



Co-funded by
the European Union

Univerzitet u Bihaću
Tehnički fakultet

OSNOVE PROGRAMIRANJA PYTHON

Una Drakulić MA, Melisa Haurdić MA

UNBI



UNIVERSITY OF LJUBLJANA
Faculty of Electrical Engineering



University of Pristina
Kosovska Mitrovica



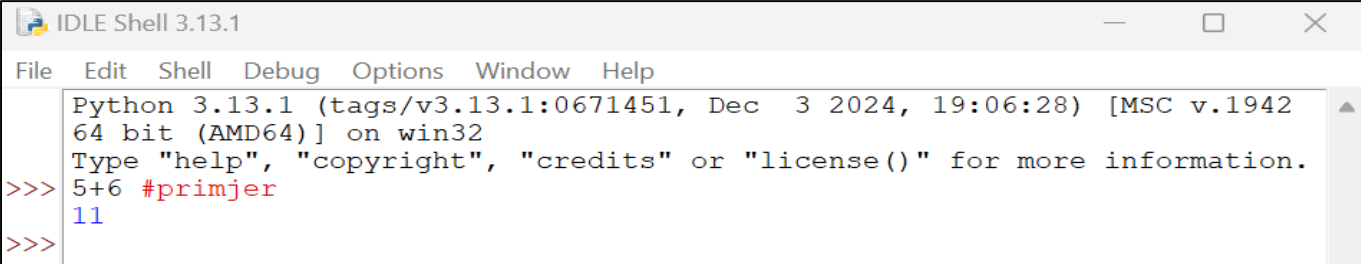
1. Uvod u Python

- 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.



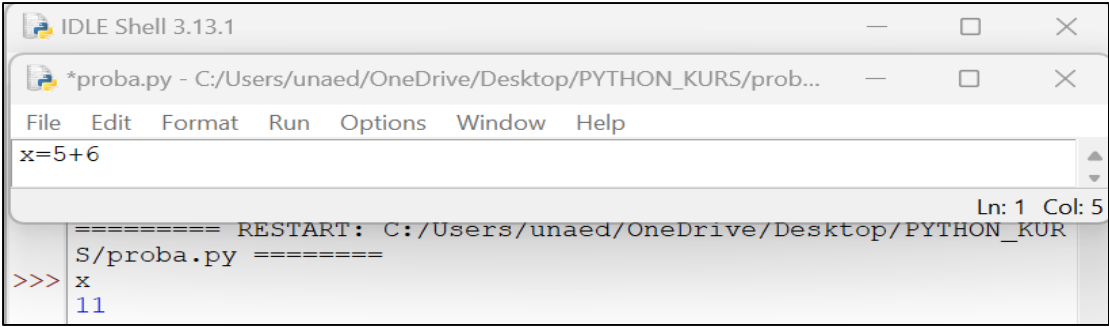
```
Python 3.13.1 (tags/v3.13.1:0671451, Dec 3 2024, 19:06:28) [MSC v.1942  
64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> 5+6 #primjer  
11  
>>>
```

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

- Tekst iza oznake # se smatra komentárom 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.



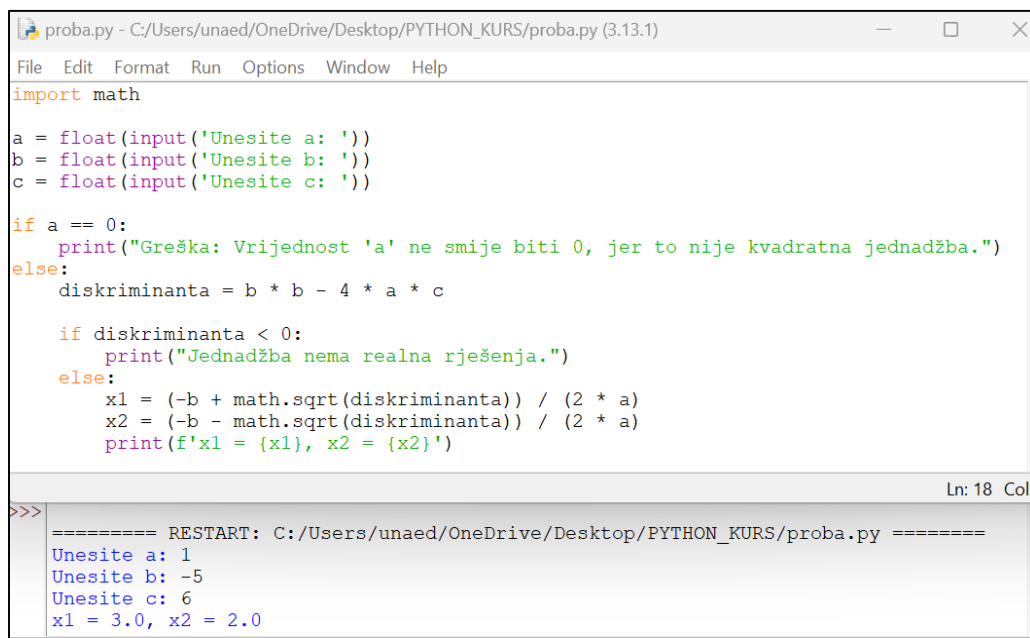
```
IDLE Shell 3.13.1
*proba.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/prob...
File Edit Format Run Options Window Help
x=5+6
Ln: 1 Col: 5
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KUR
S/proba.py =====
>>> x
11
```

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



2. Osnovni pojmovi

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



```
proba.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/proba.py (3.13.1)
File Edit Format Run Options Window Help
import math
a = float(input('Unesite a: '))
b = float(input('Unesite b: '))
c = float(input('Unesite c: '))

if a == 0:
    print("Greška: Vrijednost 'a' ne smije biti 0, jer to nije kvadratna jednadžba.")
else:
    diskriminanta = b * b - 4 * a * c

    if diskriminanta < 0:
        print("Jednadžba nema realna rješenja.")
    else:
        x1 = (-b + math.sqrt(diskriminanta)) / (2 * a)
        x2 = (-b - math.sqrt(diskriminanta)) / (2 * a)
        print(f'x1 = {x1}, x2 = {x2}')

Ln: 18 Col:

>>>
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/proba.py =====
Unesite a: 1
Unesite b: -5
Unesite c: 6
x1 = 3.0, x2 = 2.0
```

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

Math je ugrađeni Python paket koji pruža matematičke funkcije i konstante.

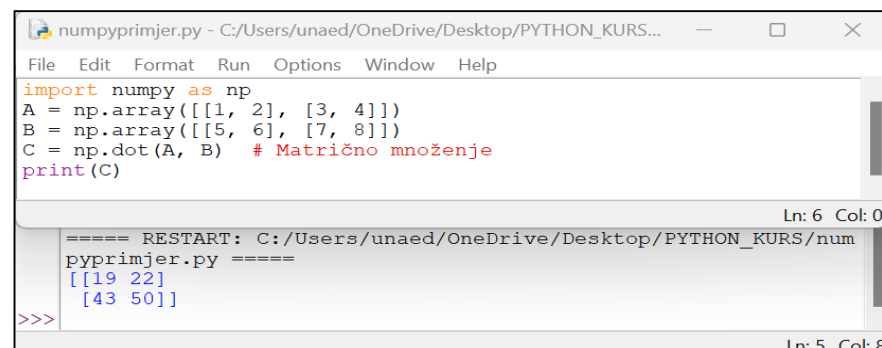
Najčešće korištene su:

- `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.



```
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
C = np.dot(A, B) # Matrično množenje
print(C)
```

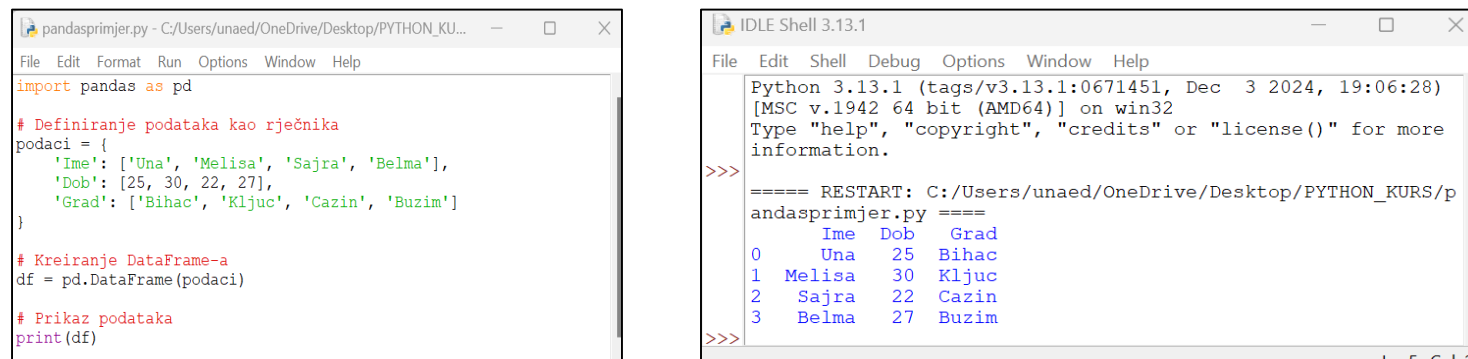
Ln: 6 Col: 0

```
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/num
pyprimjer.py =====
[[19 22]
 [43 50]]
>>>
```

Ln: 5 Col: 8

Slika 4. Primjer upotrebe numpy paketa

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



```
import pandas as pd

# Definiranje podataka kao rječnika
podaci = {
    'Ime': ['Una', 'Melisa', 'Sajra', 'Belma'],
    'Dob': [25, 30, 22, 27],
    'Grad': ['Bihac', 'Kljuc', 'Cazin', 'Buzim']
}

# Kreiranje DataFrame-a
df = pd.DataFrame(podaci)

# Prikaz podataka
print(df)
```

Python 3.13.1 (tags/v3.13.1:0671451, Dec 3 2024, 19:06:28)
[MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.

```
>>>
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/p
andasprimjer.py =====
   Ime  Dob  Grad
0  Una   25  Bihac
1 Melisa 30  Kljuc
2  Sajra 22  Cazin
3  Belma 27  Buzim
>>>
```

Ln: 5 Col: 2

Slika 5. Primjer upotrebe pandas paketa



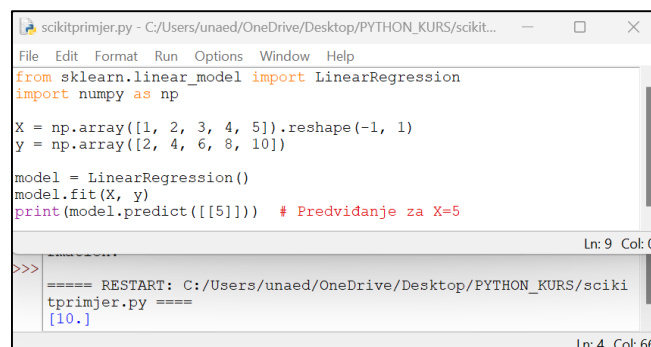
2. Osnovni pojmovi

- *Matplotlib* je standardni paket za crtanje grafova.



