSoup* je paket koji omogućava parsiranje i ekstrakciju podataka iz HTML i XML dokumenata. Način instalacije paketa je korištenje naredbe: pip install beautifulsoup4, a primjer upotrebe je prikazan na slici 11.




Slika 11. *Primjer upotrebe beautifulsoup4 paketa*

# 3. Tipovi podataka

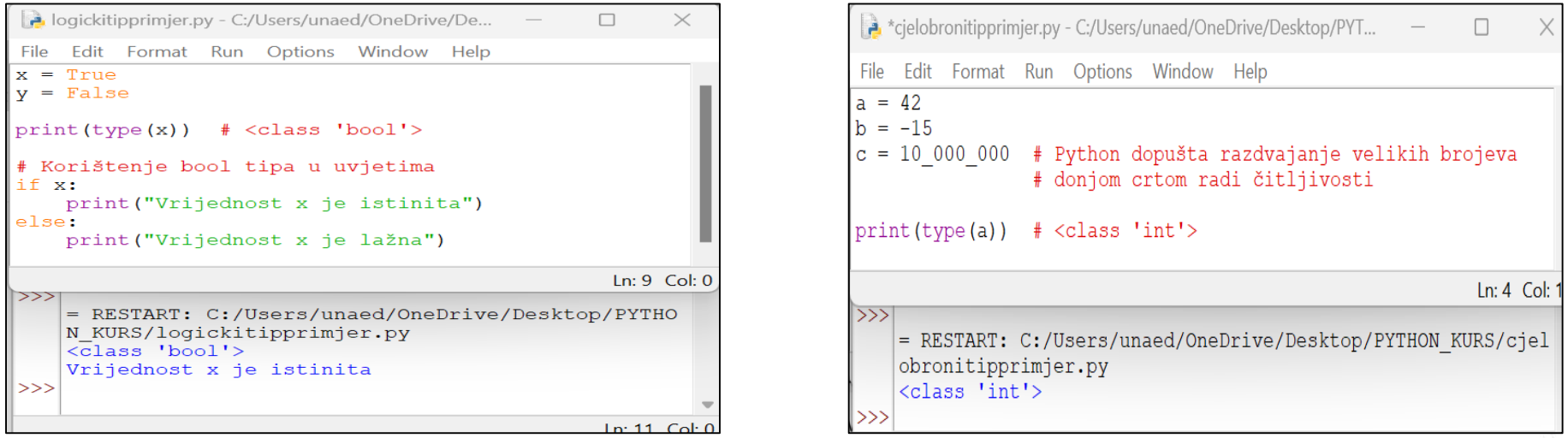
- Python pripada skupini dinamički tipiziranih programskih jezika, što znači da tip varijable nije fiksiran prilikom deklaracije.
- U Pythonu varijabla može u bilo kojem trenutku sadržavati vrijednost bilo kojeg tipa, a sam interpreter automatski određuje tip varijable na temelju dodijeljene vrijednosti.




Slika 12. *Primjer konverzije tipa podatka*

# 3. Tipovi podataka

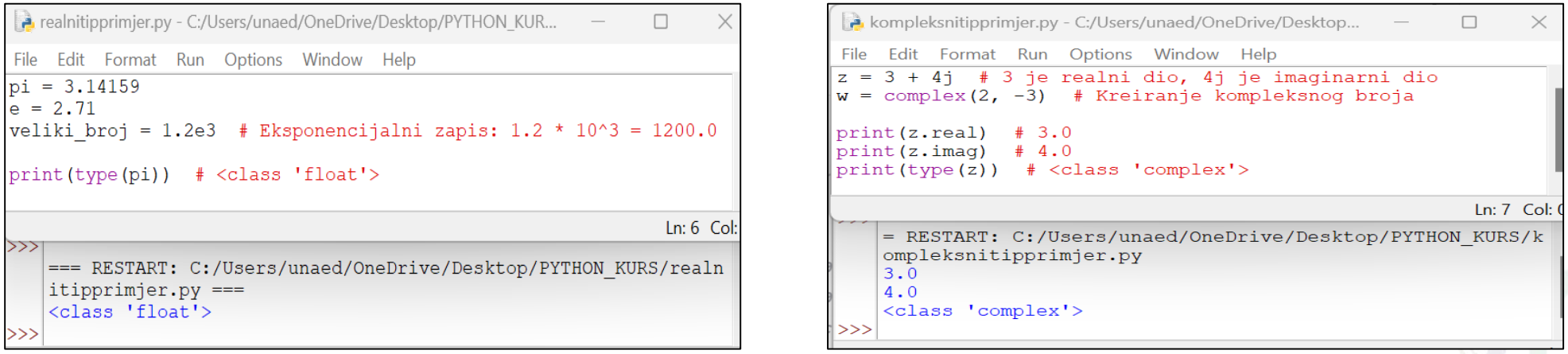
- *Logički tip* (bool) koristi se za predstavljanje istinitosnih vrijednosti, koje mogu biti True (istina) ili False (laž).
- *Cjelobrojni tip* (int) koristi se za pohranu cijelih brojeva.




Slika 13. *Primjer logičkog i cjelobrojnog tipa podataka*

# 3. Tipovi podataka

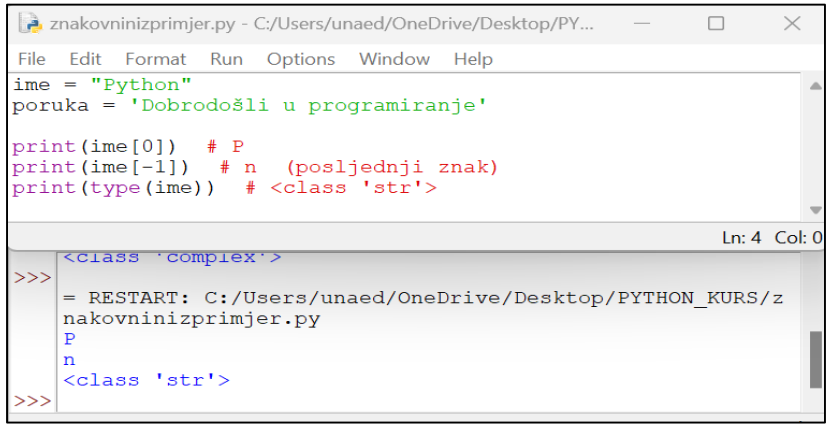
- *Tip float* koristi se za pohranu realnih brojeva s decimalnim dijelom, uključujući i eksponencijalni zapis. Python omogućava automatsku konverziju između int i float kada je to potrebno.
- *Kompleksni tip* podatka, koji se sastoje od realnog i imaginarnog dijela.



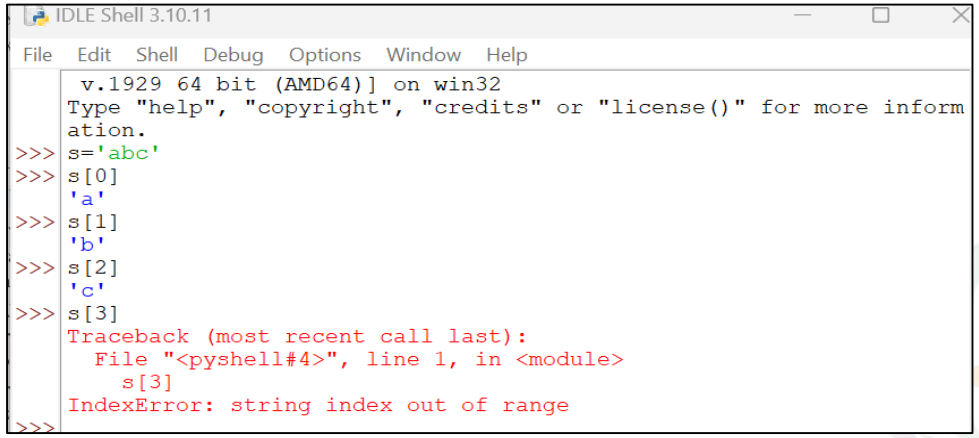

Slika 14. *Primjer realnog i kompleksnog tipa podatka*

# 3. Tipovi podataka

- *Tip str* koristi se za pohranu tekstualnih podataka (nizova znakova). Može se definirati jednostrukim ('...') ili dvostrukim ("...") navodnicima.
- S obzirom da je znakovni niz uređeni niz znakova, njegovim znakovima možemo pristupiti preko indeksa, počevši od nule. Znakovni nizovi su nepromjenjivi (immutable), što znači da ih nije moguće mijenjati nakon **deklarisanja**, već je potrebno stvoriti novi niz.



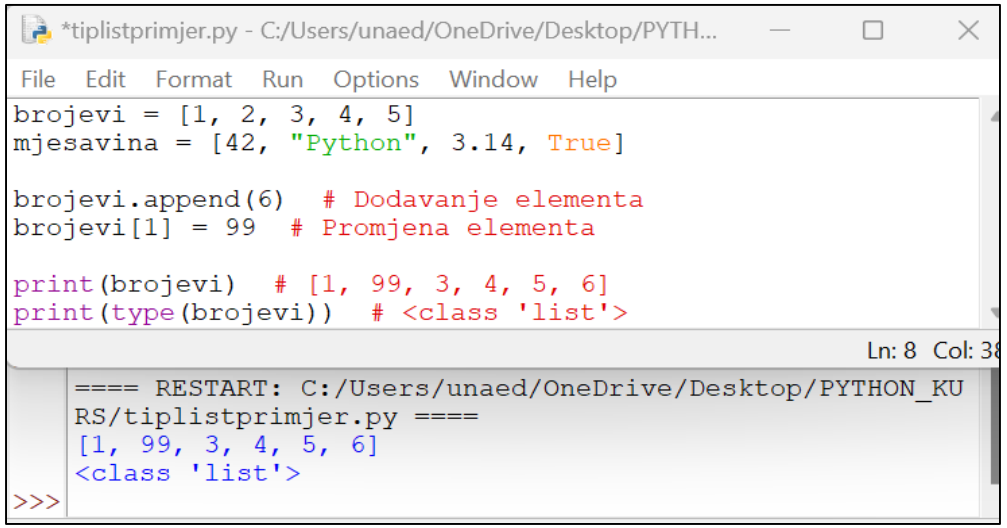

Slika 15. *Primjer znakovnog tipa podatka*



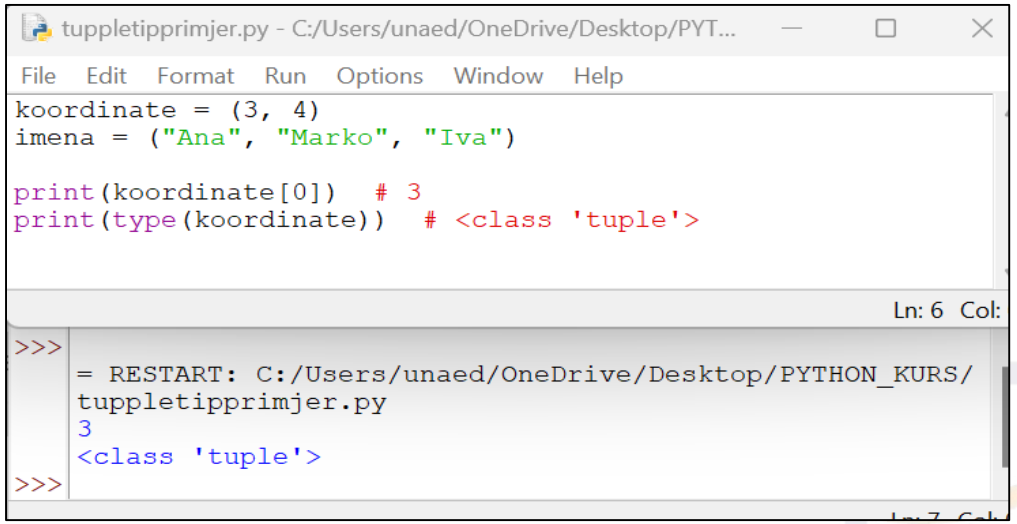

Slika 16. *Primjer pristupanja znakovima preko indeksa*

# 3. Tipovi podataka

- *Lista (list)* je promjenjive (mutable) kolekcija elemenata, koja može da sadrži različite tipove podataka.
- *Torke* su slične listama, ali su nepromjenjive (immutable), što znači da nakon stvaranja ne mogu biti izmijenjene.



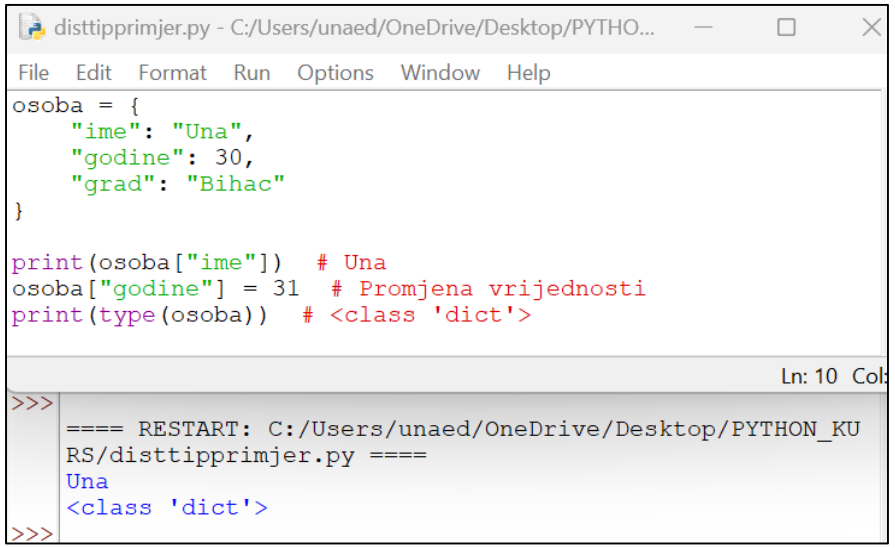

Slika 17. *Primjer list tipa podatka*



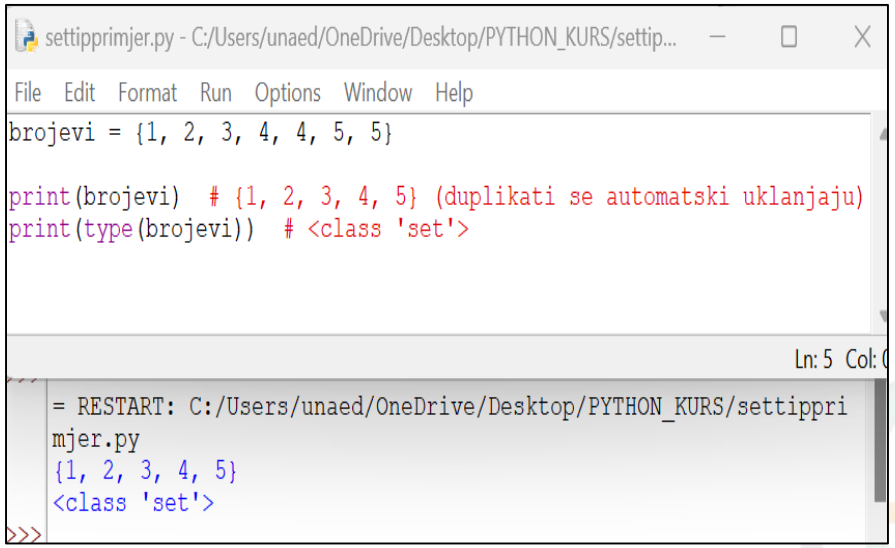

Slika 18. *Primjer tuple tipa podatka*

# 3. Tipovi podataka

- *Rječnik (dictionary)* je kolekcija ključeva i vrijednosti, gdje svaki ključ mora biti jedinstven.
- *Skup (set)* je kolekcija jedinstvenih elemenata, što znači da ne može sadržavati duplikate.