Slika 6. Primjer upotrebe matplotlib paketa

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



The screenshot shows a Python script in a text editor window titled 'scikitprimjer.py'. The script imports 'LinearRegression' from 'sklearn.linear_model' and 'numpy' as 'np'. It creates arrays 'X' and 'y', fits a 'LinearRegression' model, and prints the prediction for 'X=5'. The output shows the prediction is 10.0.

```
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)

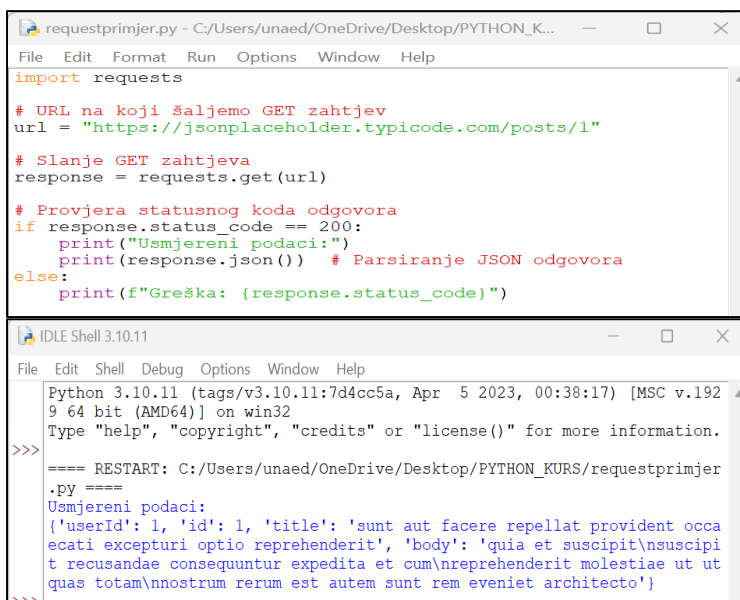
# Evaluacija modela
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test accuracy: {accuracy}")
```

```
IDLE Shell 3.10.11
File Edit Shell Debug Options Window Help
[Progress bar] [1m12/12] [0m] [32m]
[Progress bar] [0m] [37m] [0m] [1m0s] [0m] 2ms/step - accuracy: 0.8986 - loss: 0.5300
Epoch 50/50
[Progress bar] [1m 1/12] [0m] [32m] [Progress bar] [0m] [37m] [Progress bar] [0m] [1m0s] [0m] 1
9ms/step - accuracy: 0.9000 - loss: 0.5795 [Progress bar]
[Progress bar] [1m12/12] [0m] [32m]
[Progress bar] [0m] [37m] [0m] [1m0s] [0m] 2ms/step - accuracy: 0.8591 - loss: 0.5272
[Progress bar] [1m1/1] [0m] [32m] [Progress bar] [0m] [37m] [0m] [1m0s] [0m] 80m
s/step - accuracy: 0.9333 - loss: 0.5408 [Progress bar]
[Progress bar] [1m1/1] [0m] [32m]
[Progress bar] [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

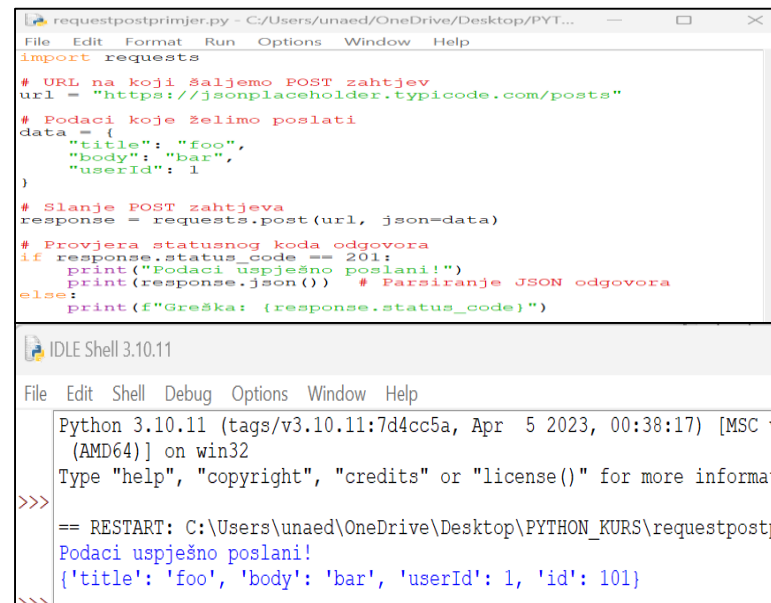
- *Requests* - HTTP paket za zahtjeve, koji omogućava slanje GET i POST zahtjeva, autentifikaciju i rad sa API-jima, preuzimanje podataka u JSON formatu i druge. Način instalacije paketa je korištenje naredbe: `pip install requests`, a primjer upotrebe je prikazan na slikama 9. i 10.



```
requestprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_K...
File Edit Format Run Options Window Help
import requests
# URL na koji šaljemo GET zahtjev
url = "https://jsonplaceholder.typicode.com/posts/1"
# Slanje GET zahtjeva
response = requests.get(url)
# Provjera statusnog koda odgovora
if response.status_code == 200:
    print("Uspjereni podaci:")
    print(response.json()) # Parsiranje JSON odgovora
else:
    print(f"Greška: {response.status_code}")

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) [MSC v.192
9 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/requestprimjer
.py ====
Uspjereni podaci:
{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occa
ecati excepturi optio reprehenderit', 'body': 'quia et suscipit\nsuscipi
t recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut
quas totam\nnostrum rerum est autem sunt rem eveniet architecto'}
```

Slika 9. Primjer upotrebe request paketa – GET zahtjev



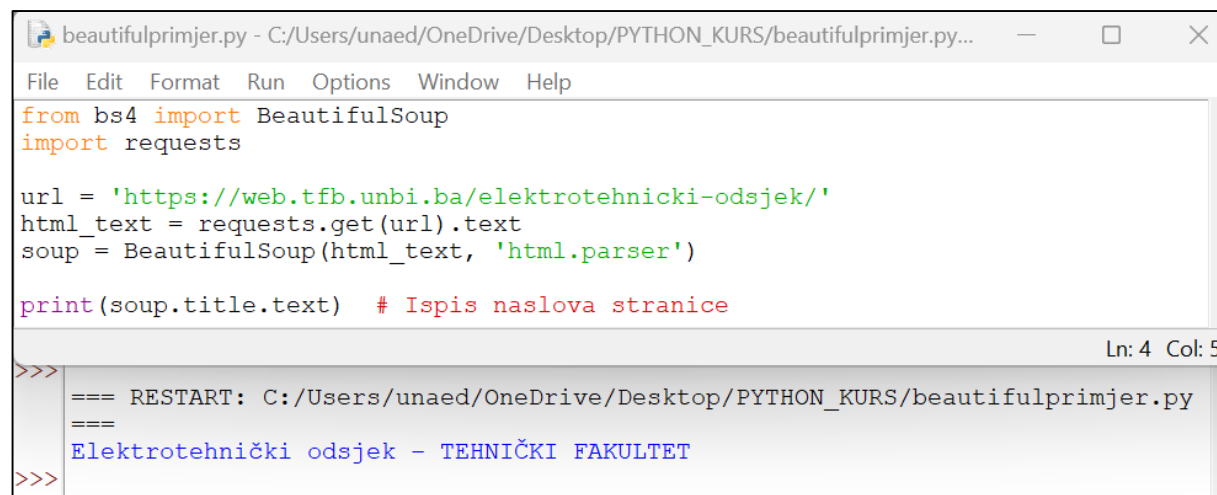
```
requestpostprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYT...
File Edit Format Run Options Window Help
import requests
# URL na koji šaljemo POST zahtjev
url = "https://jsonplaceholder.typicode.com/posts"
# Podaci koje želimo poslati
data = {
    "title": "foo",
    "body": "bar",
    "userId": 1
}
# Slanje POST zahtjeva
response = requests.post(url, json=data)
# Provjera statusnog koda odgovora
if response.status_code == 201:
    print("Podaci uspješno poslani!")
    print(response.json()) # Parsiranje JSON odgovora
else:
    print(f"Greška: {response.status_code}")

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) [MSC v
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informat
>>>
==== RESTART: C:\Users\unaed\OneDrive\Desktop\PYTHON_KURS\requestpostp
Podaci uspješno poslani!
{'title': 'foo', 'body': 'bar', 'userId': 1, 'id': 101}
```

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.



```
beautifulprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/beautifulprimjer.py...
File Edit Format Run Options Window Help
from bs4 import BeautifulSoup
import requests

url = 'https://web.tfb.unbi.ba/elektrotehnicki-odsjek/'
html_text = requests.get(url).text
soup = BeautifulSoup(html_text, 'html.parser')

print(soup.title.text) # Ispis naslova stranice

Ln: 4 Col: 5

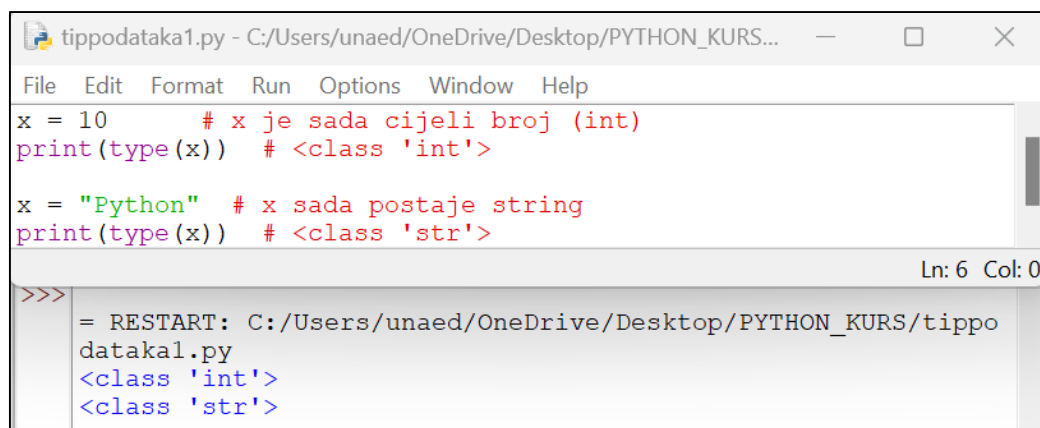
>>>
=== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/beautifulprimjer.py ===
Elektrotehnicki odsjek - TEHNIČKI FAKULTET
>>>
```

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.



```
tippodataka1.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS...
File Edit Format Run Options Window Help
x = 10      # x je sada cijeli broj (int)
print(type(x)) # <class 'int'>

x = "Python" # x sada postaje string
print(type(x)) # <class 'str'>

Ln: 6 Col: 0

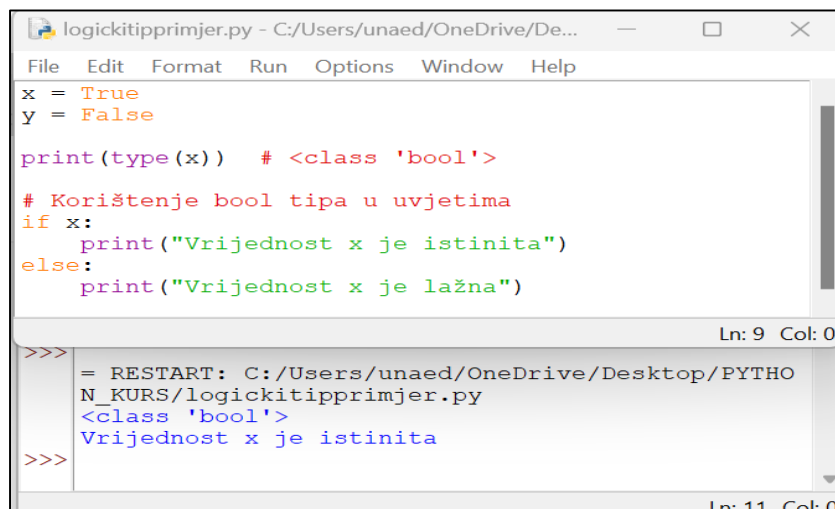
>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/tippo
datakal.py
<class 'int'>
<class 'str'>
```

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.



```
logickitipprimjer.py - C:/Users/unaed/OneDrive/De...
File Edit Format Run Options Window Help
x = True
y = False

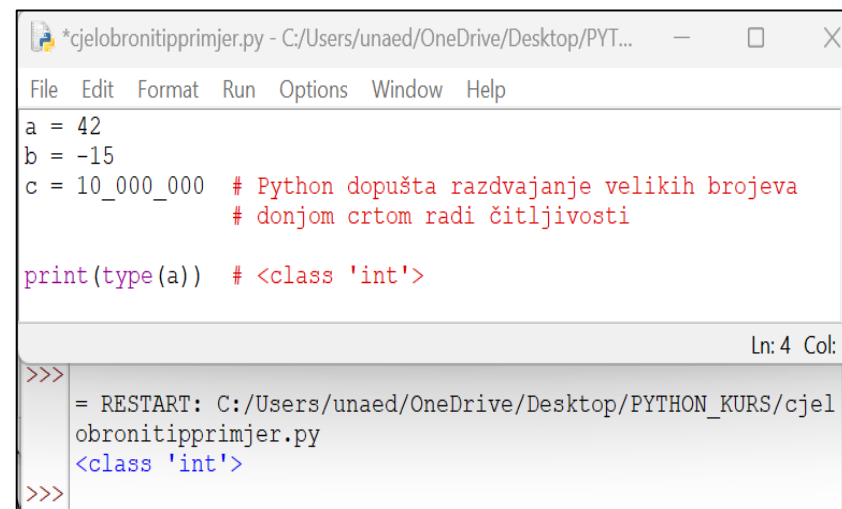
print(type(x)) # <class 'bool'>

# Korištenje bool tipa u uvjetima
if x:
    print("Vrijednost x je istinita")
else:
    print("Vrijednost x je lažna")

Ln: 9 Col: 0

>>> = RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHO
N_KURS/logickitipprimjer.py
<class 'bool'>
Vrijednost x je istinita
>>>

Ln: 11 Col: 0
```



```
*cjelobronitipprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYT...
File Edit Format Run Options Window Help
a = 42
b = -15
c = 10_000_000 # Python dopušta razdvajanje velikih brojeva
               # donjom crtom radi čitljivosti

print(type(a)) # <class 'int'>

Ln: 4 Col: 1

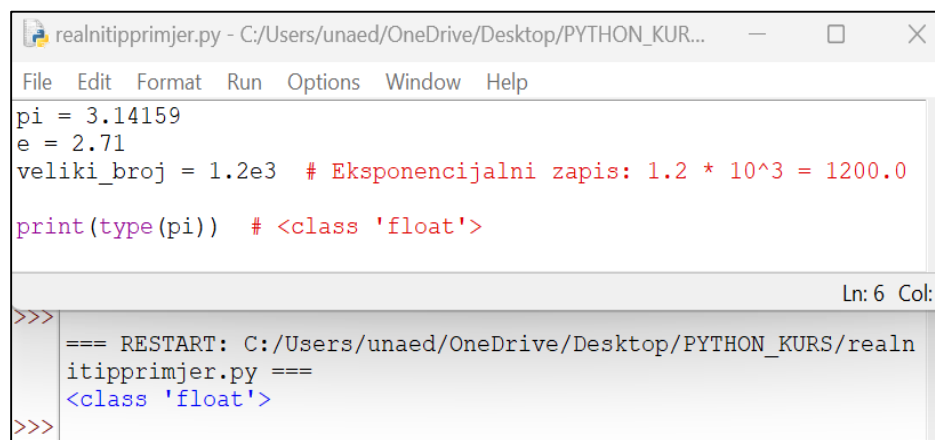
>>> = RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/cjel
obronitipprimjer.py
<class 'int'>
>>>
```

Slika 13. Primjer logičkog i cjelobrojnog tipa podataka



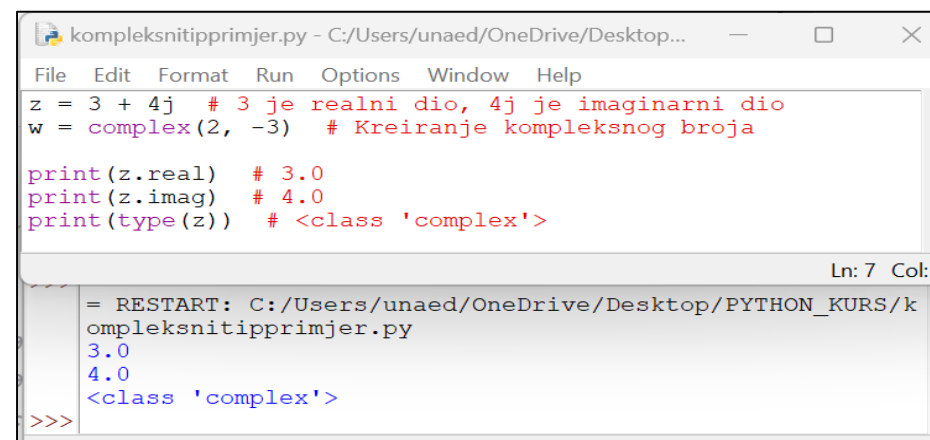
3. Tipovi podataka

- *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.



```
File Edit Format Run Options Window Help
pi = 3.14159
e = 2.71
veliki_broj = 1.2e3 # Eksponencijalni zapis: 1.2 * 10^3 = 1200.0
print(type(pi)) # <class 'float'>

Ln: 6 Col: 1
>>>
=== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/realn
itipprimjer.py ===
<class 'float'>
>>>
```



```
File Edit Format Run Options Window Help
z = 3 + 4j # 3 je realni dio, 4j je imaginarni dio
w = complex(2, -3) # Kreiranje kompleksnog broja

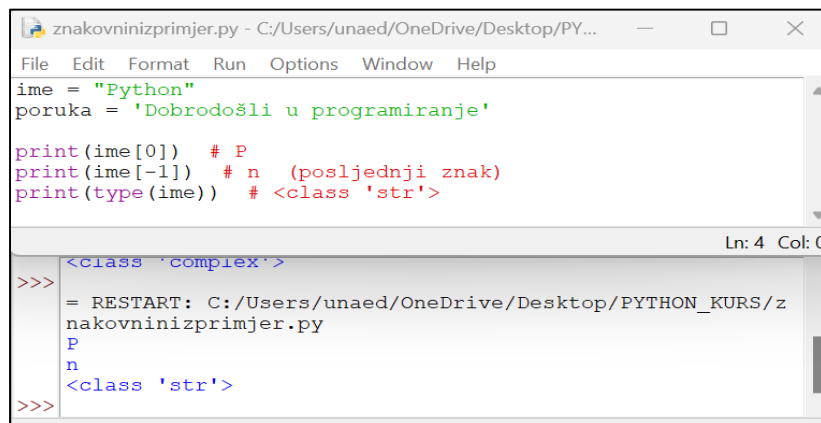
print(z.real) # 3.0
print(z.imag) # 4.0
print(type(z)) # <class 'complex'>

Ln: 7 Col: 0
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/k
ompleksnitipprimjer.py
3.0
4.0
<class 'complex'>
>>>
```

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.



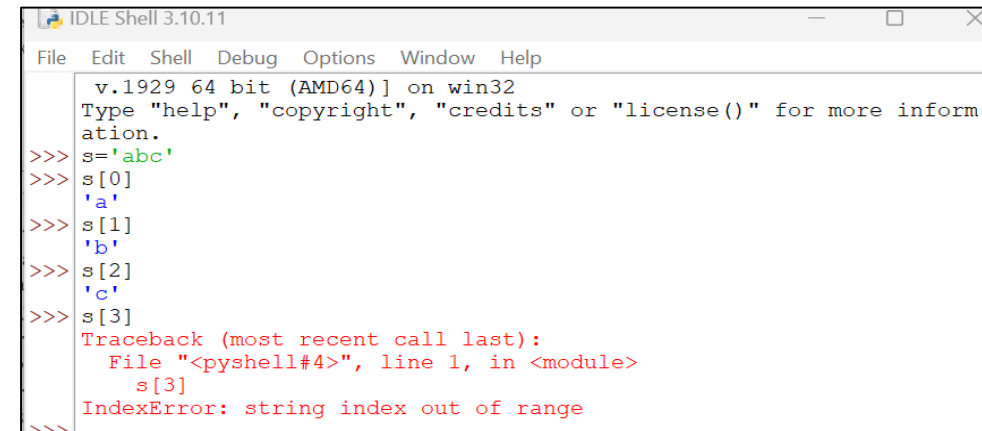
```
znakovninizprimjer.py - C:/Users/unaed/OneDrive/Desktop/PY...
File Edit Format Run Options Window Help
ime = "Python"
poruka = 'Dobrodošli u programiranje'

print(ime[0]) # P
print(ime[-1]) # n (posljednji znak)
print(type(ime)) # <class 'str'>

Ln: 4 Col: 0

>>> <class 'complex'>
>>> = RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/z
nakovninizprimjer.py
P
n
<class 'str'>
>>>
```

Slika 15. Primjer znakovnog tipa podatka

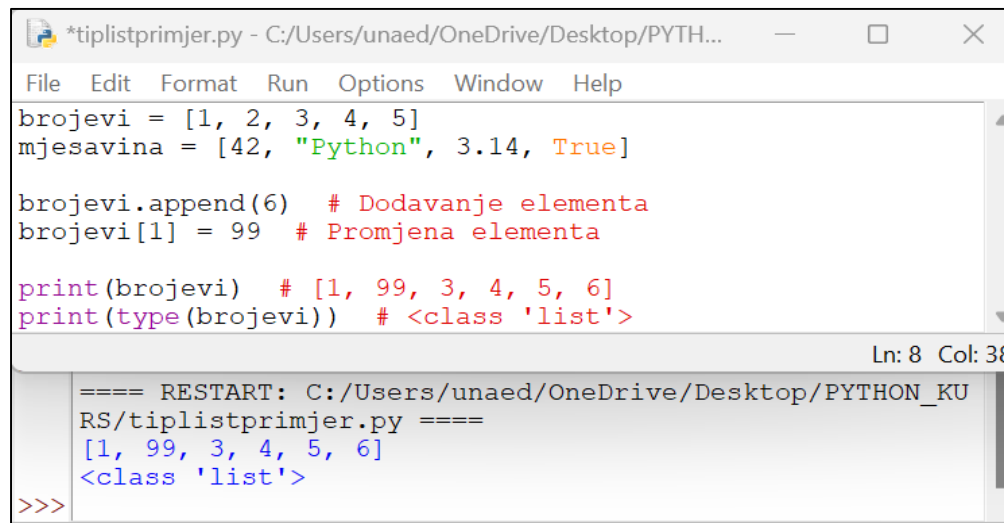


```
IDLE Shell 3.10.11
File Edit Shell Debug Options Window Help
v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inform
ation.
>>> s='abc'
>>> s[0]
'a'
>>> s[1]
'b'
>>> s[2]
'c'
>>> s[3]
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    s[3]
IndexError: string index out of range
>>>
```

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.