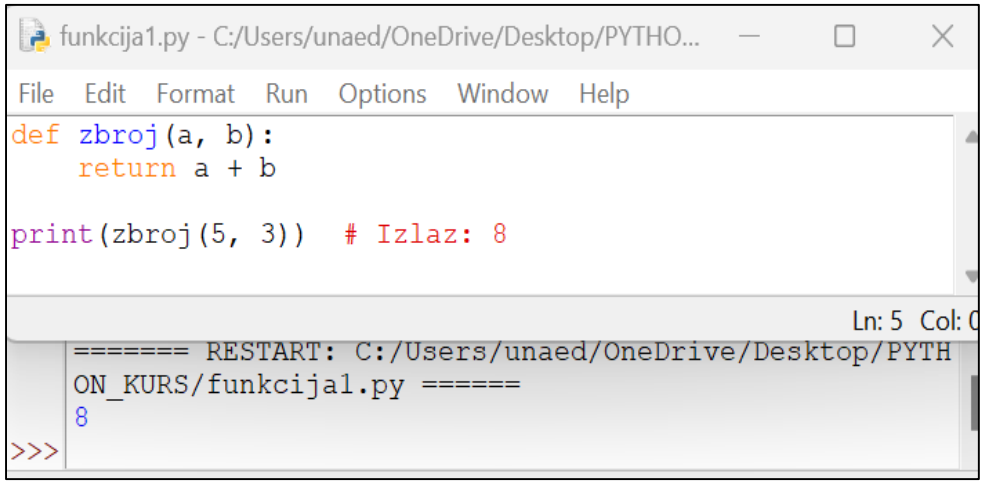

Slika 19. *Primjer dic tip podatka*



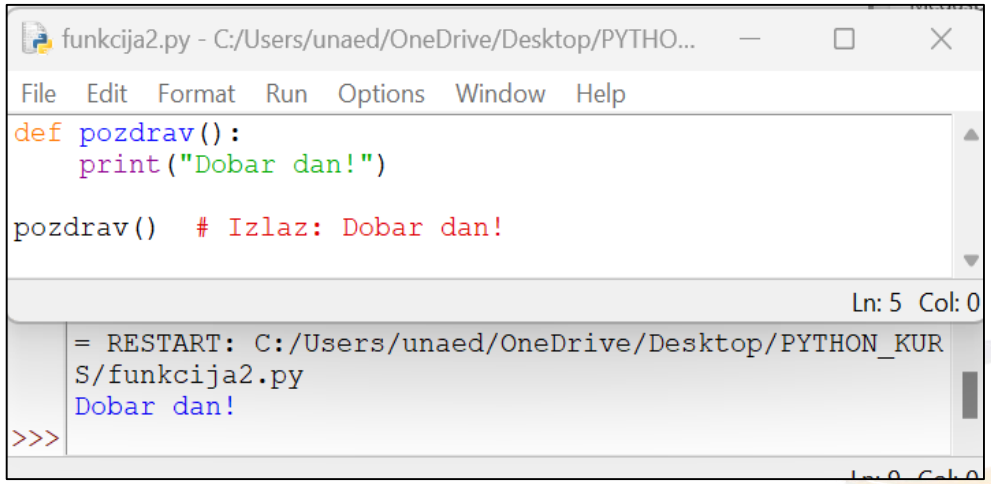

Slika 20. *Primjer set tip podatka*

# 4. Funkcije

- *Standardne (definirane) funkcije* - ove funkcije definiramo pomoću ključne riječi def.
- *Funkcije bez parametara* – funkcija koja ne prima parametre, može se jednostavno pozvati po imenu.



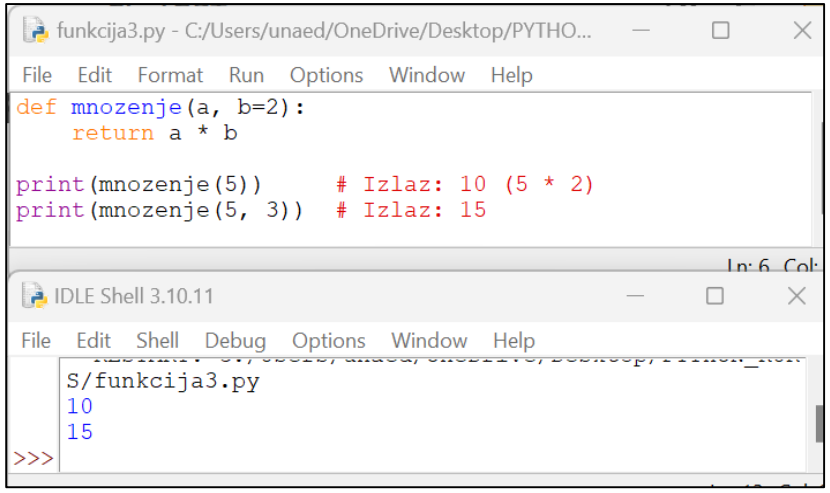

Slika 21. *Primjer standardne funkcije sa def*



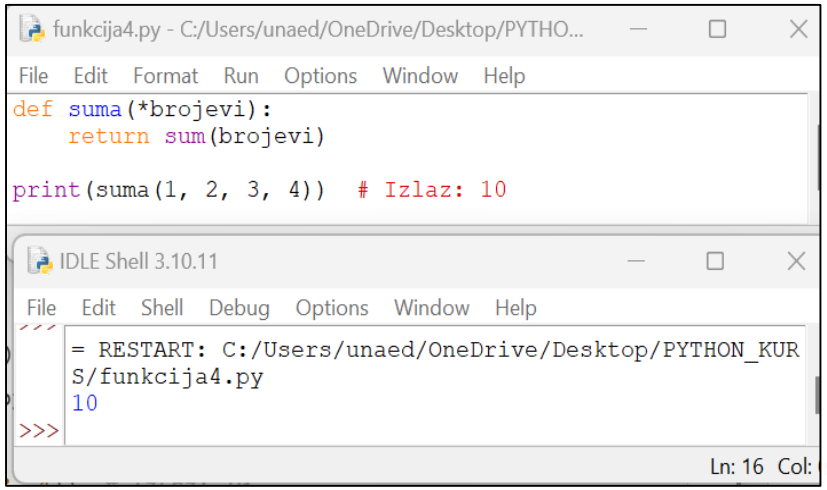

Slika 22. *Primjer funkcije bez parametra*

# 4. Funkcije

- *Funkcije sa podrazumijevanim vrijednostima parametara* - ako se argument ne proslijedi, koristi se podrazumijevana vrijednost.
- *Funkcije sa proizvoljnim brojem argumenata (*args)* - omogućuje prosljeđivanje više argumenata koji se tretiraju kao tuple.



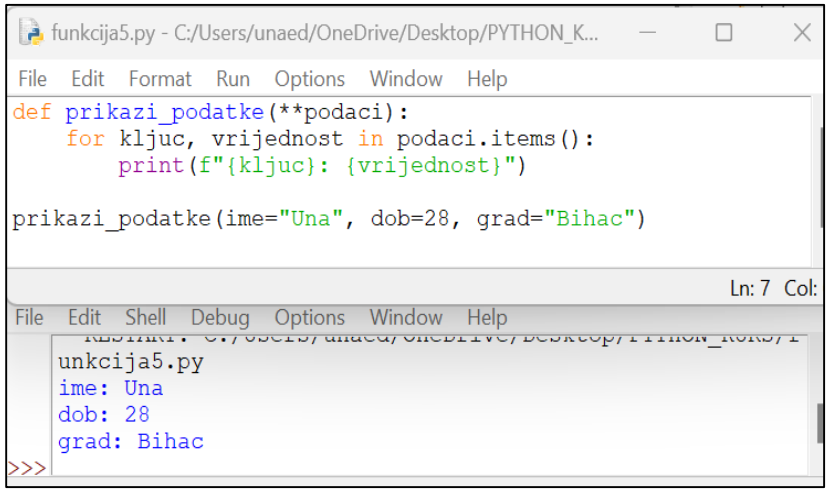

Slika 23. *Funkcije sa podrazumijevanim vrijednostima parametara*



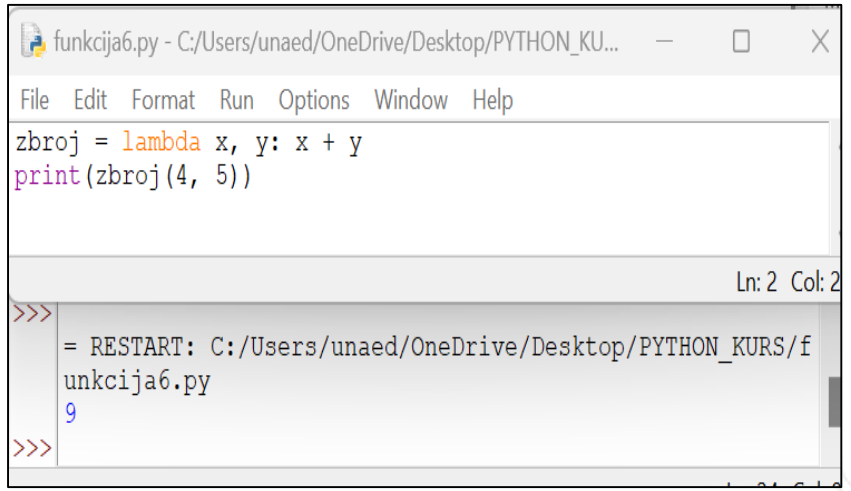

Slika 24. *Funkcije sa proizvoljnim brojem argumenata (*args)*

# 4. Funkcije

- *Funkcije sa proizvoljnim imenovanim argumentima (**kwargs)* - omogućuje prosljeđivanje imenovanih argumenata kao rječnik (dict).
- *Lambda (anonimne) funkcije* - kompaktne funkcije koje se definiraju pomoću lambda izraza.



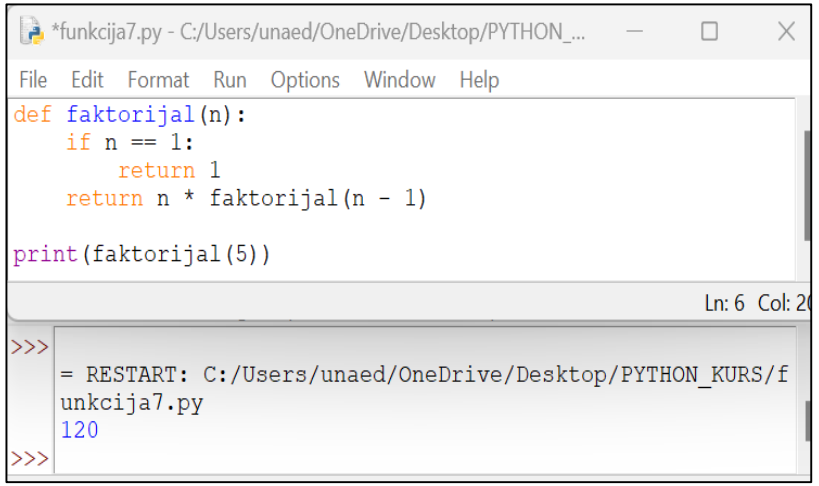

Slika 25. *Funkcije sa sa proizvoljnim imenovanim argumentima (**kwargs)*



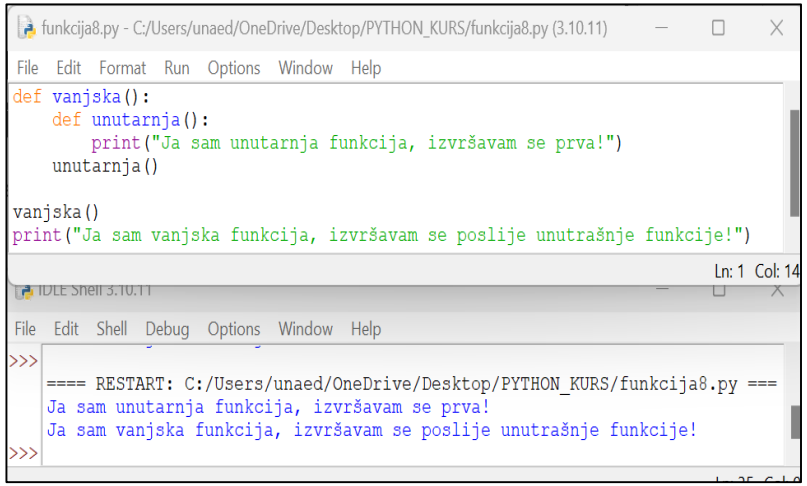

Slika 26. *Lambda (anonimne) funkcije*

# 4. Funkcije

- *Rekurzivne funkcije* - funkcije koje pozivaju same sebe
- *Funkcije unutar funkcija (ugniježdene funkcije) - funkcija definirana unutar druge funkcije.*




Slika 27. *Rekurzivna funkcija*




Slika 28. *Funkcija unutar funkcije*

# 4. Funkcije

ugnijezdenafunkcijaKalkulator.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KU...

File Edit Format Run Options Window Help

```
def kalkulator(operacija):
    """Glavna funkcija koja vraća odgovarajuću matematičku operaciju."""

    def zbroj(a, b):
        return a + b

    def oduzmi(a, b):
        return a - b

    def pomnozi(a, b):
        return a * b

    def podijeli(a, b):
        if b == 0:
            return "Greška: Dijeljenje s nulom nije dozvoljeno."
        return a / b

    # Odabir i vraćanje odgovarajuće funkcije
    if operacija == "zbroj":
        return zbroj
    elif operacija == "oduzmi":
        return oduzmi
    elif operacija == "pomnozi":
        return pomnozi
    elif operacija == "podijeli":
        return podijeli
    else:
        return None  # Nevažeća operacija

# Korisnik bira operaciju
operacija = kalkulator("zbroj")  # Vraća funkciju za zbrajanje
if operacija:
    rezultat = operacija(10, 5)
    print("Rezultat:", rezultat)  # Izlaz: Rezultat: 15

operacija = kalkulator("podijeli")
if operacija:
    rezultat = operacija(20, 4)
    print("Rezultat:", rezultat)  # Izlaz: Rezultat: 5.0
```

IDLE Shell 3.10.11

File Edit Shell Debug Options Window Help

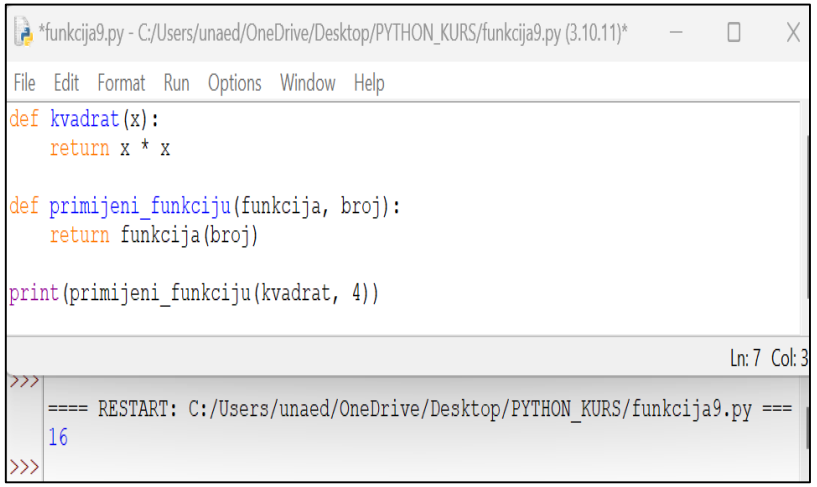
```
>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS
/ugnijezdenafunkcijaKalkulator.py
Rezultat: 15
Rezultat: 5.0
>>>
```

Ln: 23 Col: 0

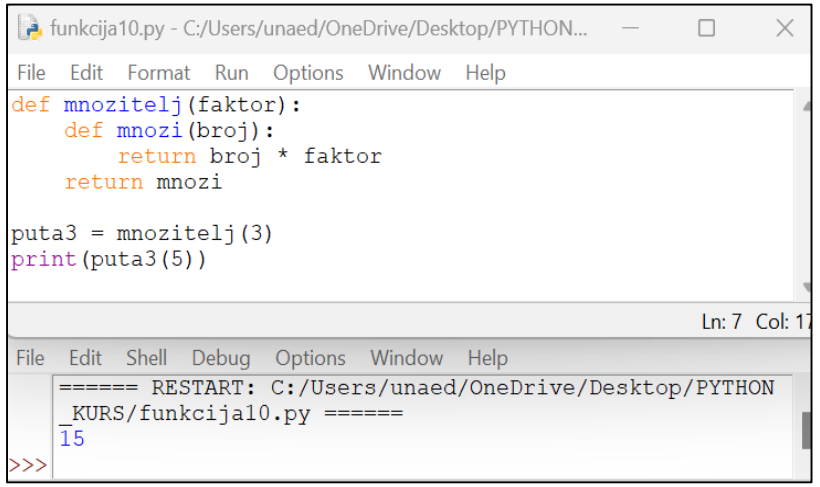
Slika 29. *Funkcija unutar funkcije*

# 4. Funkcije

- *Funkcije kao argumenti drugih funkcija* - funkcije, koje se mogu prosljeđivati kao argumenti drugim funkcijama.
- *Funkcije koje vraćaju druge funkcije*.




Slika 30. *Funkcija kao argument druge funkcije*




Slika 31. *Funkcija koja vraća drugu funkciju*

# 4. Funkcije

*funkcijakojavracafunkcijuprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/f...

File Edit Format Run Options Window Help

```
def eksponenta_funkcija(exponent):
    """Funkcija koja vraća funkciju za izračunavanje broja na eksponent."""

    def izracunaj_exponent(x):
        return x ** exponent

    return izracunaj_exponent

# Kreiraj funkciju koja računa kvadrat (eksponent = 2)
kvadrat = eksponenta_funkcija(2)

# Kreiraj funkciju koja računa kub (eksponent = 3)
kub = eksponenta_funkcija(3)

# Ispis rezultata
print(kvadrat(4))
print(kub(4))
```

IDLE Shell 3.10.11

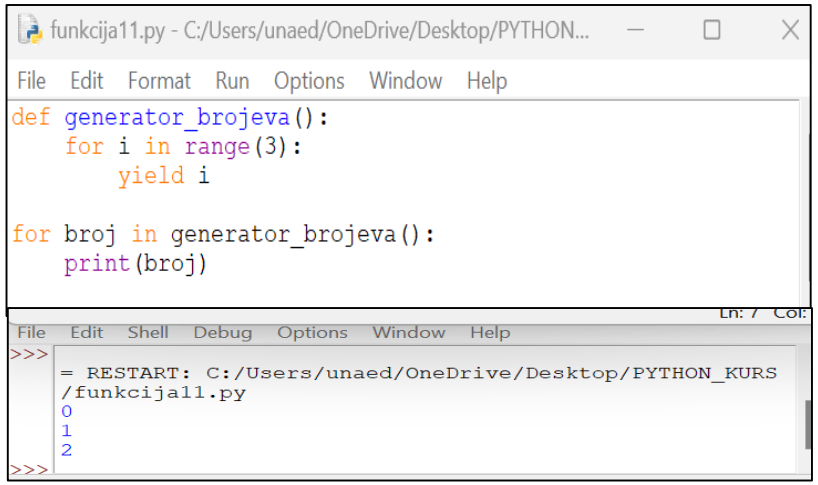
File Edit Shell Debug Options Window Help

```
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MS
C v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more infor
mation.
>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcijako
javracafunkcijuprimjer.py
16
64
>>>
```

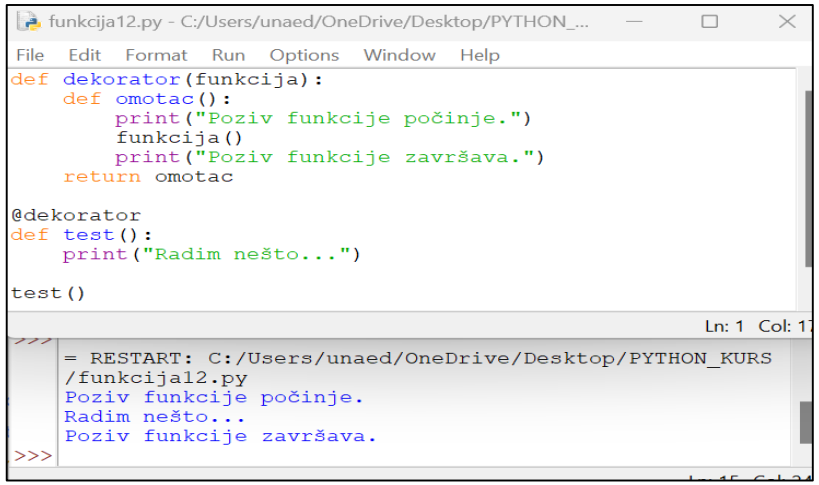
Slika 32. *Funkcija koja vraća drugu funkciju*

# 4. Funkcije

- *Generator funkcije (yield)* - umjesto return, koriste yield za stvaranje iteratora.
- *Funkcije dekoratori* - funkcije koje mijenjaju ponašanje drugih funkcija.



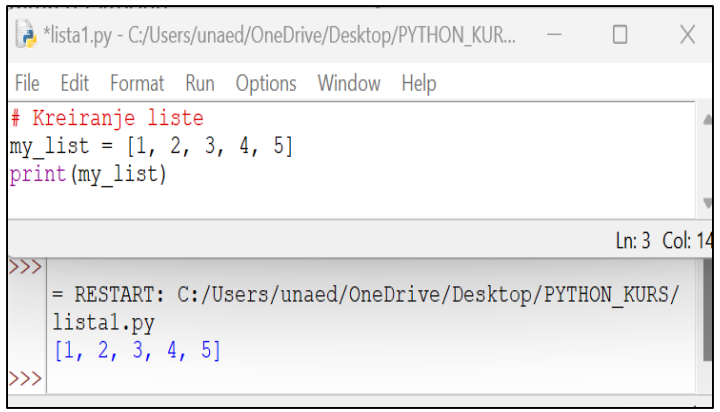

Slika 33. *Generator funkcija*

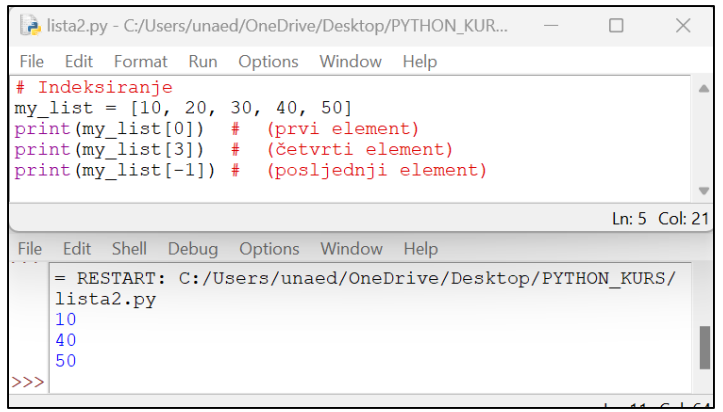


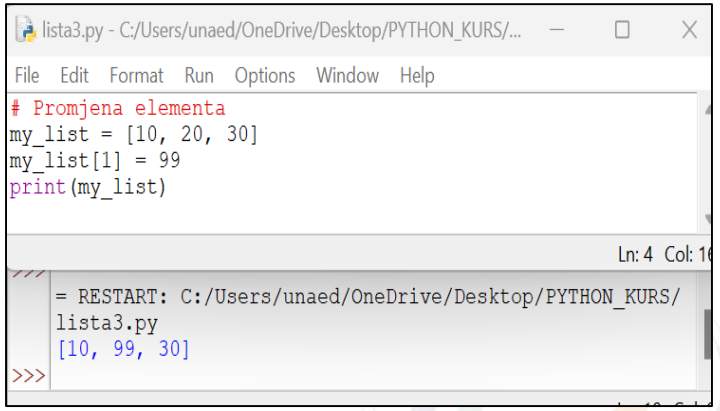

Slika 34. *Funkcija dekorator*

# 5. Liste

- Lista se može kreirati pomoću uglatih zagrada [ ], a elementi unutar liste su odvojeni zarezom.
- Indeksiranje elemenata unutar liste počinje od 0.
- Elementi unutar liste se mogu mijenjati.



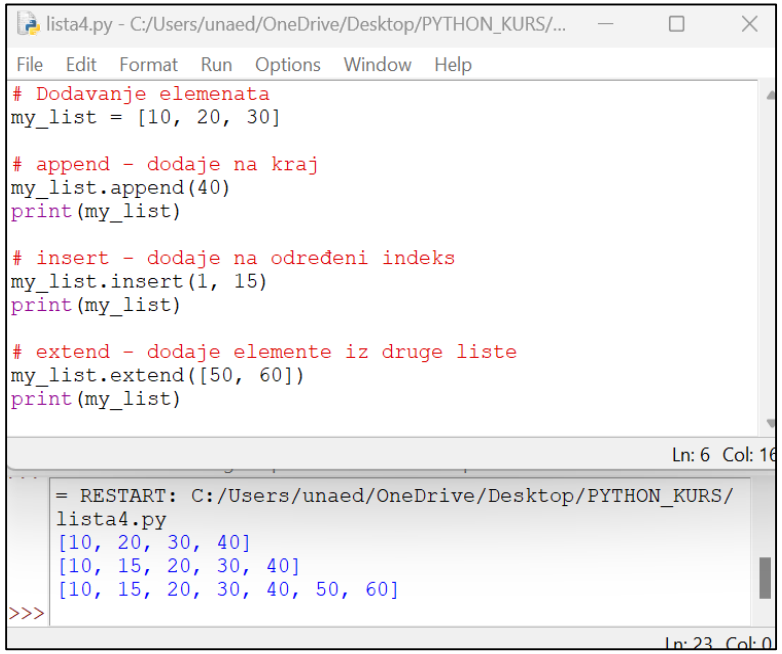



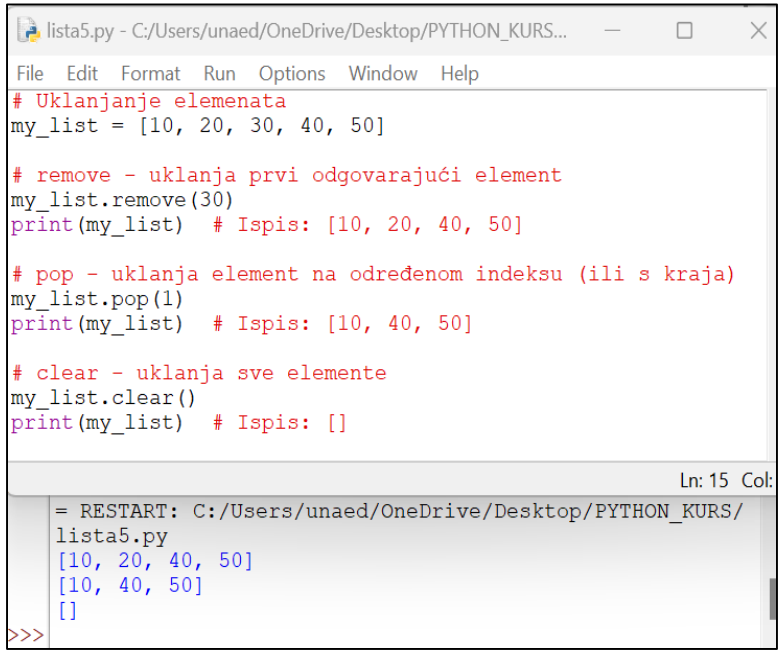




Slika 35. *Kreiranje liste; Indeksiranje liste; Promjena elementa u listi*

# 5. Liste

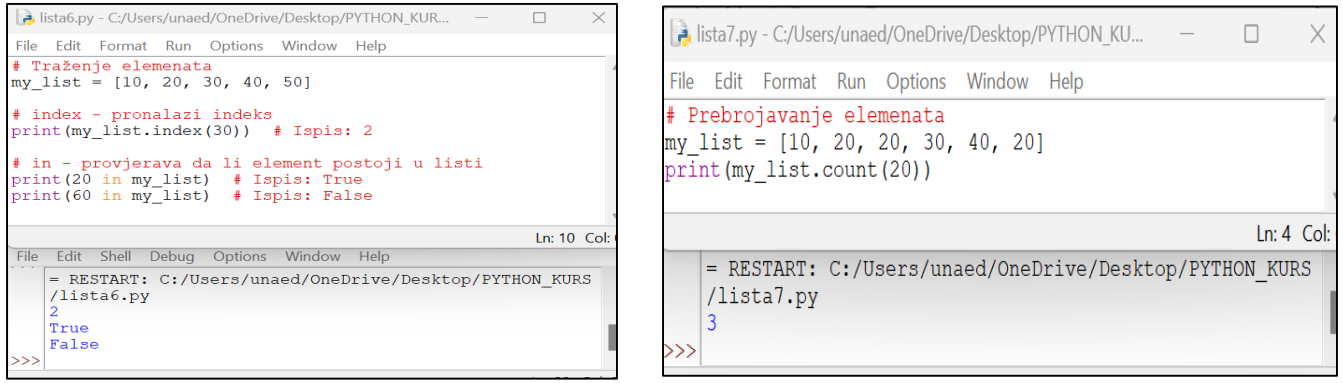
- Nove elemente možemo dodavati u listu, a postojeće elemente možemo uklanjati, tražiti i prebrojavati.
- Liste možemo sortirati, kopirati i spajati.



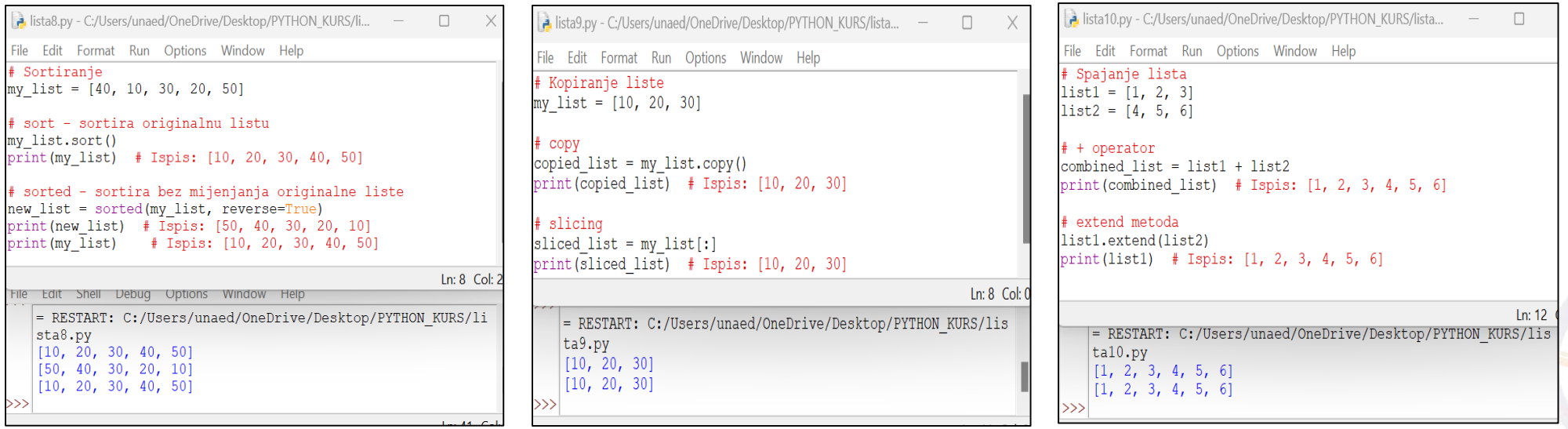




Slika 36. *Dodavanje i uklanjanje elemenata u listama*

# 5. Liste



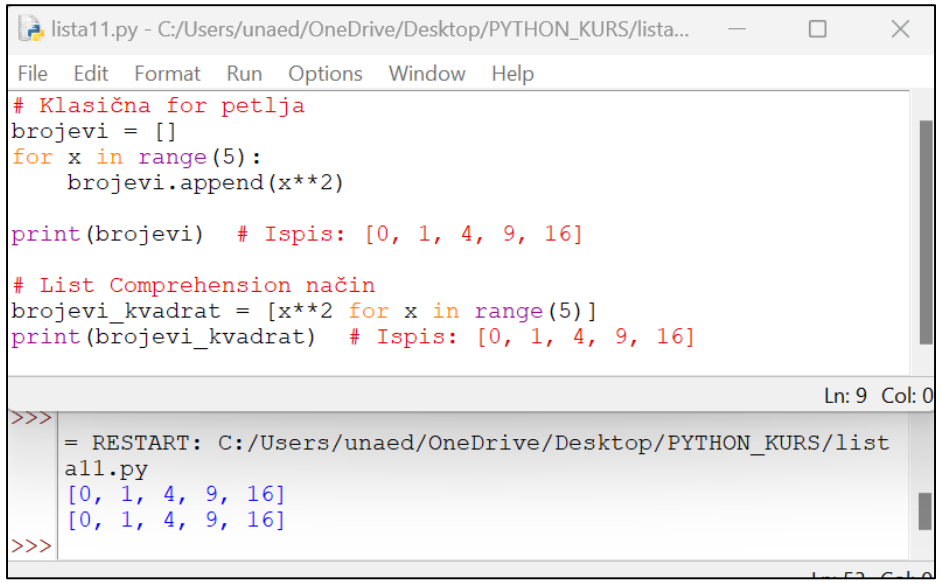

Slika 37. *Traženje i prebrojavanje elemenata u listama*



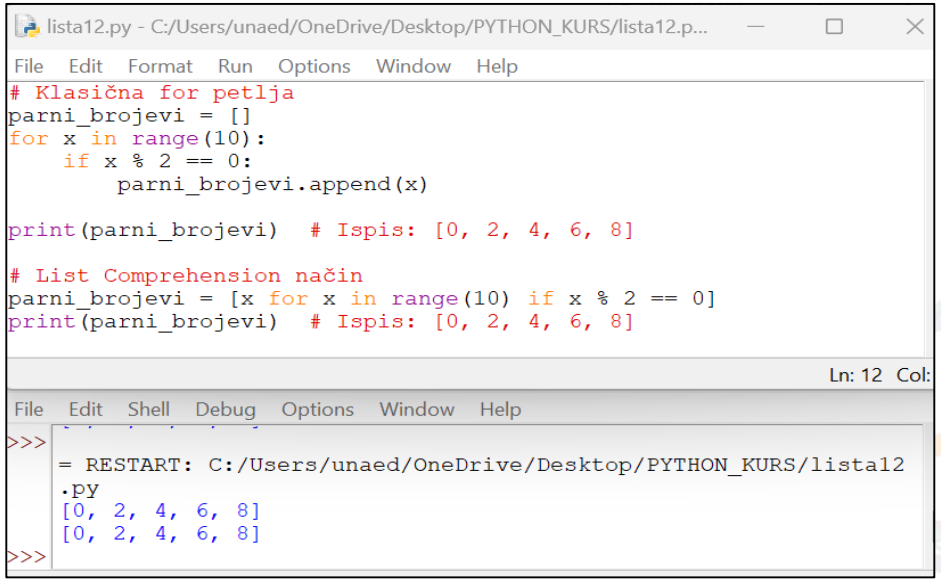

Slika 38. *Sortiranje, kopiranje i spajanje lista*

# 5. Liste

- *List Comprehension* (ili generiranje listi) je način za stvaranje novih lista koristeći kraću i efikasniju sintaksu u odnosu na klasične for petlje.
- Koristi se za: generiranje novih listi pomoću for petlje, filtriranje podataka pomoću if uvjeta, transformaciju elemenata pomoću izraza unutar liste.