```
*tiplistprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTH...
File Edit Format Run Options Window Help
brojevi = [1, 2, 3, 4, 5]
mjesavina = [42, "Python", 3.14, True]

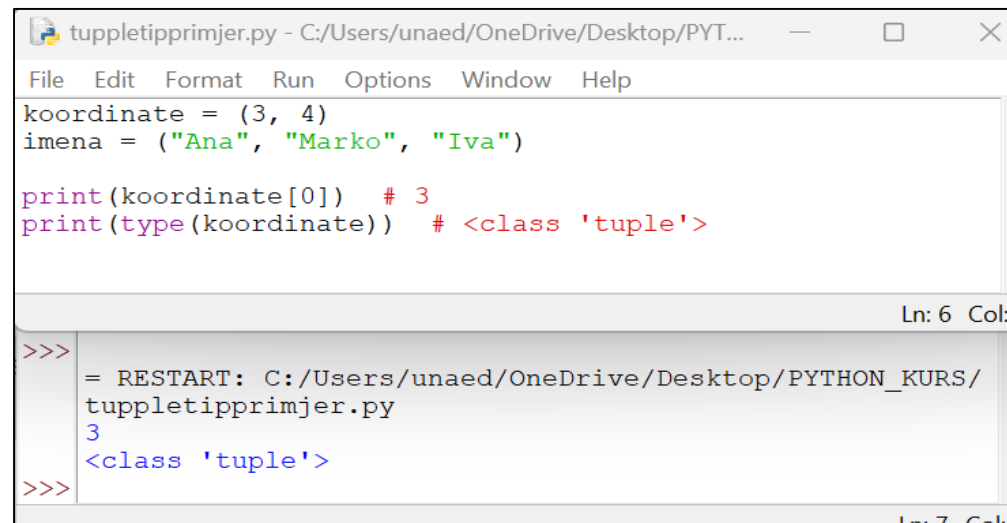
brojevi.append(6) # Dodavanje elementa
brojevi[1] = 99 # Promjena elementa

print(brojevi) # [1, 99, 3, 4, 5, 6]
print(type(brojevi)) # <class 'list'>

Ln: 8 Col: 38

==== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KU
RS/tiplistprimjer.py ====
[1, 99, 3, 4, 5, 6]
<class 'list'>
>>>
```

Slika 17. Primjer list tipa podatka



```
tuppletipprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYT...
File Edit Format Run Options Window Help
koordinata = (3, 4)
imena = ("Ana", "Marko", "Iva")

print(koordinata[0]) # 3
print(type(koordinata)) # <class 'tuple'>

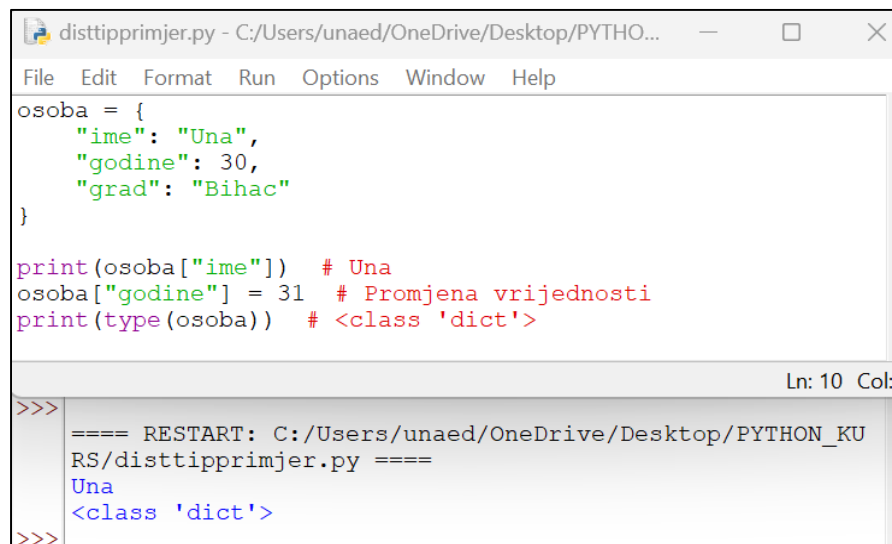
Ln: 6 Col: 10

>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/
tuppletipprimjer.py
3
<class 'tuple'>
>>>
```

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.

A screenshot of a Python IDE window titled 'disttiprimjer.py'. The code defines a dictionary 'osoba' with keys 'ime', 'godine', and 'grad'. It then prints the value of 'ime', updates 'godine' to 31, and prints the type of 'osoba'. The output in the console shows 'Una' and '<class 'dict'>'.

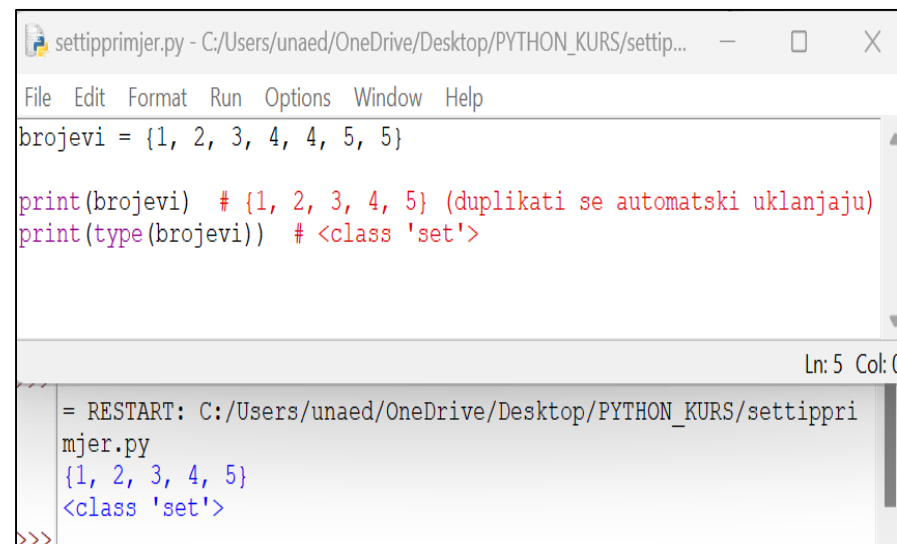
```
osoba = {
    "ime": "Una",
    "godine": 30,
    "grad": "Bihac"
}

print(osoba["ime"]) # Una
osoba["godine"] = 31 # Promjena vrijednosti
print(type(osoba)) # <class 'dict'>
```

Ln: 10 Col: 1

```
>>>
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/disttiprimjer.py =====
Una
<class 'dict'>
>>>
```

Slika 19. Primjer dic tip podatka

A screenshot of a Python IDE window titled 'settiprimjer.py'. The code defines a set 'brojevi' with duplicate values, prints the set (showing duplicates removed), and prints the type of 'brojevi'. The output in the console shows '{1, 2, 3, 4, 5}' and '<class 'set'>'.

```
brojevi = {1, 2, 3, 4, 4, 5, 5}

print(brojevi) # {1, 2, 3, 4, 5} (duplikati se automatski uklanjaju)
print(type(brojevi)) # <class 'set'>
```

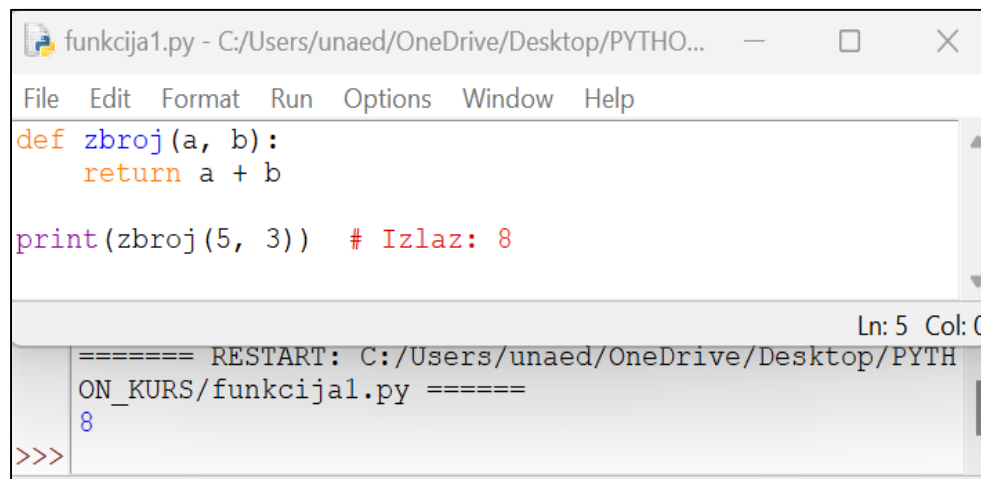
Ln: 5 Col: 1

```
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/settiprimjer.py
{1, 2, 3, 4, 5}
<class 'set'>
>>>
```

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.

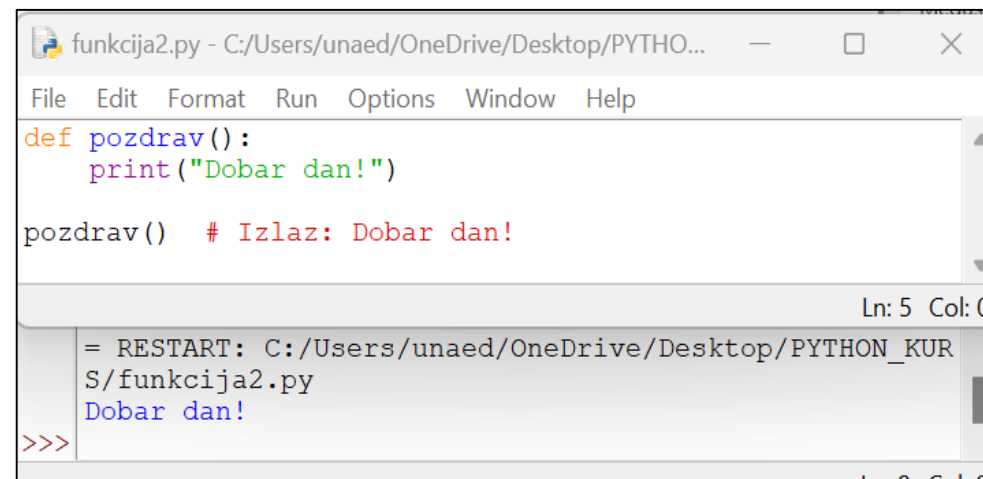


```
def zbroj(a, b):  
    return a + b  
  
print(zbroj(5, 3)) # Izlaz: 8
```

Ln: 5 Col: 0

```
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcija1.py =====  
8  
>>>
```

Slika 21. Primjer standardne funkcije sa def



```
def pozdrav():  
    print("Dobar dan!")  
  
pozdrav() # Izlaz: Dobar dan!
```

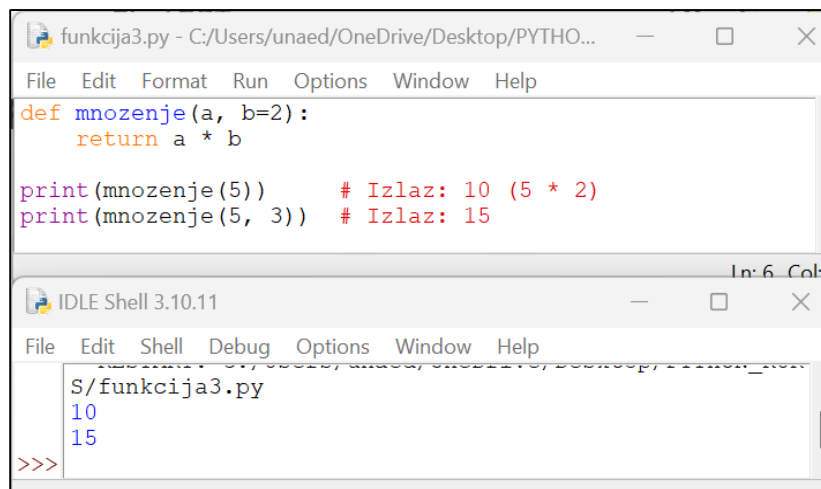
Ln: 5 Col: 0

```
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcija2.py  
Dobar dan!  
>>>
```

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.

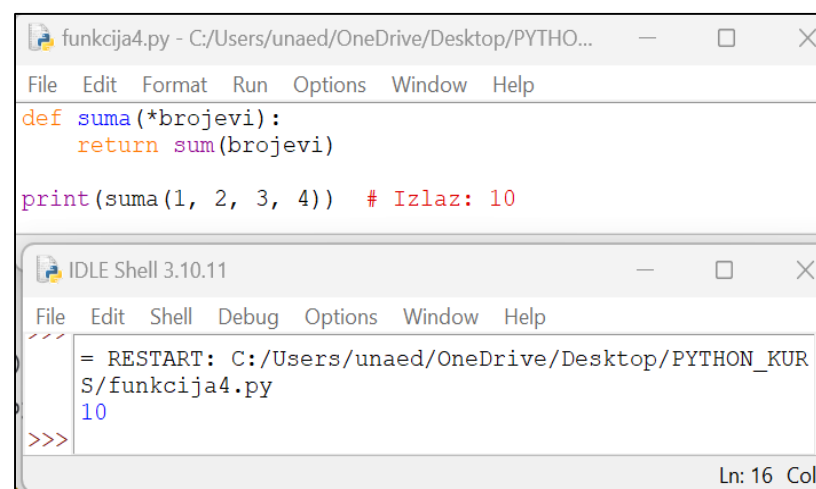


The screenshot shows a Python IDLE window titled 'funkcija3.py'. The code defines a function 'mnozenje' with two parameters, 'a' and 'b', where 'b' has a default value of 2. Below the function definition, there are two print statements: 'print(mnozenje(5))' and 'print(mnozenje(5, 3))'. Comments next to the print statements indicate the output: '# Izlaz: 10 (5 * 2)' and '# Izlaz: 15'. Below the code editor, the IDLE Shell window shows the execution results: '10' and '15' on separate lines, with the prompt '>>>' at the bottom.