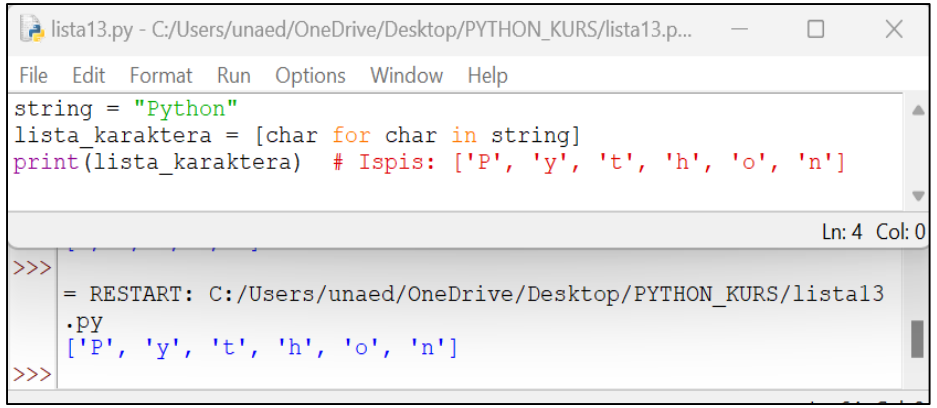



Slika 39. *Primjer generiranja liste brojeva*

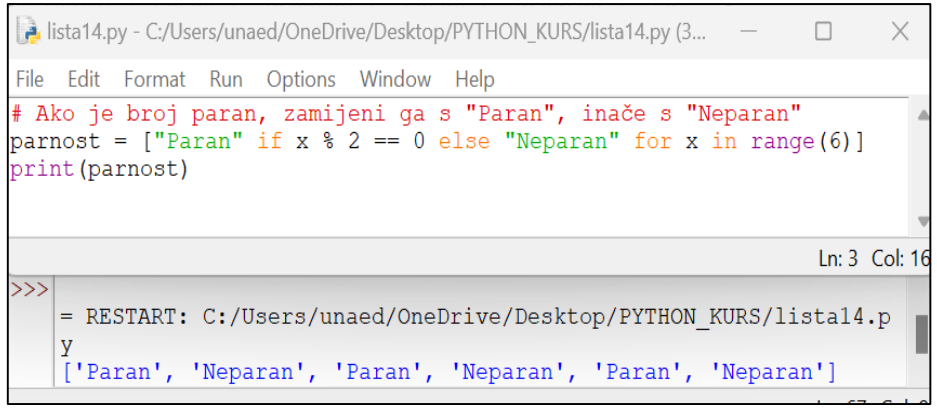



Slika 40. *Primjer filtriranja liste brojeva*

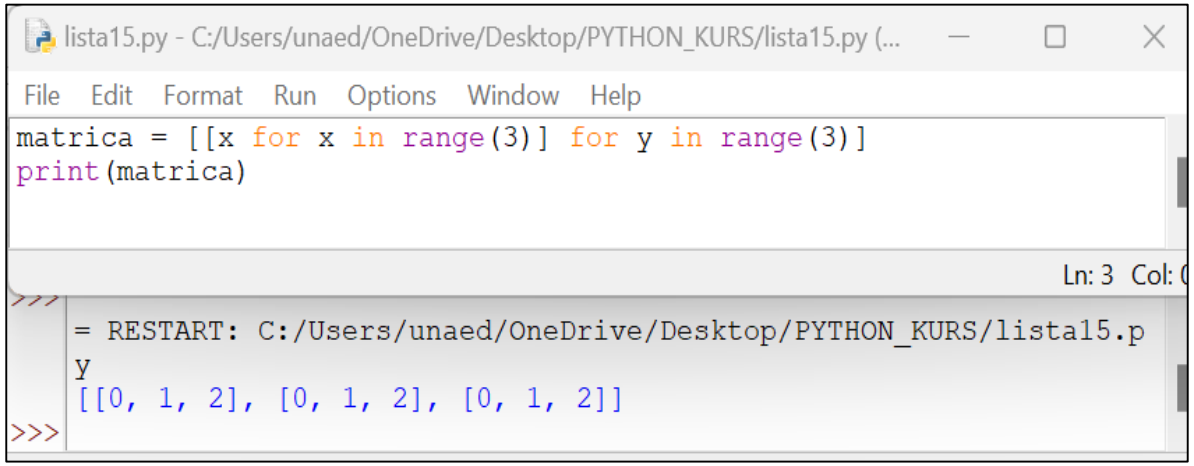
# 5. Liste




Slika 41. *Primjer konverzije stringa u listu karaktera*



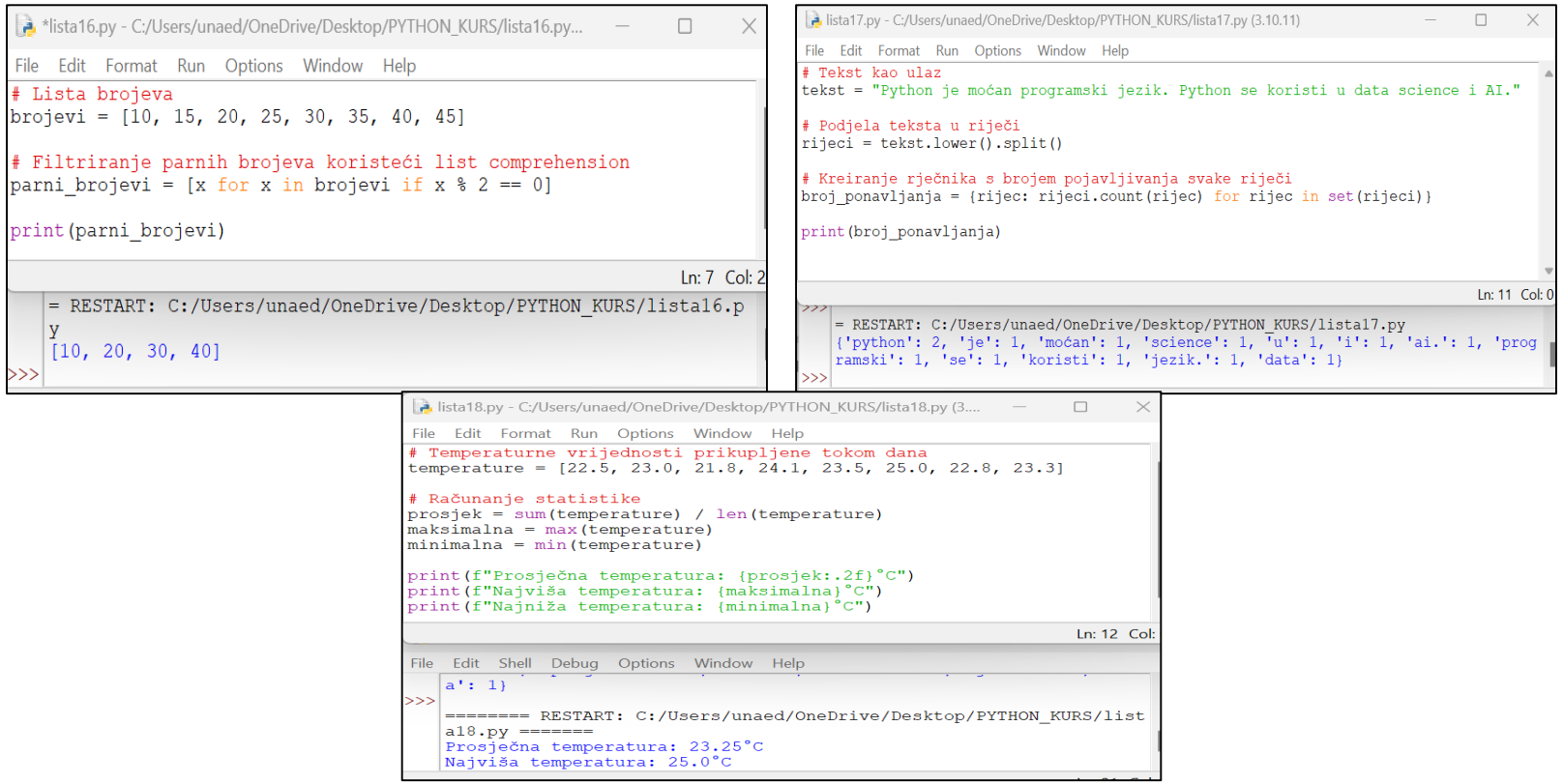

Slika 42. *Primjer zamjene if – else u List Comprehension*




Slika 43. *Primjer za Ugniježdeni List Comprehension*

# 5. Liste

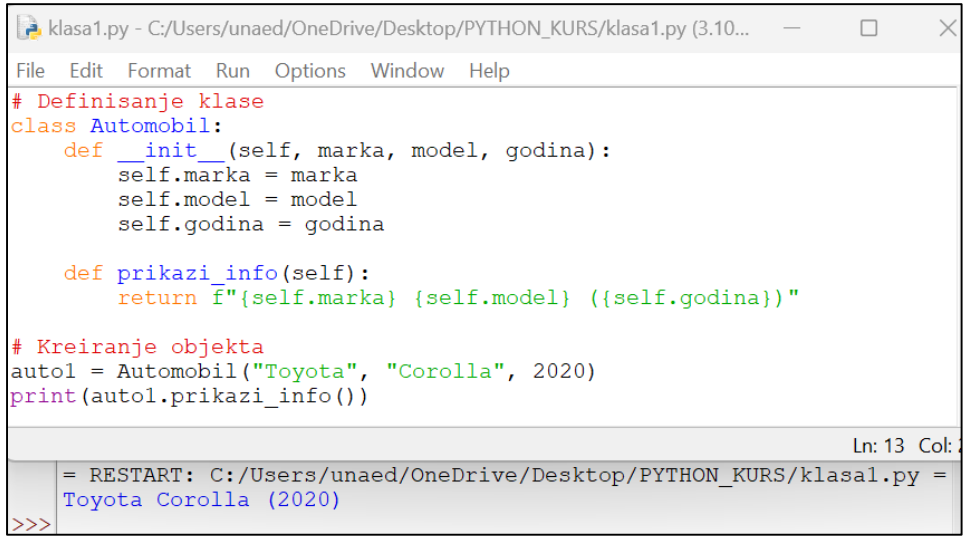
- Liste su vrlo korisne u Pythonu i koriste se u različitim situacijama, od obrade podataka do rada s bazama podataka i analizom istih.




Slika 44. *Filtriranje brojeva; Rad sa tekstom u listi, Analiza podataka*

# 6. Klase

- Klase su temelj objektno orijentisanog programiranja (OOP) u Pythonu. Omogućavaju organizaciju koda u objekte koji sadrže podatke (atribute) i metode (funkcije koje rade s tim podacima).
- Klasa se definiše pomoću ključne riječi *class*, a objekti se kreiraju iz nje.

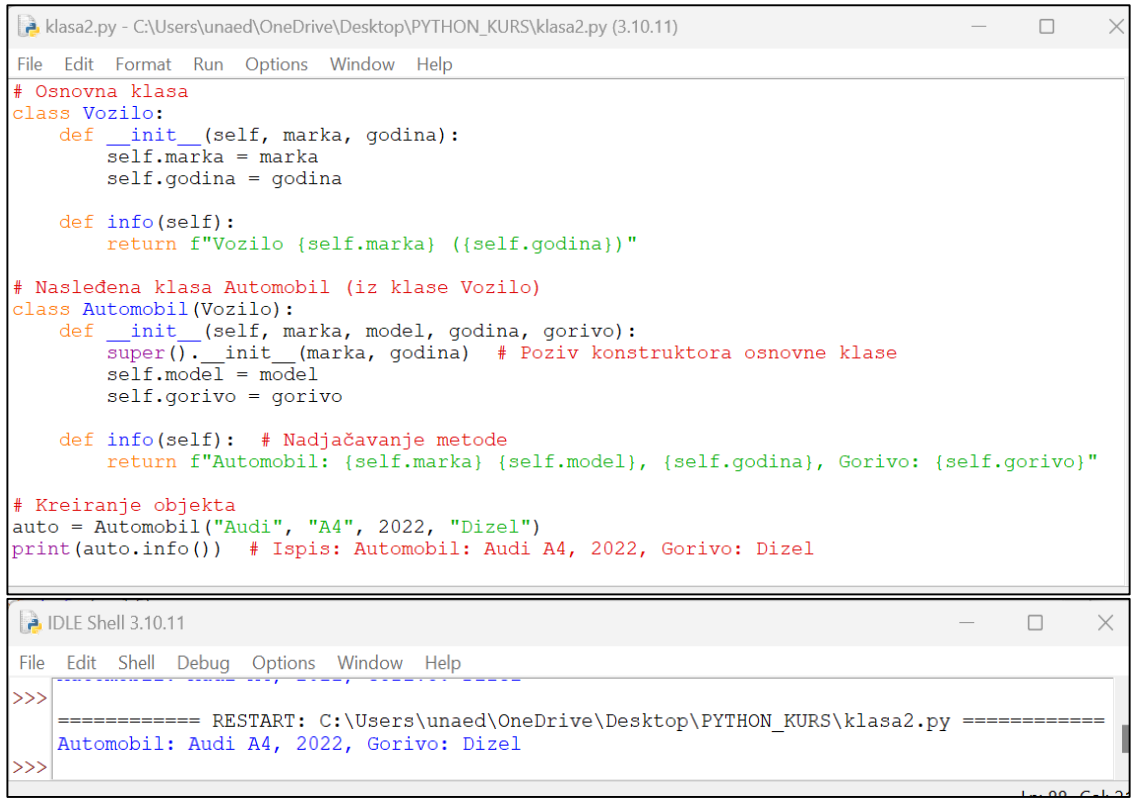



Slika 45. *Primjer kreiranja objekta*

- __init__ je konstruktor koji inicijalizuje atribute prilikom kreiranja objekta;
- self označava referencu na instancu klase;
- prikazi_info() vraća string sa podacima o automobilu.

# 6. Klase

- Nasljeđivanje (Inheritance) omogućava stvaranje nove klase koja preuzima osobine postojeće klase, čime se izbjegava dupliranje koda.

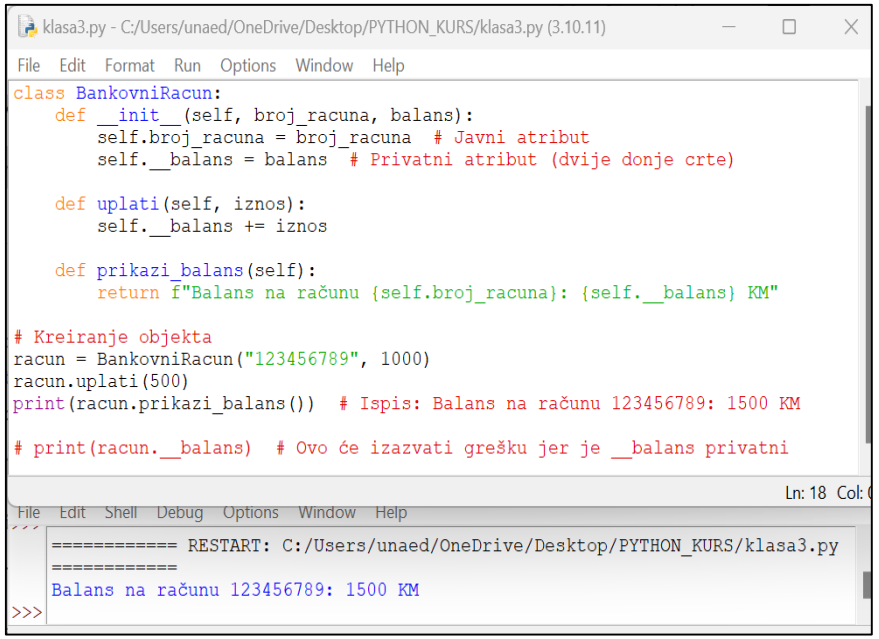



Slika 46. *Primjer nasljeđivanja*

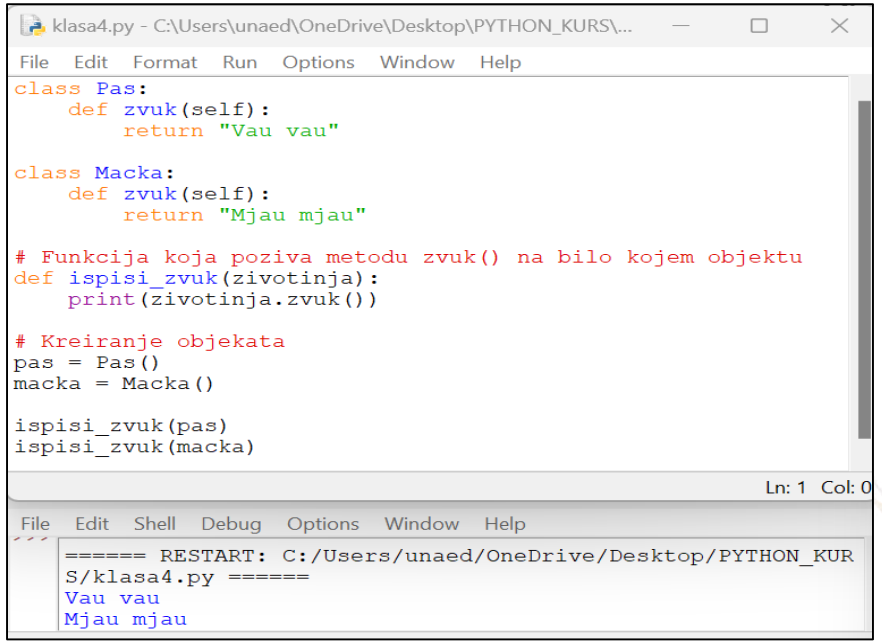
- Klasa Automobil nasljeđuje klasu Vozilo, što znači da preuzima njene metode i atribute;
- super().__init__(marka, godina) poziva konstruktor osnovne klase;
- info() metoda je nadjačana i sada daje specifičniji ispis.

# 6. Klase

- Enkapsulacija (Private i Protected atributi) ograničava pristup podacima unutar klase, štiteći ih od vanjske izmjene.
- Polimorfizam (metode s istim imenom u različitim klasama), omogućava korištenje istih metoda u različitim klasama sa različitim ponašanjem.



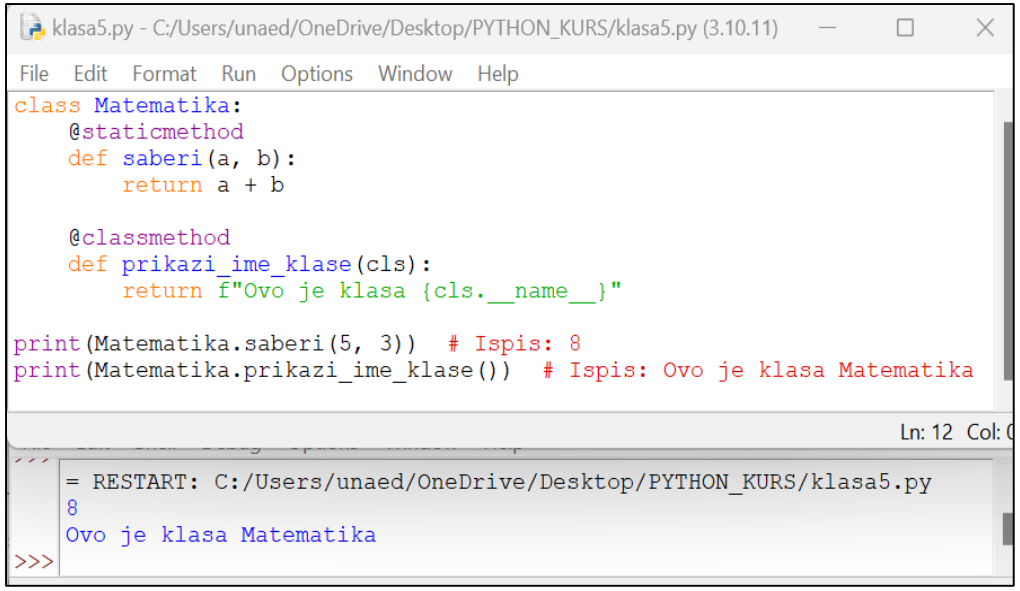

Slika 47. *Primjer enkapsulacije*




Slika 48. *Primjer polimorfizma*

# 6 Klase

- Statičke metode ne koriste self i ne zavise od instance klase.
- Metode klase koriste cls i rade sa svim instancama klase.




Slika 49. *Primjer statičke metode i metode klase*

- @staticmethod označava metodu koja ne zavisi od instance (self);
- @classmethod koristi cls, što omogućava rad s klasom, a ne samo instancom.

# 6. Klase

- Klase se koriste u raznim aplikacijama, od web programiranja do vještačke inteligencije i obrade podataka.

KLASA6.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/KLASA6.py...

File Edit Format Run Options Window Help

```
from django.db import models

class Korisnik(models.Model):
    ime = models.CharField(max_length=50)
    email = models.EmailField(unique=True)
    datum_registracije = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.ime} - {self.email}"
```

Slika 50. *Primjer definisanja Django*

klasa7.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa7.py (3...

File Edit Format Run Options Window Help

```
import tensorflow as tf
from tensorflow import keras

class Mreza(keras.Model):
    def __init__(self):
        super(Mreza, self).__init__()
        self.sloj1 = keras.layers.Dense(64, activation='relu')
        self.sloj2 = keras.layers.Dense(10, activation='softmax')

    def call(self, ulaz):
        x = self.sloj1(ulaz)
        return self.sloj2(x)

# Kreiranje modela
model = Mreza()
```

Slika 51. *Primjer definisanja neuronske mreže*

klasa8.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa8.py (3.10.11)

File Edit Format Run Options Window Help

```
import pandas as pd

class AnalizaPodataka:
    def __init__(self, fajl):
        self.podaci = pd.read_csv(fajl)

    def prikazi_info(self):
        return self.podaci.info()

    def prikazi_prvih_n(self, n=5):
        return self.podaci.head(n)

# Kreiranje objekta
analiza = AnalizaPodataka("podaci.csv")
print(analiza.prikazi_info())
print(analiza.prikazi_prvih_n(3))
```

Slika 52. *Primjer obrade podataka*

klasa9.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa9.py (3.10.11)

File Edit Format Run Options Window Help

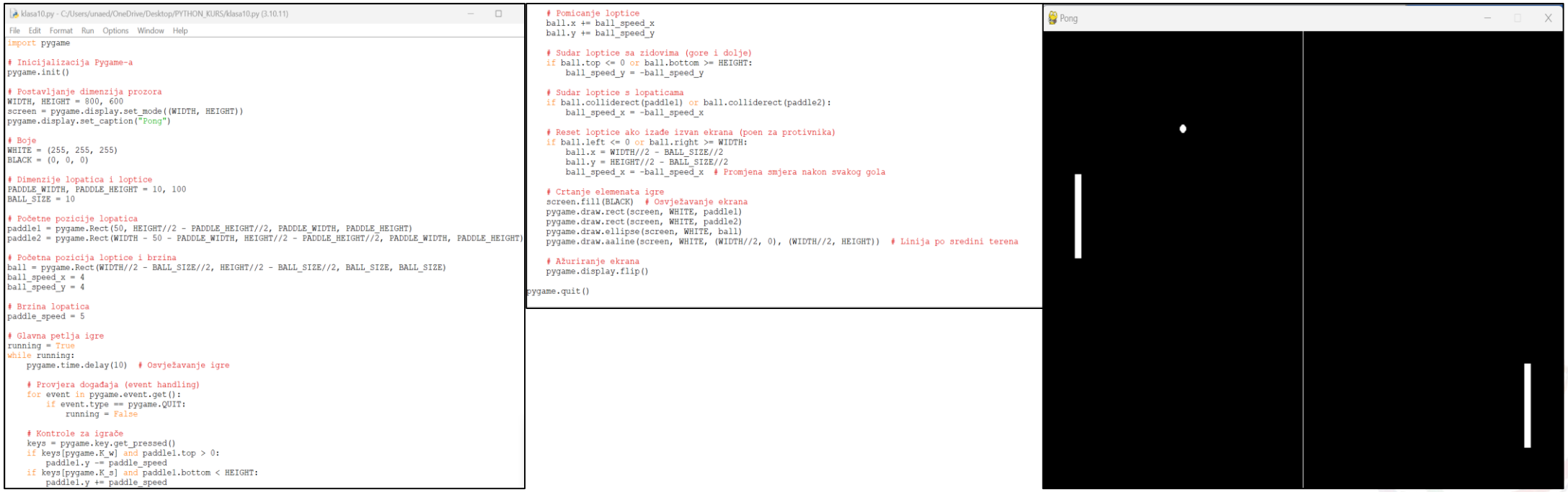
```
import requests
from bs4 import BeautifulSoup

class WebScraper:
    def __init__(self, url):
        self.url = url
        self.sadrzaj = self.preuzmi_stranicu()

    def preuzmi_stranicu(self):
        odgovor = requests.get(self.url)
        return BeautifulSoup(odgovor.text, 'html.parser')

    def pronadji_naslove(self):
        naslovi = self.sadrzaj.find_all('h2')
        return [naslov.text for naslov in naslovi]

# Kreiranje objekta
scraper = WebScraper("https://web.tfb.unbi.ba/elektrotehnicki-odsjek/")
print(scraper.pronadji_naslove())
```

```
Ln: 10 Col
========= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa9.py
=========
['Elektrotehnički odsjek', 'Vijesti', 'O odsjeku', 'Voditelj odsjeka',
'Nastavno osoblje', 'Studijski programi', 'Galerija']
>>>
```

Slika 53. *Primjer preuzimanja podataka*

# 6. Klase

- Klase se koriste za pravljenje video igrica, simulacija i sistema za učenje pomoću pojačanja. Paketi koji se najčešće koriste su Pygame, OpenAI Gym.



```
klasa10.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa10.py (3.10.11)
File  Edit  Format  Run  Options  Window  Help
import pygame

# Inicijalizacija Pygame-a
pygame.init()

# Postavljanje dimenzija prozora
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Pong")

# Boje
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

# Dimenzije lopatica i loptice
PADDLE_WIDTH, PADDLE_HEIGHT = 10, 100
BALL_SIZE = 10

# Početne pozicije lopatica
paddle1 = pygame.Rect(50, HEIGHT//2 - PADDLE_HEIGHT//2, PADDLE_WIDTH, PADDLE_HEIGHT)
paddle2 = pygame.Rect(WIDTH - 50 - PADDLE_WIDTH, HEIGHT//2 - PADDLE_HEIGHT//2, PADDLE_WIDTH, PADDLE_HEIGHT)

# Početna pozicija loptice i brzina
ball = pygame.Rect(WIDTH//2 - BALL_SIZE//2, HEIGHT//2 - BALL_SIZE//2, BALL_SIZE, BALL_SIZE)
ball_speed_x = 4
ball_speed_y = 4

# Brzina lopatica
paddle_speed = 5

# Glavna petlja igre
running = True
while running:
    pygame.time.delay(10)  # Osvježavanje igre

    # Provjera događaja (event handling)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Kontrole za igrače
    keys = pygame.key.get_pressed()
    if keys[pygame.K_w] and paddle1.top > 0:
        paddle1.y -= paddle_speed
    if keys[pygame.K_s] and paddle1.bottom < HEIGHT:
        paddle1.y += paddle_speed
```

```
    # Pomicanje loptice
    ball.x += ball_speed_x
    ball.y += ball_speed_y

    # Sudar loptice sa zidovima (gore i dolje)
    if ball.top <= 0 or ball.bottom >= HEIGHT:
        ball_speed_y = -ball_speed_y

    # Sudar loptice s lopaticama
    if ball.colliderect(paddle1) or ball.colliderect(paddle2):
        ball_speed_x = -ball_speed_x

    # Reset loptice ako izađe izvan ekrana (poen za protivnika)
    if ball.left <= 0 or ball.right >= WIDTH:
        ball.x = WIDTH//2 - BALL_SIZE//2
        ball.y = HEIGHT//2 - BALL_SIZE//2
        ball_speed_x = -ball_speed_x  # Promjena smjera nakon svakog gola

    # Crtanje elemenata igre
    screen.fill(BLACK)  # Osvježavanje ekrana
    pygame.draw.rect(screen, WHITE, paddle1)
    pygame.draw.rect(screen, WHITE, paddle2)
    pygame.draw.ellipse(screen, WHITE, ball)
    pygame.draw.aaline(screen, WHITE, (WIDTH//2, 0), (WIDTH//2, HEIGHT))  # Linija po sredini terena

    # Ažuriranje ekrana
    pygame.display.flip()

pygame.quit()
```




Slika 54. *Primjer kreiranja igrice Pong*

# 6. Klase

klasa11.py - C:\Users\unaed\OneDrive\Desktop\PYTHON_KURS\klasa11.py (3.10.11)

File Edit Format Run Options Window Help

```
import random

def pogodi_broj():
    broj = random.randint(1, 5)  # Računar bira broj
    pokusaji = 0

    while True:
        pokusaj = int(input("Pogodi broj (1-5): "))
        pokusaji += 1

        if pokusaj < broj:
            print("Previše malo! Pokušaj ponovo.")
        elif pokusaj > broj:
            print("Previše veliko! Pokušaj ponovo.")
        else:
            print(f"Čestitam! Pogodio si broj {broj} u {pokusaji} pokušaja.")
            break

pogodi_broj()
```

*IDLE Shell 3.10.11*

File Edit Shell Debug Options Window Help

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa11.py =
======
Pogodi broj (1-5): 4
Previše veliko! Pokušaj ponovo.
Pogodi broj (1-5): 2
Previše malo! Pokušaj ponovo.
Pogodi broj (1-5): 3
Čestitam! Pogodio si broj 3 u 3 pokušaja.
>>>
```

Slika 55. *Primjer kreiranja igrice Pogađanja broja*

klasa12.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa12.py (3....

File Edit Format Run Options Window Help

```
import random

def igra():
    opcije = ["kamen", "papir", "makaze"]
    racunar = random.choice(opcije)
    igrac = input("Izaberi: kamen, papir ili makaze? ").lower()

    if igrac not in opcije:
        print("Nevažeći unos!")
        return

    print(f"Računar je izabrao: {racunar}")

    if igrac == racunar:
        print("Neriješeno!")
    elif (igrac == "kamen" and racunar == "makaze") or \
         (igrac == "papir" and racunar == "kamen") or \
         (igrac == "makaze" and racunar == "papir"):
        print("Pobijedio si!")
    else:
        print("Izgubio si!")