```
def mnozenje(a, b=2):  
    return a * b  
  
print(mnozenje(5))      # Izlaz: 10 (5 * 2)  
print(mnozenje(5, 3))  # Izlaz: 15
```

```
>>> 10  
>>> 15  
>>>
```

Slika 23. *Funkcije sa podrazumijevanim vrijednostima parametara*



The screenshot shows a Python IDLE window titled 'funkcija4.py'. The code defines a function 'suma' with a single parameter '*brojevi'. Below the function definition, there is a print statement: 'print(suma(1, 2, 3, 4))'. A comment next to the print statement indicates the output: '# Izlaz: 10'. Below the code editor, the IDLE Shell window shows the execution results: '10' on a line, with the prompt '>>>' at the bottom. The shell window also shows a 'RESTART' message at the top.

```
def suma(*brojevi):  
    return sum(brojevi)  
  
print(suma(1, 2, 3, 4))  # Izlaz: 10
```

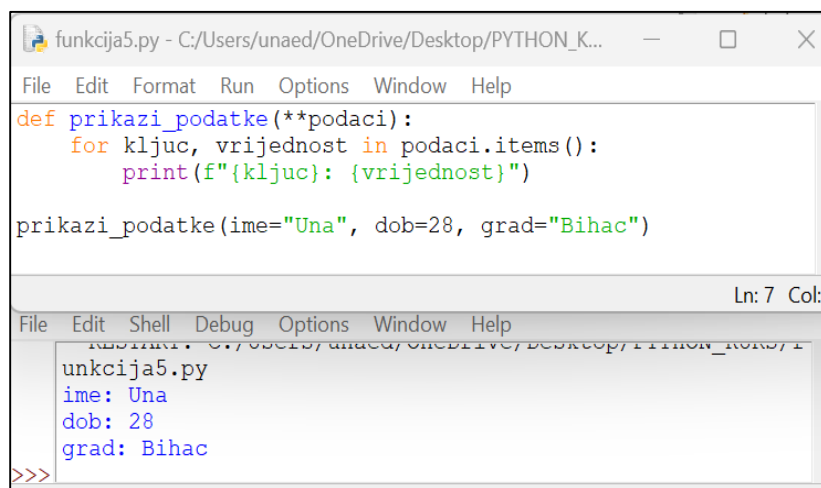
```
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KUR  
S/funkcija4.py  
10  
>>>
```

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



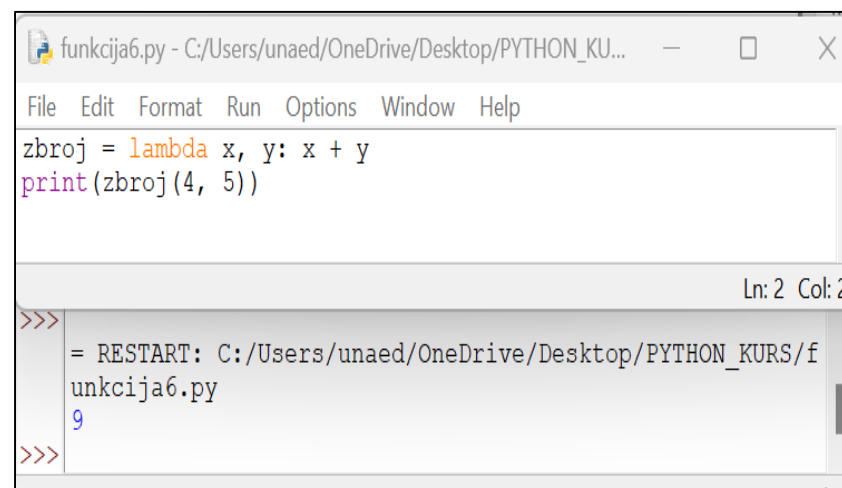
4. Funkcije

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



```
funkcija5.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_K...  
File Edit Format Run Options Window Help  
def prikazi_podatke(**podaci):  
    for kljuc, vrijednost in podaci.items():  
        print(f"{kljuc}: {vrijednost}")  
  
prikazi_podatke(ime="Una", dob=28, grad="Bihac")  
  
Ln: 7 Col: 1  
File Edit Shell Debug Options Window Help  
unkcija5.py  
ime: Una  
dob: 28  
grad: Bihac  
>>>
```

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



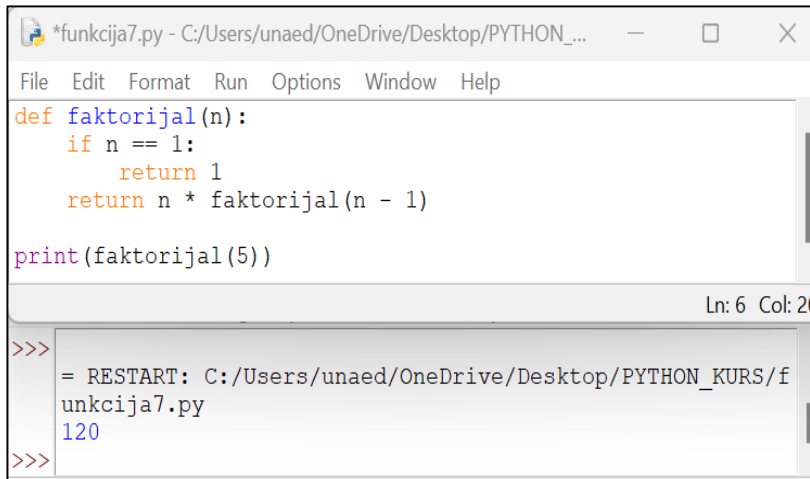
```
funkcija6.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KU...  
File Edit Format Run Options Window Help  
zbroj = lambda x, y: x + y  
print(zbroj(4, 5))  
  
Ln: 2 Col: 2  
>>>  
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/f  
unkcija6.py  
9  
>>>
```

Slika 26. Lambda (anonimne) funkcije



4. Funkcije

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

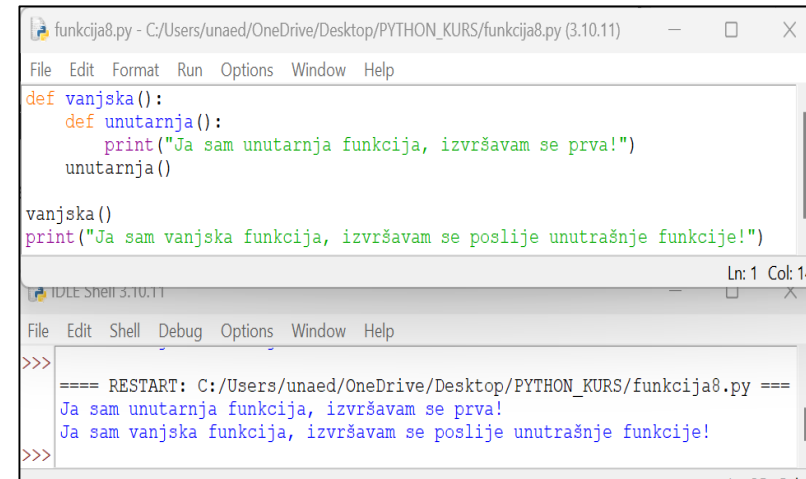


```
*funkcija7.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_...
File Edit Format Run Options Window Help
def faktorijal(n):
    if n == 1:
        return 1
    return n * faktorijal(n - 1)
print(faktorijal(5))

Ln: 6 Col: 20

>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcija7.py
120
>>>
```

Slika 27. Rekurzivna funkcija



```
funkcija8.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcija8.py (3.10.11)
File Edit Format Run Options Window Help
def vanjska():
    def unutarnja():
        print("Ja sam unutarnja funkcija, izvršavam se prva!")
    unutarnja()

vanjska()
print("Ja sam vanjska funkcija, izvršavam se poslije unutrašnje funkcije!")

Ln: 1 Col: 14

IDLE Shell 3.10.11
File Edit Shell Debug Options Window Help
>>>
==== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcija8.py ====
Ja sam unutarnja funkcija, izvršavam se prva!
Ja sam vanjska funkcija, izvršavam se poslije unutrašnje funkcije!
>>>
```

Slika 28. Funkcija unutar funkcije



4. Funkcije

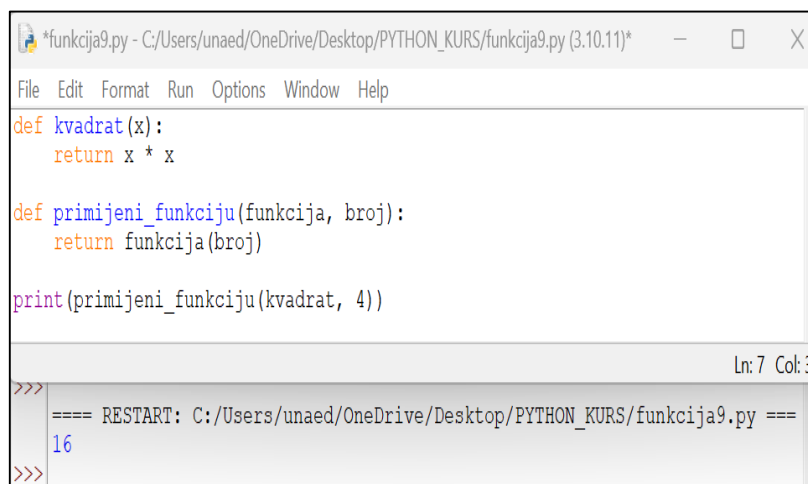
```
ugnijezdenafunkcijaKalkulator.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KU...
File Edit Format Run Options Window Help
def kalkulator(operacija):
    """Glavna funkcija koja vraća odgovarajuću matematičku operaciju."""
    def zbroj(a, b):
        return a + b
    def oduzmi(a, b):
        return 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 # Nevažuća 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.*



```
*funkcija9.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcija9.py (3.10.11)*
File Edit Format Run Options Window Help
def kvadrat(x):
    return x * x

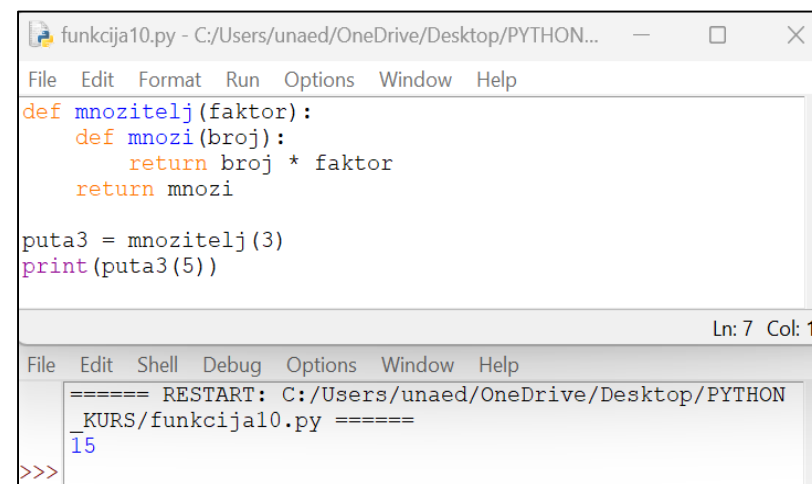
def primijeni_funkciju(funkcija, broj):
    return funkcija(broj)

print(primijeni_funkciju(kvadrat, 4))

Ln: 7 Col: 3

>>>
==== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcija9.py ====
16
>>>
```