igra()
```

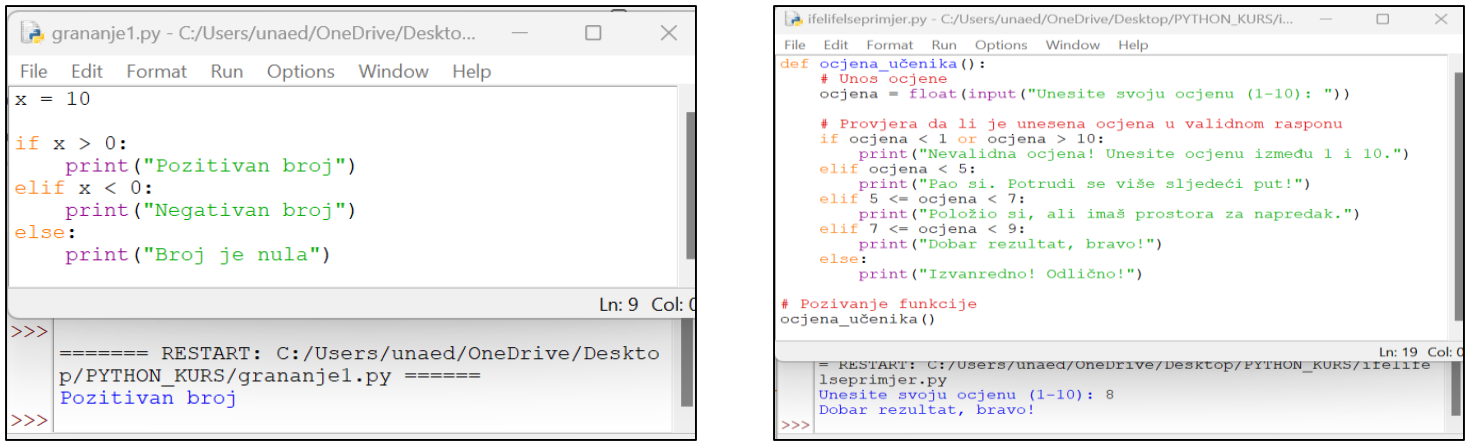
Ln: 24 Col

```
======== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klas
a12.py =======
Izaberi: kamen, papir ili makaze? kamen
Računar je izabrao: makaze
Pobijedio si!
>>>
======== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klas
a12.py =======
Izaberi: kamen, papir ili makaze? papir
Računar je izabrao: makaze
Izgubio si!
>>>
```

Slika 56. *Primjer kreiranja igrice Kamen, papir, makaze*

# 7. Upravljačke naredbe

- Upravljačke naredbe su ključni element svakog programskog jezika, pa tako i u Pythonu, jer omogućavaju kontrolu toka izvršavanja programa. Kroz naredbe kao što su if, elif, else, for, while, break, continue, i pass, Python omogućava uslovnu i petljastu kontrolu, kao i mogućnost upravljanja tokom izvršavanja programa na temelju različitih uvjeta.
- Uslovne naredbe grananja omogućavaju donošenje odluka na osnovu uslova.




Slika 57. *Primjer primjene naredbi grananja*

# 7. Upravljačke naredbe

- Petlje (for, while) omogućavaju ponavljanje dijela koda više puta.

forprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/forprimjer.py (3.10.11)

File Edit Format Run Options Window Help

```
def prosti_brojevi_opseg():
    # Unos početnog i krajnjeg broja
    pocetak = int(input("Unesite početni broj opsega: "))
    kraj = int(input("Unesite krajnji broj opsega: "))

    # Lista za čuvanje prostih brojeva
    prosti_brojevi = []
    suma = 0
    broj_prostih = 0

    # Provjera svih brojeva u opsegu
    for broj in range(pocetak, kraj + 1):
        # Provjera da li je broj prost
        if broj > 1:
            prost = True
            for i in range(2, broj):
                if broj % i == 0:
                    prost = False
                    break
            if prost:
                prosti_brojevi.append(broj)
                suma += broj
                broj_prostih += 1

    # Ispis rezultata
    print(f"Prosti brojevi u opsegu od {pocetak} do {kraj} su: {prosti_brojevi}")
    print(f"Suma prostih brojeva je: {suma}")
    print(f"Broj prostih brojeva u opsegu je: {broj_prostih}")

# Pozivanje funkcije
prosti_brojevi_opseg()
```

IDLE Shell 3.10.11

File Edit Shell Debug Options Window Help

```
Dobar rezultat, bravo!
>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/forprimjer.py
Unesite početni broj opsega: 5
Unesite krajnji broj opsega: 25
Prosti brojevi u opsegu od 5 do 25 su: [5, 7, 11, 13, 17, 19, 23]
Suma prostih brojeva je: 95
Broj prostih brojeva u opsegu je: 7
>>>
```

Slika 58. *Primjer primjene for*

whileprimjer1.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/whileprimjer...

File Edit Format Run Options Window Help

```
def provjera_prostog_broja():
    while True:
        try:
            broj = int(input("Unesite broj za provjeru je li prost: "))

            # Provjera da li je broj manji od 2 (ne može biti prost broj)
            if broj < 2:
                print("Broj mora biti veći od 1!")
                continue

            # Provjera prostog broja
            je_prost = True
            for i in range(2, broj):
                if broj % i == 0:
                    je_prost = False
                    break

            # Ispisivanje rezultata
            if je_prost:
                print(f"Broj {broj} je prost!")
                break
            else:
                print(f"Broj {broj} nije prost. Pokušajte ponovo.")

        except ValueError:
            print("Molimo vas da unesete validan broj!")

# Pozivanje funkcije
provjera_prostog_broja()
```

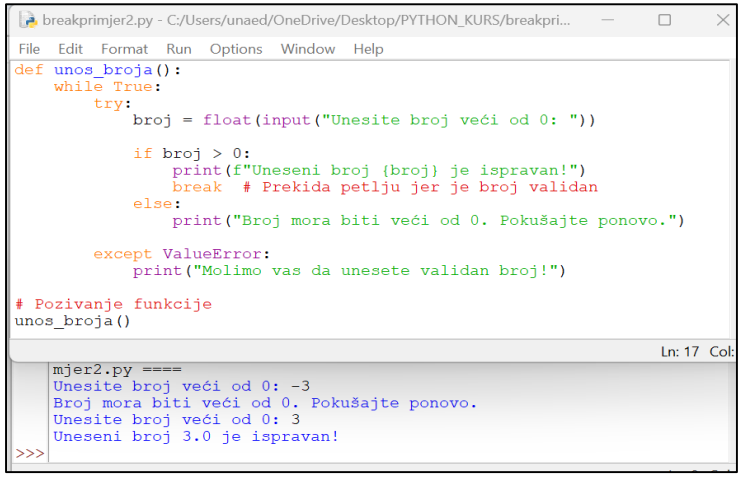
Ln: 30 Col:

```
>>>
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/whileprimje
r1.py ====
Unesite broj za provjeru je li prost: 6
Broj 6 nije prost. Pokušajte ponovo.
Unesite broj za provjeru je li prost: 7
```

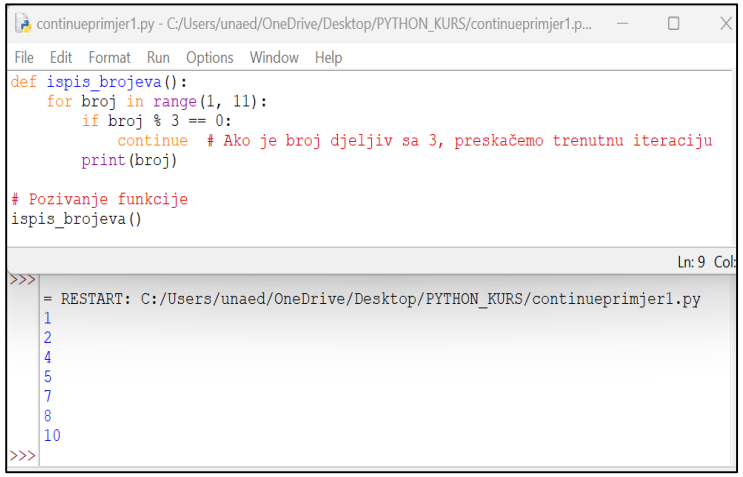
Slika 59. *Primjer primjene while*

# 7. Upravljačke naredbe

- Naredba break se koristi za prekidanje petlje (bilo for ili while) prije nego što je dosegla svoj normalni kraj. Kada se break izvrši, petlja odmah prestaje s radom, a program prelazi na naredbu koja slijedi nakon petlje.
- Naredba continue koristi se za preskakanje trenutne iteracije petlje i prelazak na sljedeću.



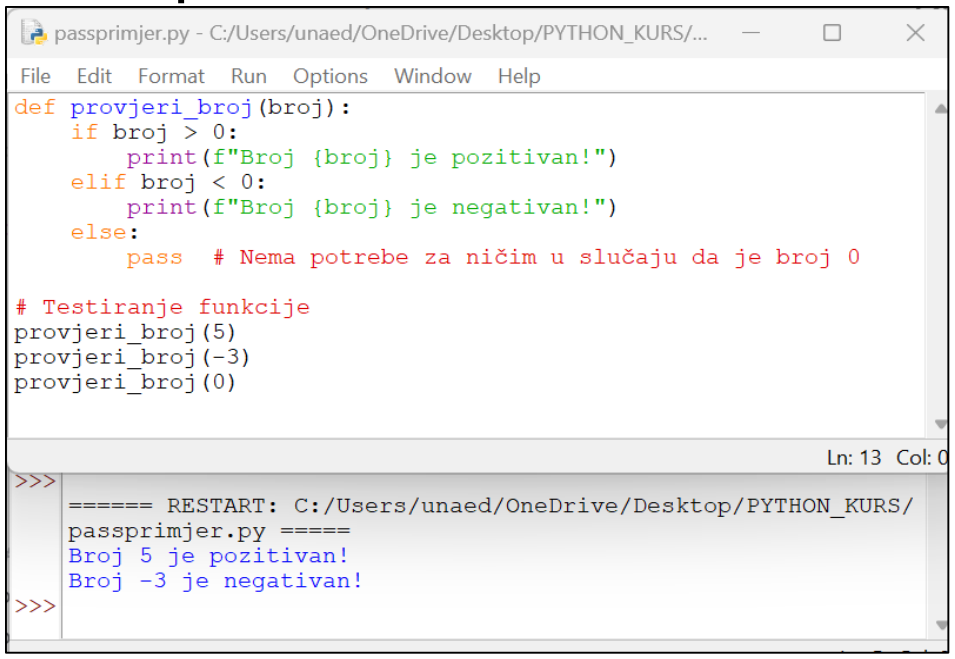

Slika 60. *Primjer primjene naredbe break*




Slika 61. *Primjer primjene naredbe continue*

# 7. Upravljačke naredbe

- Naredba pass je prazna naredba koja se koristi kao placeholder u situacijama gdje sintaktički Python očekuje neki kod, ali ništa se ne želi izvršiti. Često se koristi u definicijama funkcija, petlji ili uvjetnim granama koje još nisu implementirane.




Slika 62. *Primjer primjene naredbe pass*

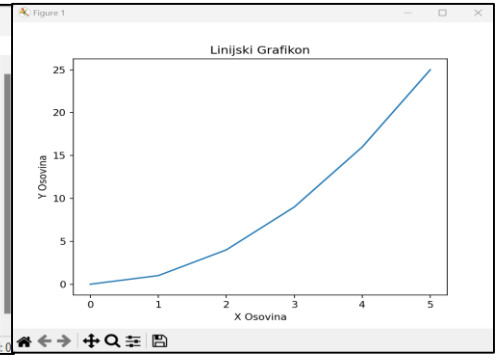
Co-funded by the European Union

# 8. Funkcije za grafički prikaz

- U Pythonu, za vizualizaciju podataka u 2D i 3D prostoru, najčešće se koristi paket Matplotlib. Matplotlib omogućava jednostavan i fleksibilan način za generiranje širokog spektra grafova i dijagrama, uključujući 2D i 3D prikaze podataka. Osim toga, za napredniji grafički prikaz i interaktivne 3D grafike, možemo koristiti mpl_toolkits.mplot3d, koji je ekstenzija Matplotlib-a koja omogućava rad sa 3D podacima.
- 2D grafički prikazi su najčešći i koriste se za vizualizaciju podataka u dvodimenzionalnom prostoru. Matplotlib pruža bogat skup funkcionalnosti za crtanje linijskih grafova, scatter grafova, histogram grafova, bar grafova i drugih.

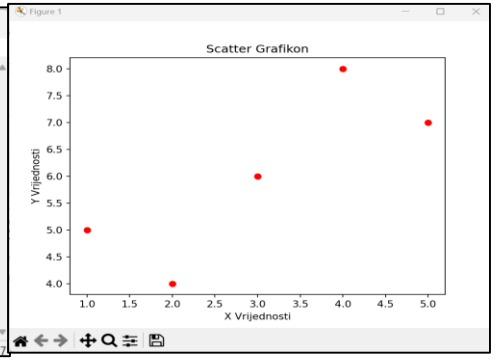
# 8. Funkcije za grafički prikaz

```
import matplotlib.pyplot as plt

# Definisanje podataka
x = [0, 1, 2, 3, 4, 5]
y = [0, 1, 4, 9, 16, 25]

# Crtanje linijskog grafikona
plt.plot(x, y)

# Oznaka za osovinu x i y
plt.xlabel('X Osovina')
plt.ylabel('Y Osovina')

# Naslov grafikona
plt.title('Linijski Grafikon')

# Prikazivanje grafikona
plt.show()
```



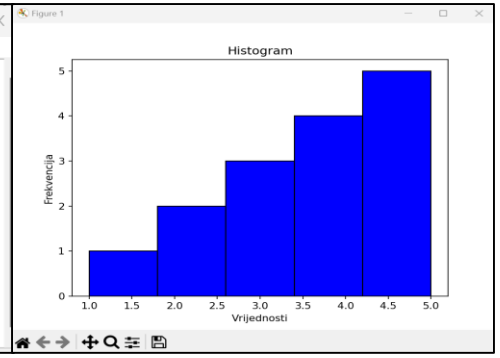

Slika 63. *Primjer linijskog grafa*

```
import matplotlib.pyplot as plt

# Podaci za scatter grafikon
x = [1, 2, 3, 4, 5]
y = [5, 4, 6, 8, 7]

# Crtanje scatter grafikona
plt.scatter(x, y, color='red')

# Oznake
plt.xlabel('X Vrijednosti')
plt.ylabel('Y Vrijednosti')

# Naslov
plt.title('Scatter Grafikon')

# Prikazivanje grafikona
plt.show()
```



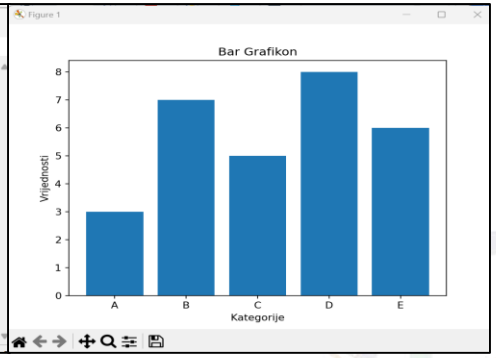

Slika 64. *Primjer scatter grafa*

```
import matplotlib.pyplot as plt

# Podaci za histogram
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

# Crtanje histograma
plt.hist(data, bins=5, color='blue', edgecolor='black')

# Oznake
plt.xlabel('Vrijednosti')
plt.ylabel('Frekvencija')

# Naslov
plt.title('Histogram')

# Prikazivanje grafikona
plt.show()
```




Slika 65. *Primjer histograma*

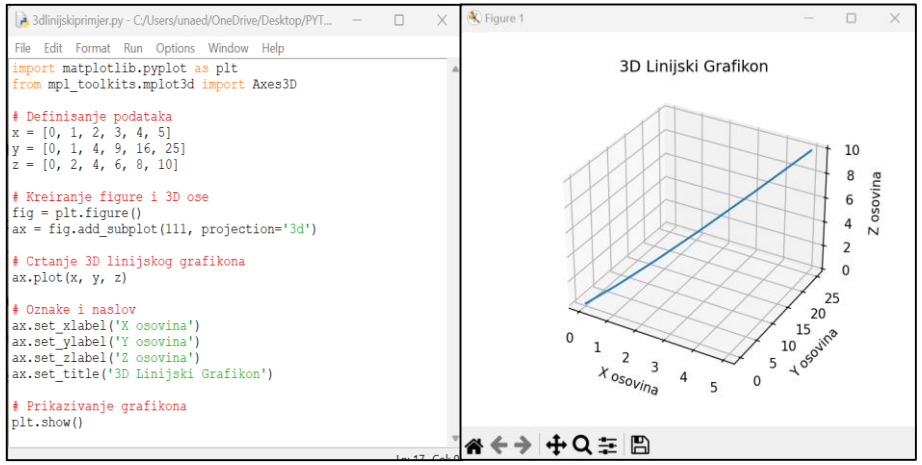
```
import matplotlib.pyplot as plt

# Kategorije i vrednosti
categories = ['A', 'B', 'C', 'D', 'E']
values = [3, 7, 5, 8, 6]

# Crtanje bar grafikona
plt.bar(categories, values)

# Oznake
plt.xlabel('Kategorije')
plt.ylabel('Vrijednosti')

# Naslov
plt.title('Bar Grafikon')

# Prikazivanje grafikona
plt.show()
```



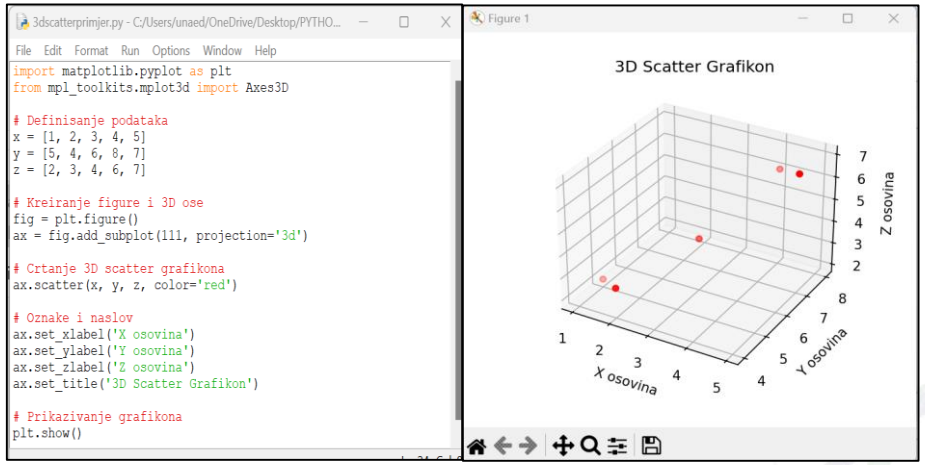

Slika 66. *Primjer bar grafa*

# 8. Funkcije za grafički prikaz

- Za rad sa 3D grafovima u Pythonu koristi se ekstenzija mpl_toolkits.mplot3d, koja omogućava crtanje trodimenzionalnih objekata kao što su 3D linijski grafovi, scatter grafovi, površinski grafovi, itd.

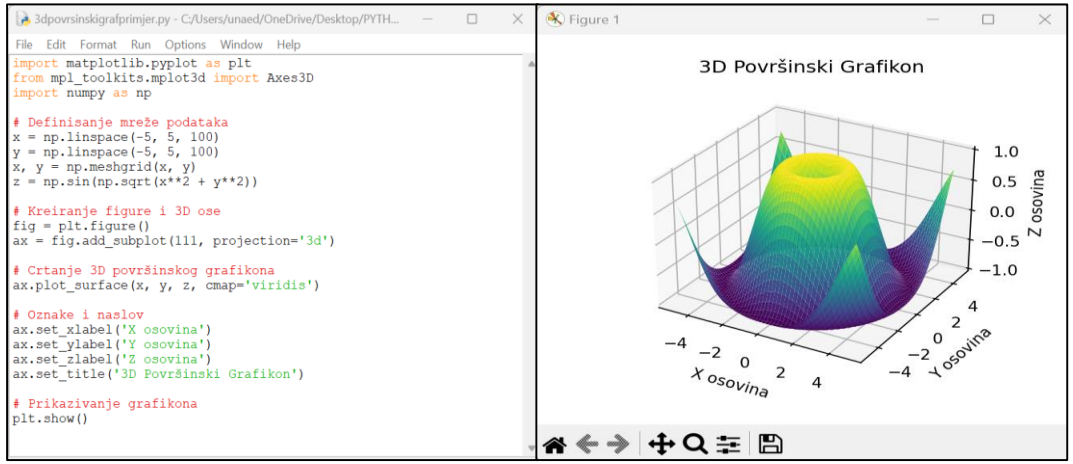



Slika 67. *Primjer 3D linijskog grafa*

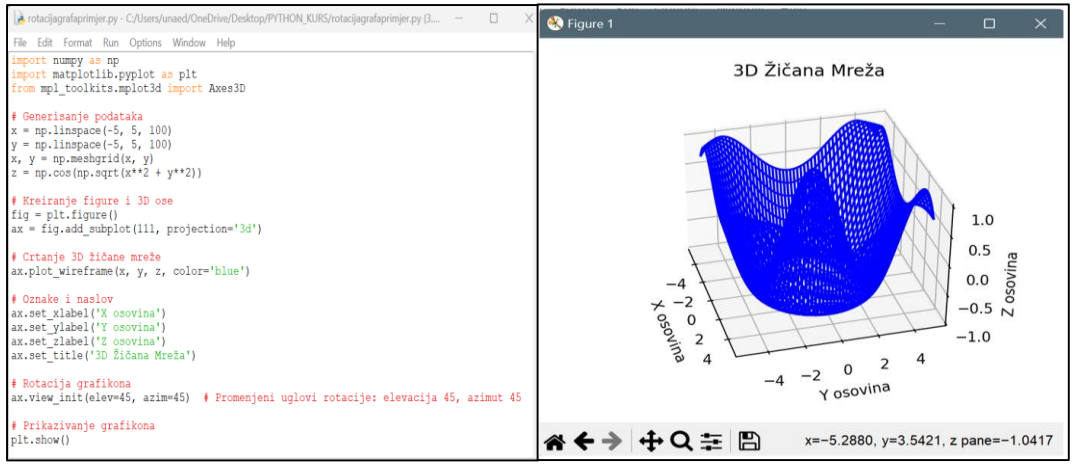



Slika 68. *Primjer 3D scatter grafa*

# 8. Funkcije za grafički prikaz




Slika 69. *Primjer 3D površinskog grafa*




Slika 70. *Primjer rotacije 3D grafa*

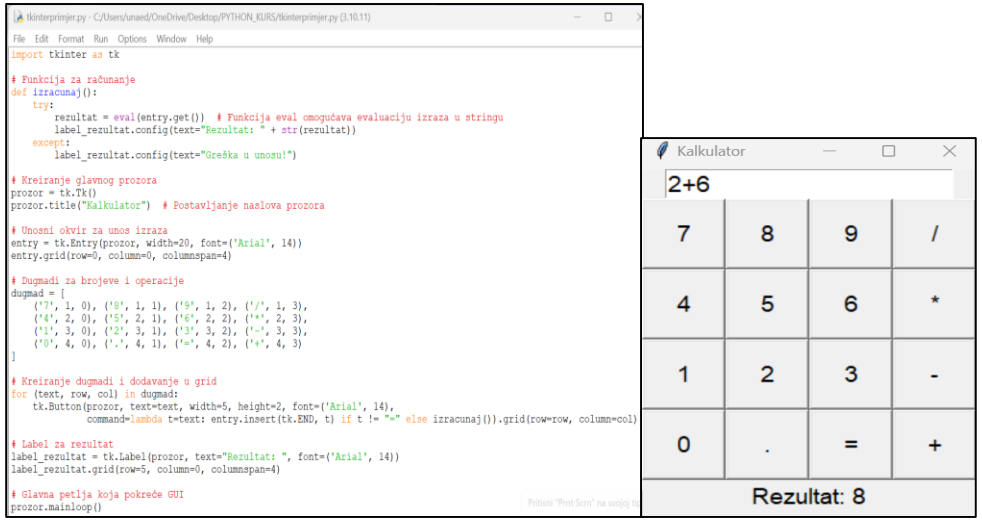
# 9. Grafički interfejs

- Grafički korisnički interfejs (GUI) u Pythonu omogućava razvoj aplikacija s interaktivnim prozorima, gumbima, unosima i drugim grafičkim komponentama. U Pythonu postoje različiti paketi i biblioteke koje omogućavaju kreiranje GUI aplikacija, a najpoznatiji i najčešće korišteni su:

  1. Tkinter – Osnovna biblioteka za GUI u Pythonu, koja je standardno uključena u Python distribuciju.
  2. PyQt – Moćna biblioteka koja se temelji na Qt frameworku, koja pruža veće mogućnosti i fleksibilnost.
  3. wxPython – Također popularan GUI framework, zasnovan na wxWidgets biblioteci.
  4. Kivy – Biblioteka za izradu GUI aplikacija s naglaskom na mobilne uređaje i touch-based aplikacije.

# 9. Grafički interfejs

- Tkinter je standardna biblioteka u Pythonu za izradu desktop aplikacija. Pruža jednostavan način za kreiranje prozora, dugmadi, tekstualnih polja, i drugih GUI elemenata.



```
import tkinter as tk

# Funkcija za računanje
def izracunaj():
    try:
        rezultat = eval(entry.get())  # Funkcija eval omogućava evaluaciju izraza u stringu
        label_rezultat.config(text="Rezultat: " + str(rezultat))
    except:
        label_rezultat.config(text="Greška u unosu!")

# Kreiranje glavnog prozora
prozor = tk.Tk()
prozor.title("Kalkulator")  # Postavljanje naslova prozora

# Unosni okvir za unos izraza
entry = tk.Entry(prozor, width=20, font=('Arial', 14))
entry.grid(row=0, column=0, columnspan=4)

# Dugmadi za brojeve i operacije
dugmad = [
    ('7', 1, 0), ('8', 1, 1), ('9', 1, 2), ('/', 1, 3),
    ('4', 2, 0), ('5', 2, 1), ('6', 2, 2), ('*', 2, 3),
    ('1', 3, 0), ('2', 3, 1), ('3', 3, 2), ('-', 3, 3),
    ('0', 4, 0), ('.', 4, 1), ('=', 4, 2), ('+', 4, 3)
]

# Kreiranje dugmadi i dodavanje u grid
for (text, row, col) in dugmad:
    tk.Button(prozor, text=text, width=5, height=2, font=('Arial', 14),
              command=lambda t=text: entry.insert(tk.END, t) if t != "=" else izracunaj()).grid(row=row, column=col)

# Label za rezultat
label_rezultat = tk.Label(prozor, text="Rezultat: ", font=('Arial', 14))
label_rezultat.grid(row=5, column=0, columnspan=4)

# Glavna petlja koja pokreće GUI
prozor.mainloop()
```



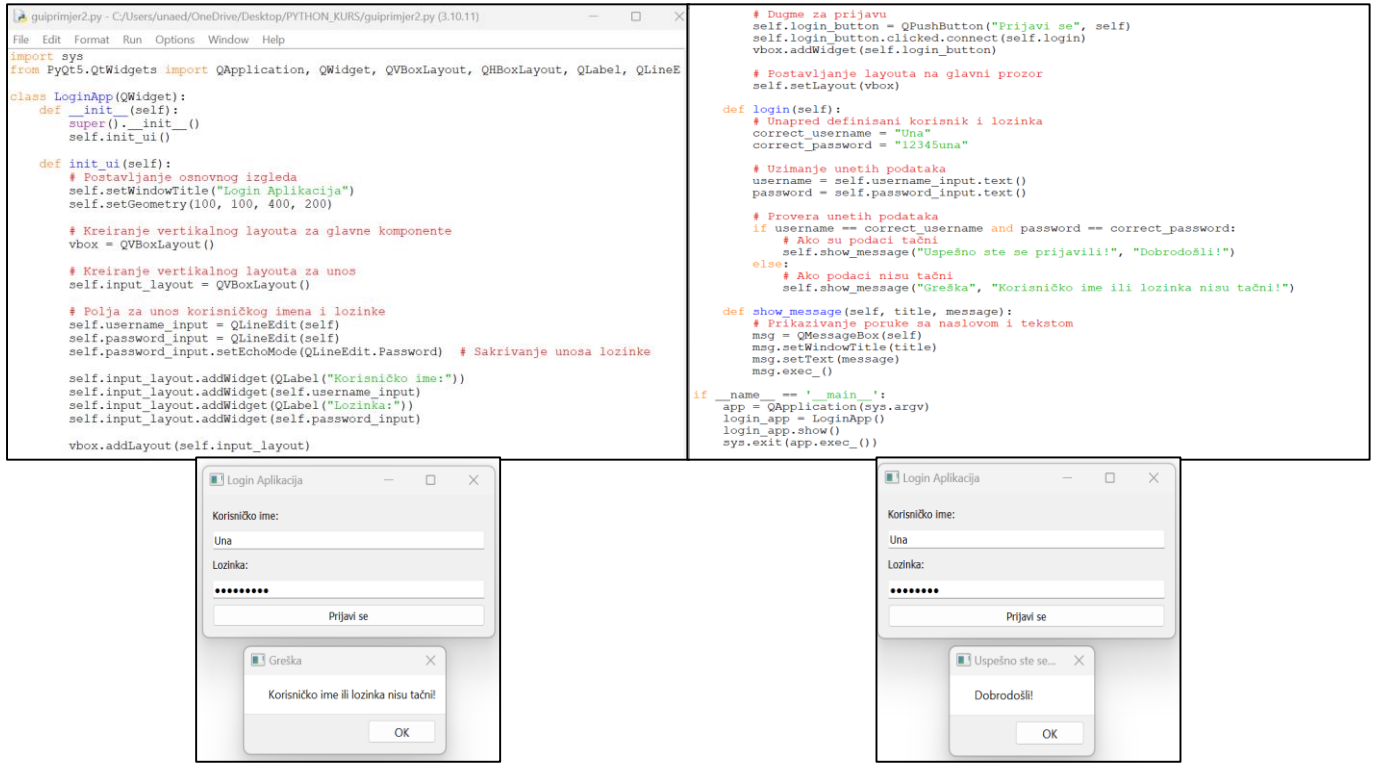
Slika 71. *Primjer primjene Tkinter biblioteke*

Osnovni elementi Tkinter-a:

1. **Tk()**: Glavni prozor aplikacije.
2. **Label()**: Prikazuje tekst.
3. **Button()**: Kreira dugme.
4. **Entry()**: Unosni okvir za tekst.
5. **Canvas()**: Površina za crtanje.
6. **MessageBox()**: Dijaloški okvir za prikazivanje poruka.

# 9. Grafički interfejs

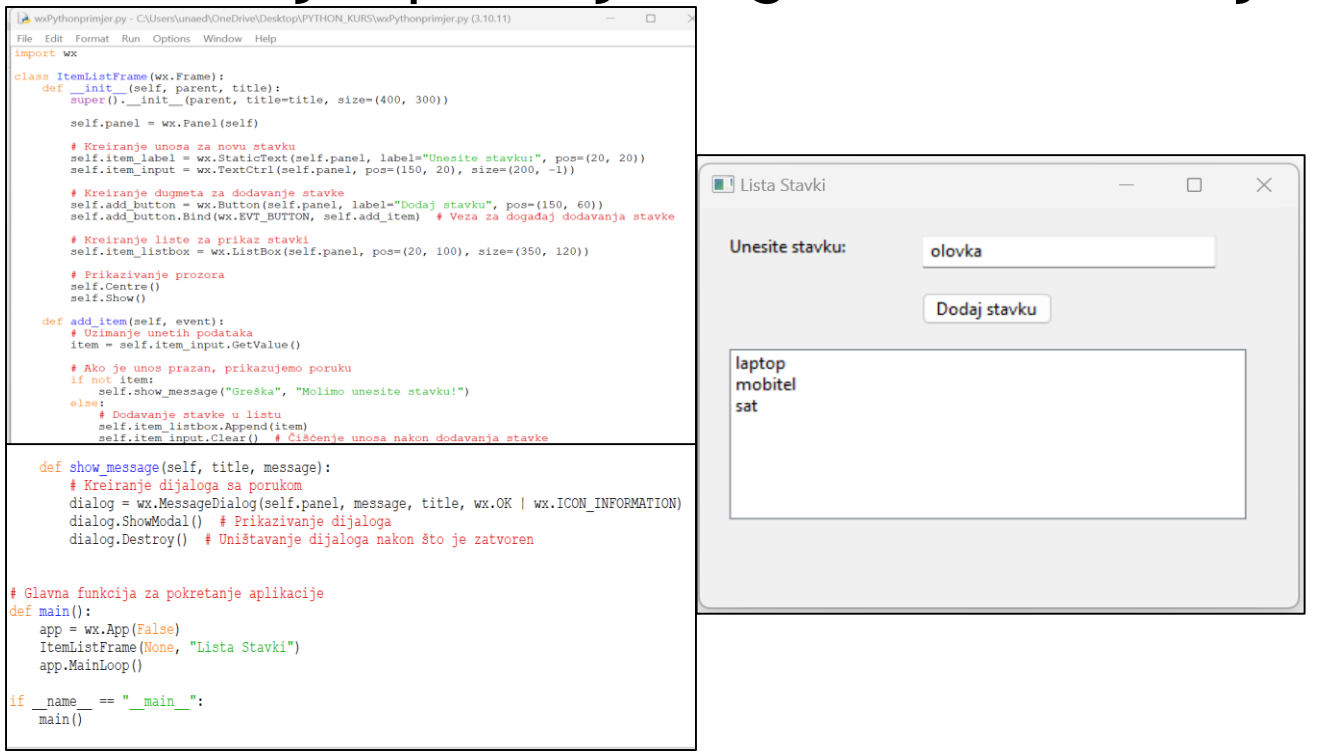
- PyQt5 je biblioteka koja se temelji na Qt frameworku, koja pruža veće mogućnosti i fleksibilnost.




Slika 73. *Primjer primjene PyQt5 biblioteke (Aplikacija za prijavu korisnika)*

# 9. Grafički interfejs

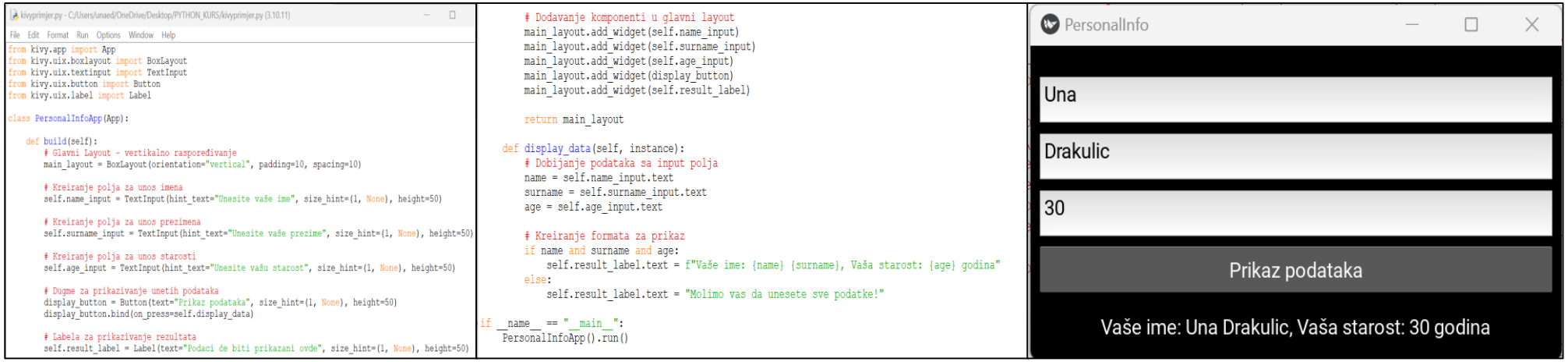
- wxPython je biblioteka za razvoj grafičkog korisničkog interfejsa (GUI) u Pythonu. To je Python wrapper za wxWidgets, koji je C++ biblioteka koja omogućava kreiranje aplikacija sa grafičkim interfejsom.




Slika 74. *Primjer primjene wxPython biblioteke*

# 9. Grafički interfejs

- Kivy je biblioteka za izradu GUI aplikacija s naglaskom na mobilne uređaje i touch-based aplikacije.




Slika 75. *Primjer primjene Kivy biblioteke*

Co-funded by
the European Union

# 9. Grafički interfejs

- Primjer kreiranja GUI aplikacije koja omogućava korisnicima da:
  a) izaberu oblik koji žele da nacrtaju (krug ili pravougaonik),
  b) klikom na platno da nacrtaju izabrani oblik,
  c) promjene boju oblika,
  d) isprazne platno ako žele da počnu ispočetka,

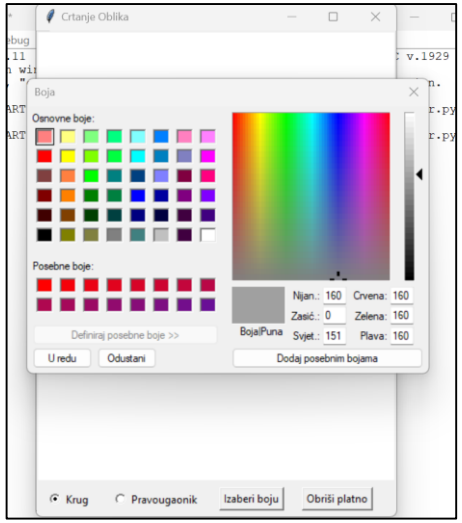
  je dat na slici 76.

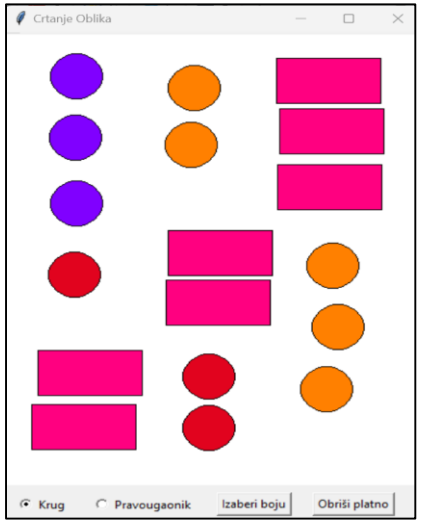
# 9. Grafički interfejs

gui primjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/gui primjer.py (3.10.11)

File Edit Format Run Options Window Help