Slika 30. *Funkcija kao argument druge funkcije*



```
funkcija10.py - C:/Users/unaed/OneDrive/Desktop/PYTHON...
File Edit Format Run Options Window Help
def mnozitelj(faktor):
    def mnozi(broj):
        return broj * faktor
    return mnozi

puta3 = mnozitelj(3)
print(puta3(5))

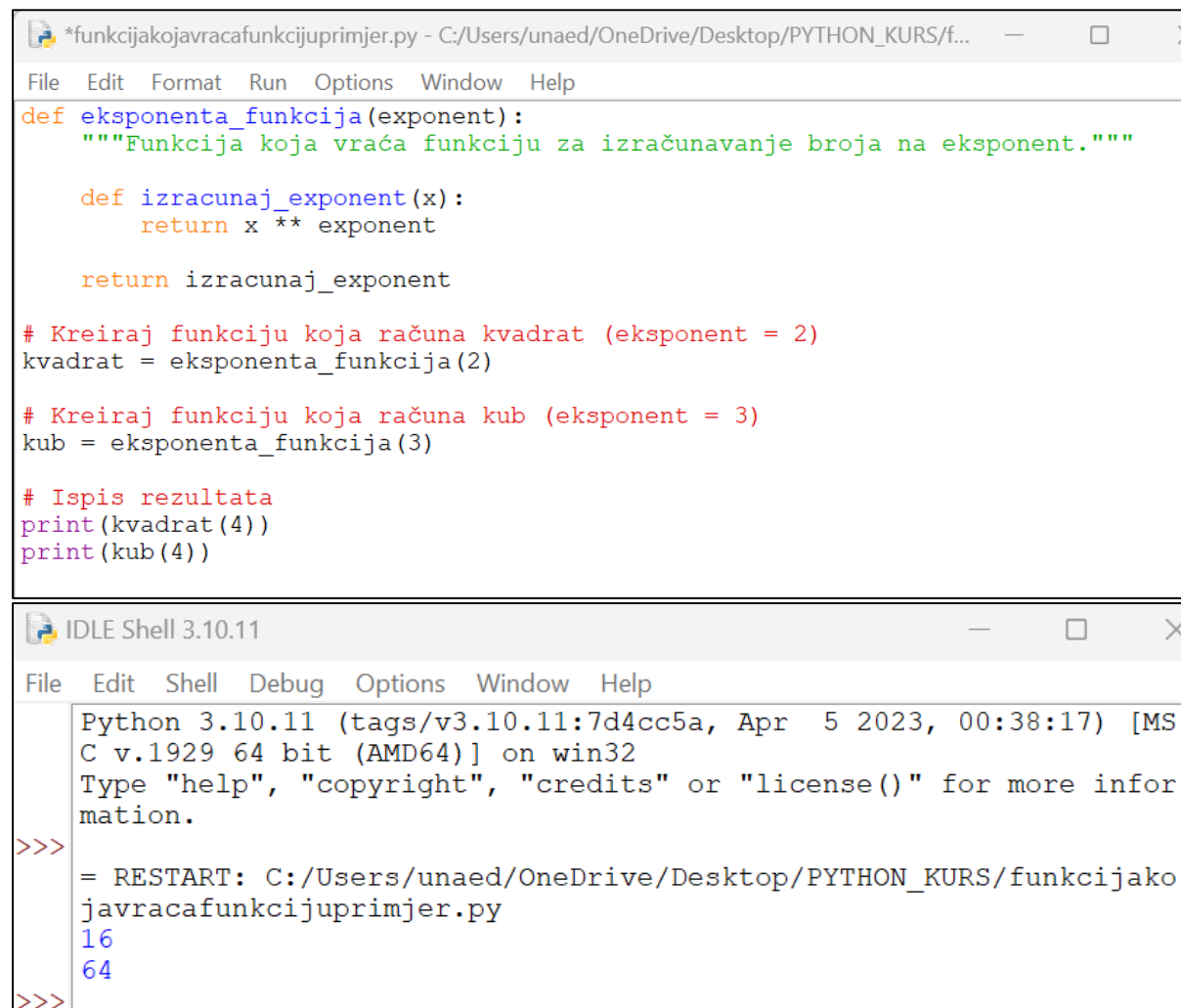
Ln: 7 Col: 17

File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcija10.py =====
15
>>>
```

Slika 31. *Funkcija koja vraća drugu funkciju*



4. Funkcije



The screenshot displays two windows from a Python IDE. The top window, titled '*funkcijakojavracafunkcijuprimjer.py', contains the following Python code:

```
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))
```

The bottom window, titled 'IDLE Shell 3.10.11', shows the execution output:

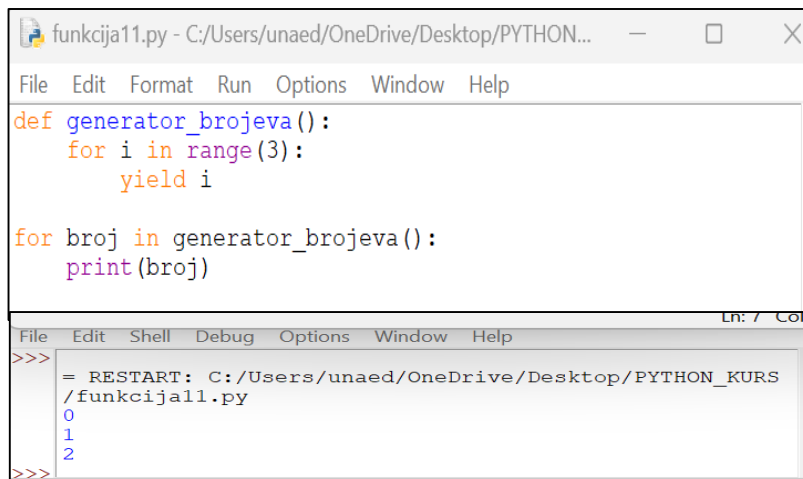
```
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr 5 2023, 00:38:17) [MS  
C v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more infor  
mation.  
>>>  
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/funkcijako  
javracafunkcijuprimjer.py  
16  
64  
>>>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

Slika 32. Funkcija koja vraća drugu funkciju



4. Funkcije

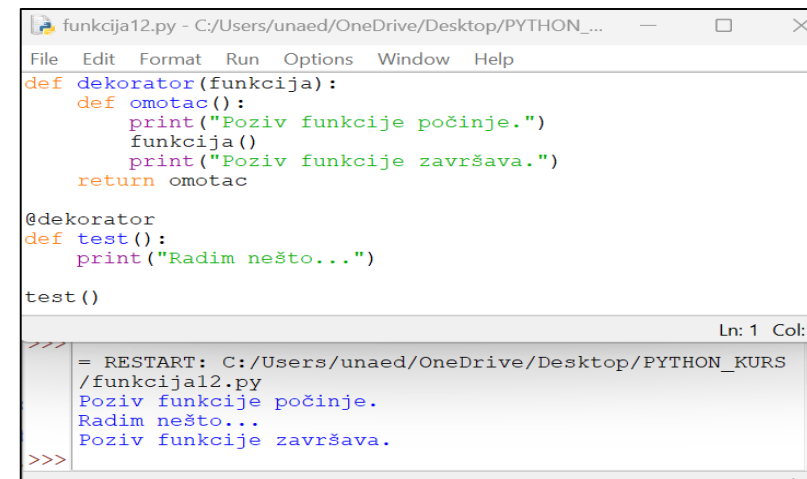
- *Generator funkcije (yield)* - umjesto return, koriste yield za stvaranje iteratora.
- *Funkcije dekoratori* - funkcije koje mijenjaju ponašanje drugih funkcija.



```
def generator_brojeva():  
    for i in range(3):  
        yield i  
  
for broj in generator_brojeva():  
    print(broj)
```

```
>>>  
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS  
/funkcija11.py  
0  
1  
2  
>>>
```

Slika 33. Generator funkcija



```
def dekorator(funkcija):  
    def omotac():  
        print("Poziv funkcije počinje.")  
        funkcija()  
        print("Poziv funkcije završava.")  
    return omotac  
  
@dekorator  
def test():  
    print("Radim nešto...")  
  
test()
```

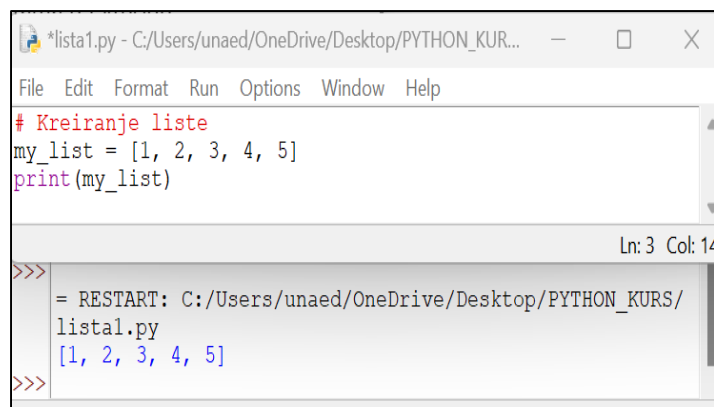
```
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS  
/funkcija12.py  
Poziv funkcije počinje.  
Radim nešto...  
Poziv funkcije završava.  
>>>
```

Slika 34. Funkcija dekorator



5. Liste

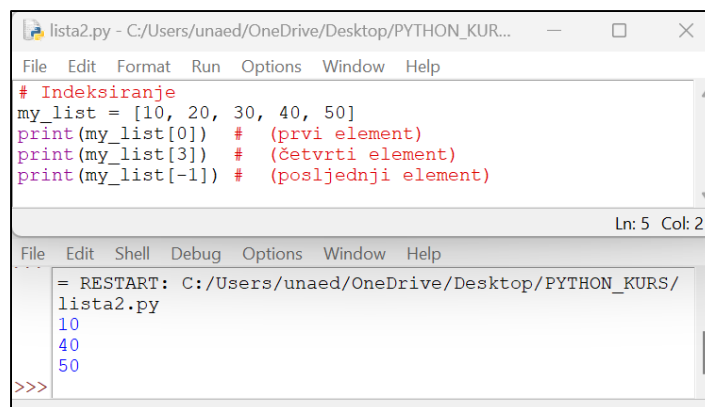
- Lista se može kreirati pomoću uglatih zagrada [], a elementi unutar liste su odvojeni zarezom.
- Indeksiranje elemenata unutar liste počinje od 0.
- Elementi unutar liste se mogu mijenjati.



```
File Edit Format Run Options Window Help
# Kreiranje liste
my_list = [1, 2, 3, 4, 5]
print(my_list)

Ln: 3 Col: 14

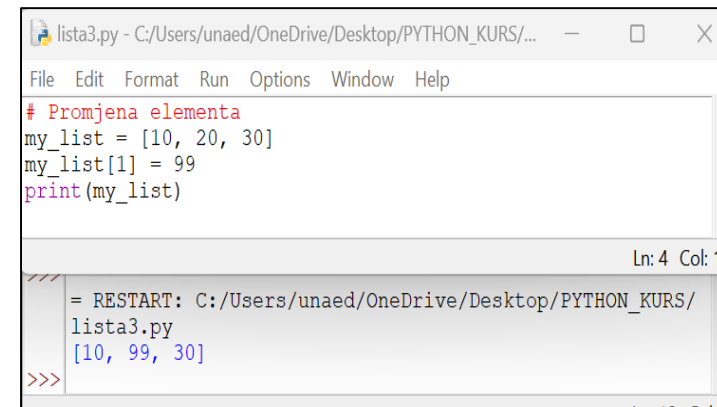
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/
lista1.py
[1, 2, 3, 4, 5]
```



```
File Edit Format Run Options Window Help
# Indeksiranje
my_list = [10, 20, 30, 40, 50]
print(my_list[0]) # (prvi element)
print(my_list[3]) # (četvrti element)
print(my_list[-1]) # (posljednji element)

Ln: 5 Col: 21

File Edit Shell Debug Options Window Help
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/
lista2.py
10
40
50
```



```
File Edit Format Run Options Window Help
# Promjena elementa
my_list = [10, 20, 30]
my_list[1] = 99
print(my_list)

Ln: 4 Col: 16

= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/
lista3.py
[10, 99, 30]
```

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.