```
import tkinter as tk
from tkinter import colorchooser

# Funkcija za crtanje pravougaonika
def draw_rectangle(event):
    if shape_var.get() == "Rectangle":
        canvas.create_rectangle(event.x - 50, event.y - 25, event.x + 50, event.y + 25, fill=color_var.get())

# Funkcija za crtanje kruga
def draw_circle(event):
    if shape_var.get() == "Circle":
        canvas.create_oval(event.x - 25, event.y - 25, event.x + 25, event.y + 25, fill=color_var.get())

# Funkcija za promenu boje oblika
def change_color():
    color_code = colorchooser.askcolor()[1]
    color_var.set(color_code)

# Funkcija za brisanje platna
def clear_canvas():
    canvas.delete("all")

# Kreiranje glavnog prozora
root = tk.Tk()
root.title("Crtanje Oblika")

# Varijable za oblik i boju
shape_var = tk.StringVar(value="Circle")  # Podrazumevani oblik je krug
color_var = tk.StringVar(value="black")   # Podrazumevana boja je crna

# Kreiranje widgeta
canvas = tk.Canvas(root, width=500, height=500, bg="white")
canvas.pack()

# Dugmadi za biranje oblika
circle_button = tk.Radiobutton(root, text="Krug", variable=shape_var, value="Circle")
circle_button.pack(side=tk.LEFT, padx=10)

rectangle_button = tk.Radiobutton(root, text="Pravougaonik", variable=shape_var, value="Rectangle")
rectangle_button.pack(side=tk.LEFT, padx=10)

# Dugme za promenu boje
color_button = tk.Button(root, text="Izaberi boju", command=change_color)
color_button.pack(side=tk.LEFT, padx=10)

# Dugme za brisanje platna
clear_button = tk.Button(root, text="Obriši platno", command=clear_canvas)
clear_button.pack(side=tk.LEFT, padx=10)

# Veza između platna i funkcija za crtanje
canvas.bind("<Button-1>", lambda event: draw_circle(event) if shape_var.get() == "Circle" else draw_rectangle(event))

# Pokretanje glavne petlje
root.mainloop()
```




Slika 76. *Primjer GUI aplikacije*

Co-funded by the European Union

# 10. Rad sa datotekama

- Rad sa datotekama u Pythonu je vrlo važan za mnoge aplikacije, od jednostavnih programa koji čitaju i pišu podatke do složenih aplikacija za analizu podataka. Python pruža jednostavne funkcije za otvaranje, čitanje, pisanje i zatvaranje datoteka.
- Za rad sa datotekama u Pythonu koristi se funkcija *open()*. Ona otvara datoteku i omogućava rad sa njom.
- Sintaksa je: *file = open ('ime_datoteke', 'mod')*, gdje *'ime_datoteke'* predstavlja naziv datoteke koja se želi otvoriti (može biti apsolutni ili relativni put), a *'mod'* određuje kako će se datoteka otvoriti.
- Najčešći modovi su: *'r'*-čitanje (read); *'w'*-pisanje (write); *'a'*-dodavanje (append); *'rb'*-čitanje u binarnom formatu; *'wb'*-pisanje u binarnom formatu.

# 10. Rad sa datotekama

- Sintaksa za čitanje sadržaja je

```
file = open('ime_datoteke.txt', 'r')

content = file.read()

print(content)

file.close()
```

- Metoda readline() čita jednu liniju iz datoteke.

```
file = open('ime_datoteke.txt', 'r')
line = file.readline()
 while line:
        print(line, end='')       # end='' je da ne bismo dodavali dodatne praznine
        line = file.readline()
file.close()
```

# 10. Rad sa datotekama

- Za pisanje u datoteku koristimo režim 'w' ili 'a'. Režim 'w' briše sadržaj datoteke ako datoteka već postoji, dok režim 'a' dodaje nove podatke na kraj datoteke.

```
file = open('ime_datoteke.txt', 'w')
file.write('Ovo je neki tekst.\n')
file.write('Ovo je još jedan red.')
 file.close()
```

- Dodavanje sadržaja u datoteku:

```
file = open('ime_datoteke.txt', 'a')
file.write('Dodajemo još jedan red u datoteku.\n')
file.close()
```

# 10. Rad sa datotekama

- Ako se radi sa binarnim datotekama, može se koristiti 'rb' i 'wb' modovi za čitanje i pisanje u binarnom formatu.

```
file = open('image.jpg', 'rb')
binary_data = file.read()
print(binary_data)      # Ispisivanje binarnih podataka
file.close()
```

- Pisanje u binarnu datoteku:

```
file = open('new_image.jpg', 'wb')
file.write(binary_data)
file.close()
```

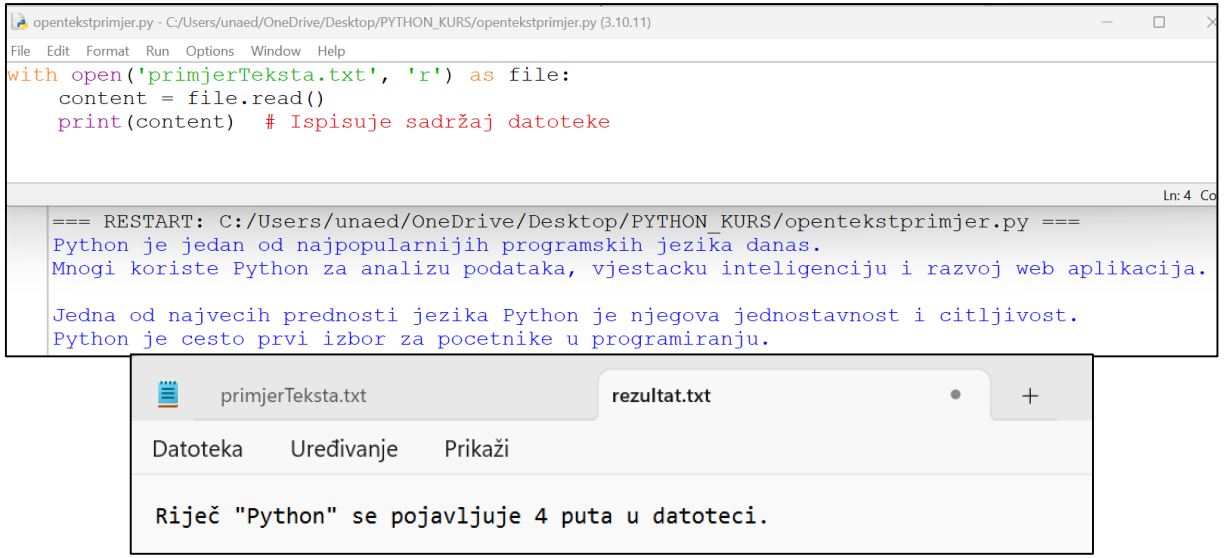
# 10. Rad sa datotekama

- Umjesto da pozivamo file.close() eksplicitno, možemo koristiti with izjavu koja automatski zatvara datoteku kada se završi rad sa njom. Ovo je bolji način jer osigurava da će datoteka biti zatvorena, čak iako dođe do greške u toku obrade.

```
with open('ime_datoteke.txt', 'r') as file:
    content = file.read()
    print(content)
```

# 10. Rad sa datotekama

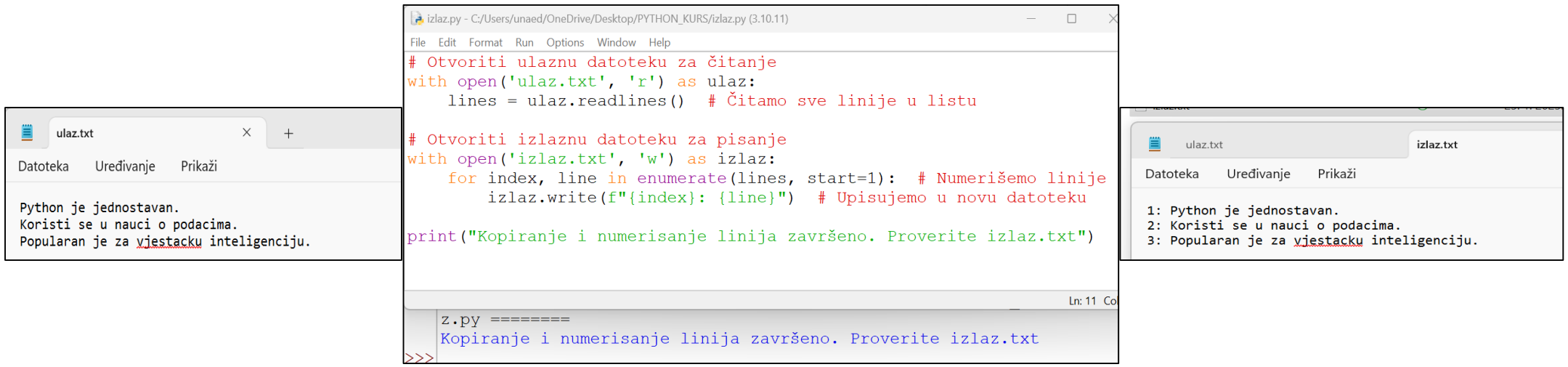
- Primjer rada sa datotekama je dat na slici 77. Kreiran je program koji čita sadržaj tekstualne datoteke, broji koliko puta se određena riječ pojavljuje u datoteci, a zatim upisuje broj u novu datoteku.




Slika 77. *Primjer rada sa datotekom*

# 10. Rad sa datotekama

- Primjer, kopiranje sadržaja iz jedne datoteke u drugu i numeriranje linija, je prikazan na slici 78.




Slika 78. *Primjer rada sa datotekom*

# Zadaci za vježbu

- Napisati program koji predstavlja jednostavan sistem za upravljanje bibliotekom, omogućavajući korisniku da:

1) Dodaje knjige – Unosi naslov i autora knjige u bazu podataka
2) Pregleda knjige – Prikazuje sve knjige u bazi koristeći tablicu
3) Pretražuje knjige – Omogućava korisniku da pretraži knjige prema naslovu ili autoru.
4) Ažurira podatke knjige – Omogućava korisniku da promijeni naslov ili autora odabrane knjige
5) Briše knjige – Omogućava korisniku da izbriše knjigu iz baze prema njenom ID-u
6) Izlazi iz programa – Zatvara aplikaciju

Co-funded by the European Union

# Zadaci za vježbu

- Napraviti aplikaciju koja omogućava dodavanje, pregled, ažuriranje i brisanje studenata i njihovih ocjena. Omogućiti računanje prosječne ocjene i sortiranje studenata po prosjeku.

  Funkcionalnosti aplikacije su:

1) Dodavanje studenta (ime, prezime)
2) Dodavanje ocjena studentu
3) Prikaz svih studenata sa prosjecima
4) Ažuriranje ocjena
5) Brisanje studenta
6) Sortiranje po prosjeku

Co-funded by the European Union

# Zadaci za vježbu

- Napisati program koji omogućava korisniku izračunavanje površine i obima za različite geometrijske oblike. Program koristi Python i matematičke funkcije iz biblioteke math za računanje površina i obima oblika kao što su: krug, pravougaonik, trougao, trapez i paralelogram.
- Napisati program koji omogućava korisniku da putem GUI grafičke aplikacije izračuna površinu i obim osnovnih geometrijskih oblika.
- Napisati program koji omogućava korisnicima da crtaju slobodne linije na grafičkom platnu koristeći miš. Kada korisnik klikne na platno (lijevi klik miša), početne koordinate (gdje je kliknuo) se pamte. Kada korisnik povuče miš dok drži pritisnut lijevi klik, linija se povlači od tačke gdje je prethodno kliknuo do trenutne pozicije miša. Svaki povučeni segment linije treba da spaja prethodne i trenutne koordinate, stvarajući neprekidnu crtu.

WBnet

Co-funded by the European Union

# PITANJA I ODGOVORI

*"Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them."*

**Network of centers for regional short study programs in the countries of the Western Balkans**

**Call:** ERASMUS-EDU-2023-CBHE

**Project number:** 101128813

ATYCC

UNIVERSITY OF L]UBL]ANA
Faculty of Electrical Engineering

POLITÉCNICA

University of Pristina
Kosovska Mitrovica