```
lista4.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/...
File Edit Format Run Options Window Help

# Dodavanje elemenata
my_list = [10, 20, 30]

# append - dodaje na kraj
my_list.append(40)
print(my_list)

# insert - dodaje na određeni indeks
my_list.insert(1, 15)
print(my_list)

# extend - dodaje elemente iz druge liste
my_list.extend([50, 60])
print(my_list)

Ln: 6 Col: 16

= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/
lista4.py
[10, 20, 30, 40]
[10, 15, 20, 30, 40]
[10, 15, 20, 30, 40, 50, 60]
>>>
```

```
lista5.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS...
File Edit Format Run Options Window Help

# Uklanjanje elemenata
my_list = [10, 20, 30, 40, 50]

# remove - uklanja prvi odgovarajući element
my_list.remove(30)
print(my_list) # Ispis: [10, 20, 40, 50]

# pop - uklanja element na određenom indeksu (ili s kraja)
my_list.pop(1)
print(my_list) # Ispis: [10, 40, 50]

# clear - uklanja sve elemente
my_list.clear()
print(my_list) # Ispis: []

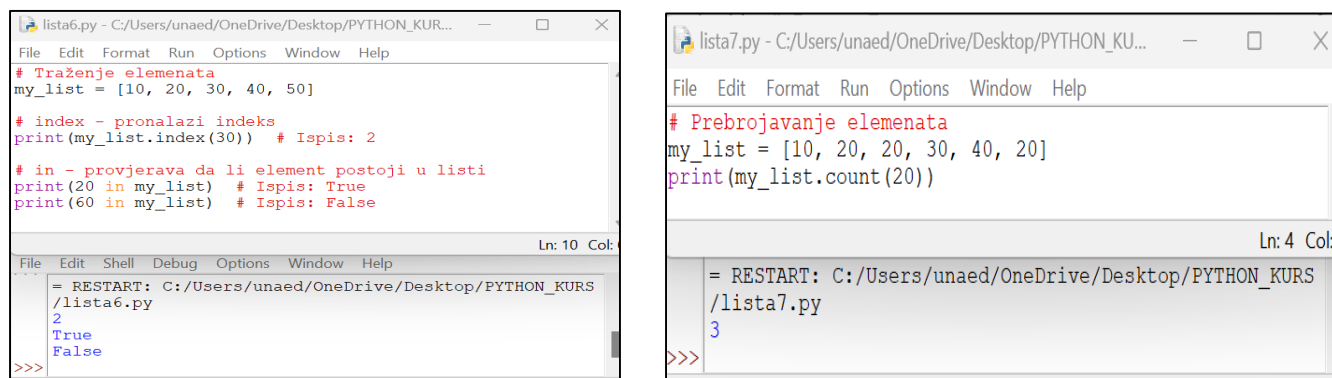
Ln: 15 Col:

= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/
lista5.py
[10, 20, 40, 50]
[10, 40, 50]
[]
>>>
```

Slika 36. Dodavanje i uklanjanje elemenata u listama



5. Liste



```
# lista6.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KUR...
File Edit Format Run Options Window Help
# Traženje elemenata
my_list = [10, 20, 30, 40, 50]

# index - pronalazi indeks
print(my_list.index(30)) # Ispis: 2

# in - provjerava da li element postoji u listi
print(20 in my_list) # Ispis: True
print(60 in my_list) # Ispis: False

Ln: 10 Col: 1

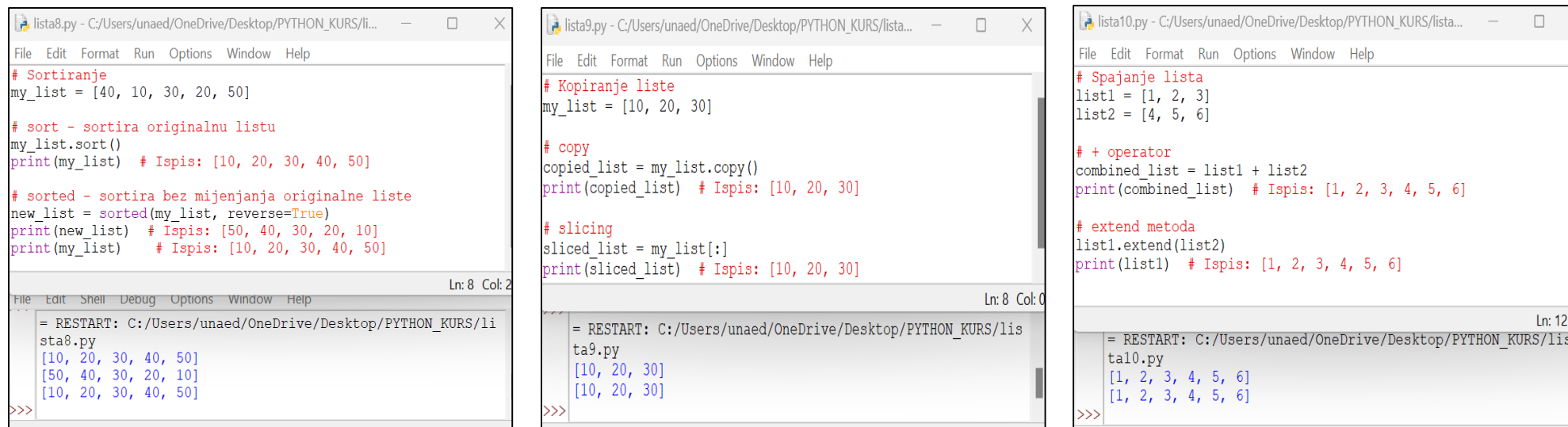
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS
/lista6.py
2
True
False
>>>

# lista7.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KU...
File Edit Format Run Options Window Help
# Prebrojavanje elemenata
my_list = [10, 20, 20, 30, 40, 20]
print(my_list.count(20))

Ln: 4 Col: 1

= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS
/lista7.py
3
>>>
```

Slika 37. Traženje i prebrojavanje elemenata u listama



```
# lista8.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/li...
File Edit Format Run Options Window Help
# Sortiranje
my_list = [40, 10, 30, 20, 50]

# sort - sortira originalnu listu
my_list.sort()
print(my_list) # Ispis: [10, 20, 30, 40, 50]

# sorted - sortira bez mijenjanja originalne liste
new_list = sorted(my_list, reverse=True)
print(new_list) # Ispis: [50, 40, 30, 20, 10]
print(my_list) # Ispis: [10, 20, 30, 40, 50]

Ln: 8 Col: 2

= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/li
sta8.py
[10, 20, 30, 40, 50]
[50, 40, 30, 20, 10]
[10, 20, 30, 40, 50]
>>>

# lista9.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/li...
File Edit Format Run Options Window Help
# Kopiranje liste
my_list = [10, 20, 30]

# copy
copied_list = my_list.copy()
print(copied_list) # Ispis: [10, 20, 30]

# slicing
sliced_list = my_list[:]
print(sliced_list) # Ispis: [10, 20, 30]

Ln: 8 Col: 0

= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lis
ta9.py
[10, 20, 30]
[10, 20, 30]
>>>

# lista10.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lis...
File Edit Format Run Options Window Help
# Spajanje lista
list1 = [1, 2, 3]
list2 = [4, 5, 6]

# + operator
combined_list = list1 + list2
print(combined_list) # Ispis: [1, 2, 3, 4, 5, 6]

# extend metoda
list1.extend(list2)
print(list1) # Ispis: [1, 2, 3, 4, 5, 6]

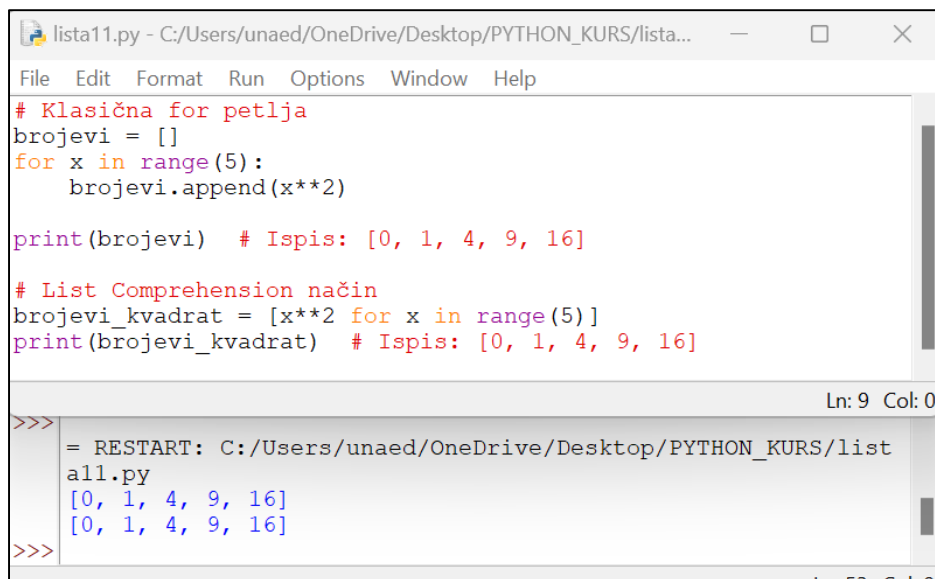
Ln: 12 Col: 0

= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lis
ta10.py
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
>>>
```

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.



```
lista11.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista...
File Edit Format Run Options Window Help
# Klasična for petlja
brojevi = []
for x in range(5):
    brojevi.append(x**2)

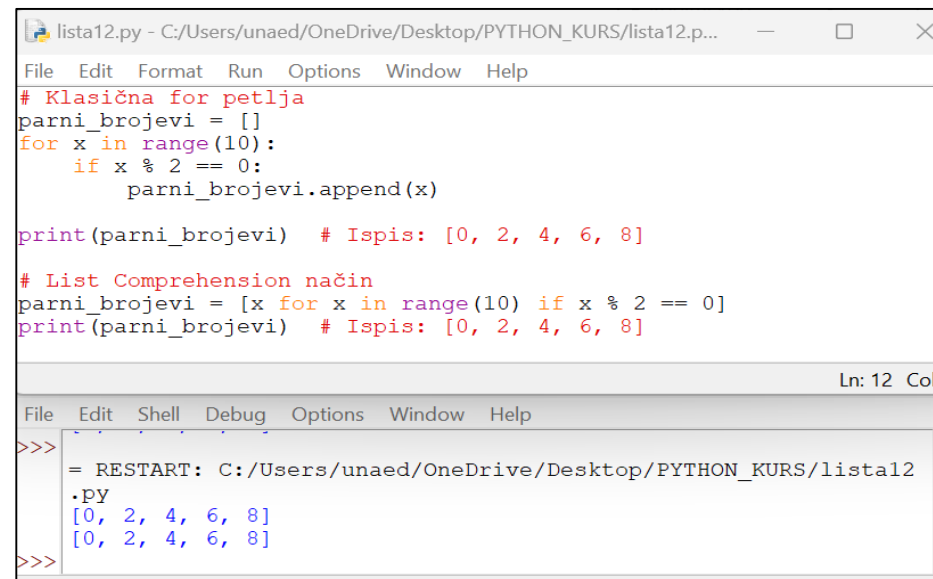
print(brojevi) # Ispis: [0, 1, 4, 9, 16]

# List Comprehension način
brojevi_kvadrat = [x**2 for x in range(5)]
print(brojevi_kvadrat) # Ispis: [0, 1, 4, 9, 16]

Ln: 9 Col: 0

>>> = RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/list
all.py
[0, 1, 4, 9, 16]
[0, 1, 4, 9, 16]
>>>
```

Slika39. Primjer generiranja liste brojeva



```
lista12.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista12.p...
File Edit Format Run Options Window Help
# Klasična for petlja
parni_brojevi = []
for x in range(10):
    if x % 2 == 0:
        parni_brojevi.append(x)

print(parni_brojevi) # Ispis: [0, 2, 4, 6, 8]

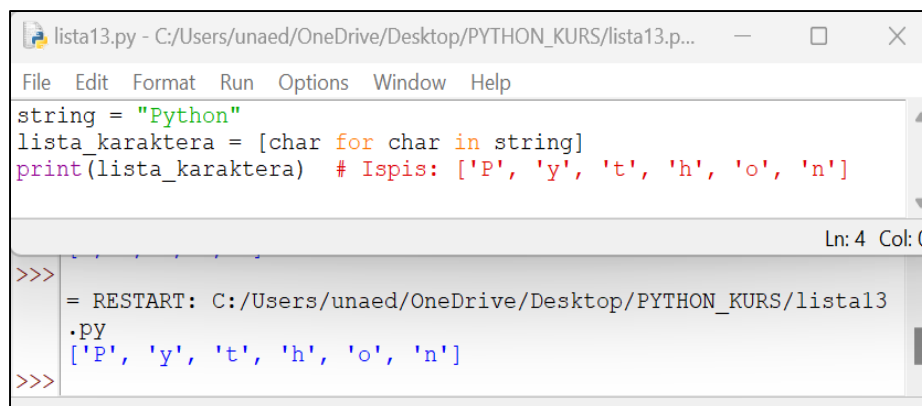
# List Comprehension način
parni_brojevi = [x for x in range(10) if x % 2 == 0]
print(parni_brojevi) # Ispis: [0, 2, 4, 6, 8]

Ln: 12 Col:

File Edit Shell Debug Options Window Help
>>> = RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista12
.py
[0, 2, 4, 6, 8]
[0, 2, 4, 6, 8]
>>>
```

Slika 40. Primjer filtriranja liste brojeva

5. Liste

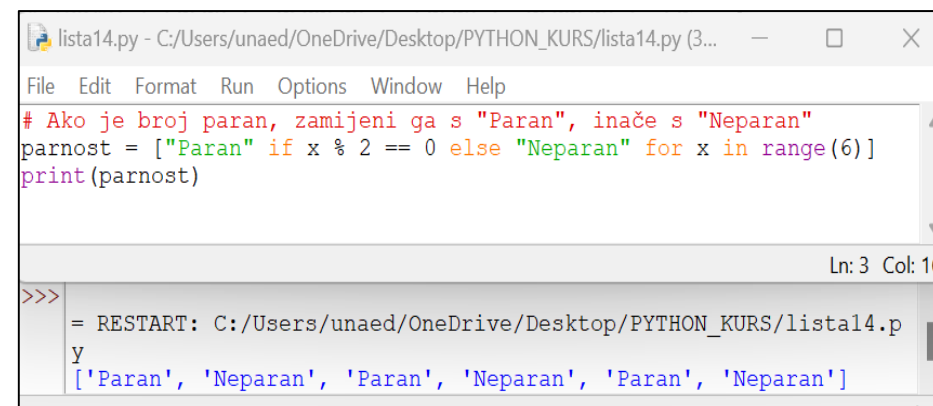


```
lista13.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista13.p...
File Edit Format Run Options Window Help
string = "Python"
lista_karaktera = [char for char in string]
print(lista_karaktera) # Ispis: ['P', 'y', 't', 'h', 'o', 'n']

Ln: 4 Col: 0

>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista13
.py
['P', 'y', 't', 'h', 'o', 'n']
>>>
```

Slika 41. *Primjer konverzije stringa u listu karaktera*

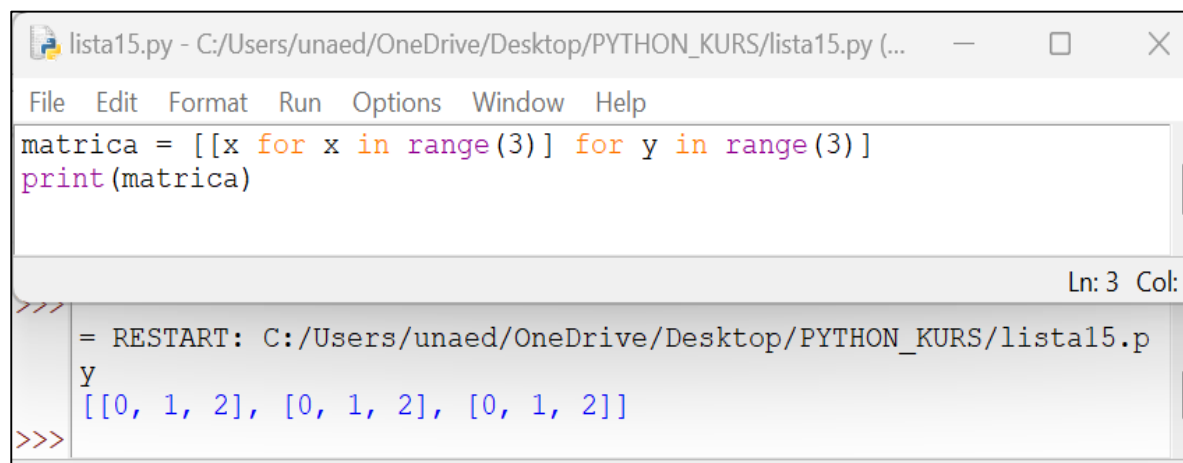


```
lista14.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista14.py (3...
File Edit Format Run Options Window Help
# Ako je broj paran, zamijeni ga s "Paran", inače s "Neparan"
parnost = ["Paran" if x % 2 == 0 else "Neparan" for x in range(6)]
print(parnost)

Ln: 3 Col: 16

>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista14.p
y
['Paran', 'Neparan', 'Paran', 'Neparan', 'Paran', 'Neparan']
```

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



```
lista15.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista15.py (...
File Edit Format Run Options Window Help
matrica = [[x for x in range(3)] for y in range(3)]
print(matrica)

Ln: 3 Col: 0

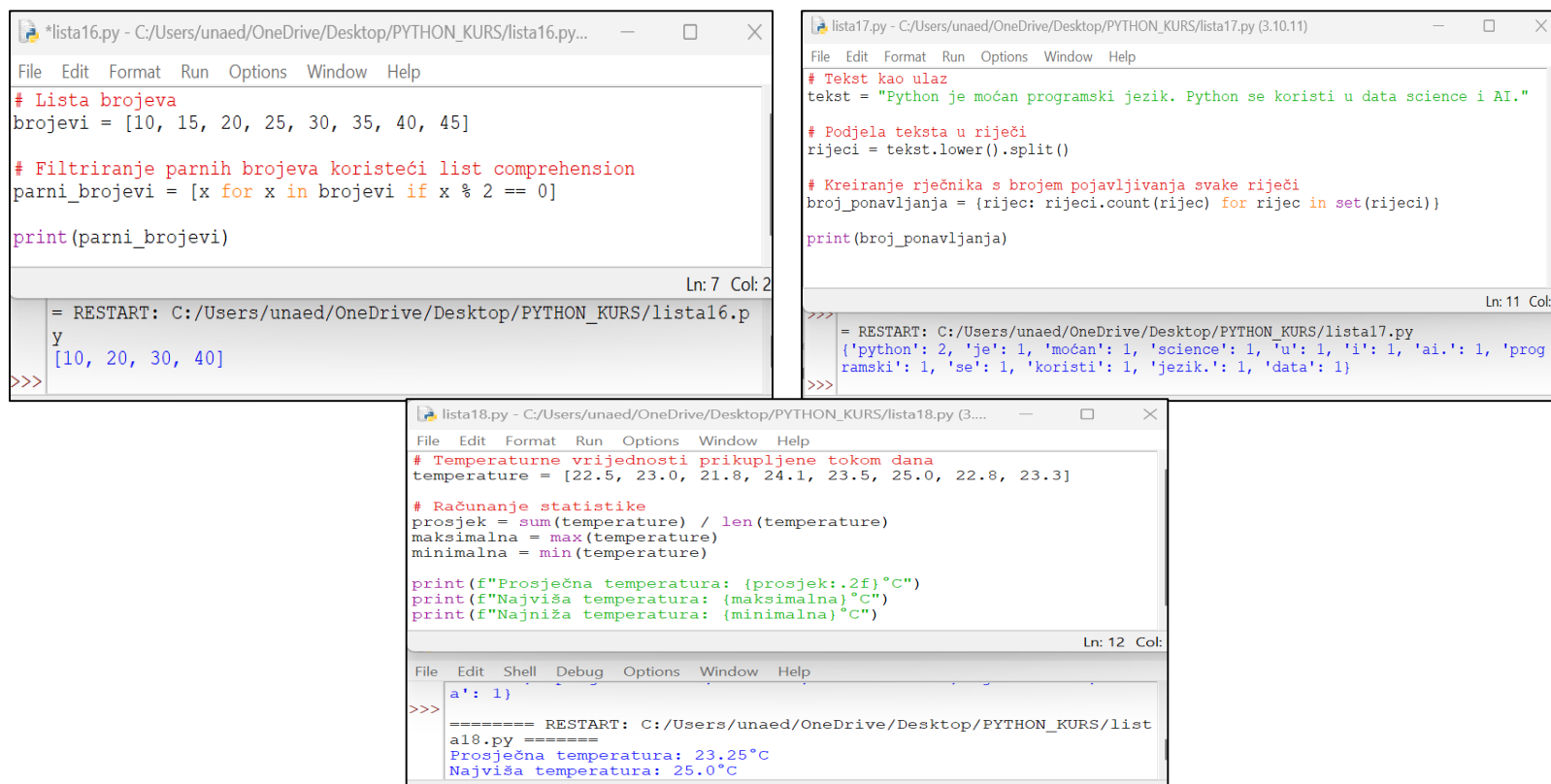
>>>
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista15.p
y
[[0, 1, 2], [0, 1, 2], [0, 1, 2]]
>>>
```

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.



```
*lista16.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista16.py...
File Edit Format Run Options Window Help
# Lista brojeva
brojevi = [10, 15, 20, 25, 30, 35, 40, 45]

# Filtriranje parnih brojeva koristeći list comprehension
parni_brojevi = [x for x in brojevi if x % 2 == 0]

print(parni_brojevi)

Ln: 7 Col: 2
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista16.p
y
[10, 20, 30, 40]
>>>
```

```
lista17.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista17.py (3.10.11)
File Edit Format Run Options Window Help
# Tekst kao ulaz
tekst = "Python je moćan programski jezik. Python se koristi u data science i AI."

# Podjela teksta u riječi
rijeci = tekst.lower().split()

# Kreiranje rječnika s brojem pojavljivanja svake riječi
broj_ponavljanja = {rijec: rijeci.count(rijec) for rijec in set(rijeci)}

print(broj_ponavljanja)

Ln: 11 Col: 0
>>> = RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista17.py
{'python': 2, 'je': 1, 'moćan': 1, 'science': 1, 'u': 1, 'i': 1, 'ai.': 1, 'prog
ramski': 1, 'se': 1, 'koristi': 1, 'jezik.': 1, 'data': 1}
>>>
```

```
lista18.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/lista18.py (3...
File Edit Format Run Options Window Help
# Temperaturne vrijednosti prikupljene tokom dana
temperature = [22.5, 23.0, 21.8, 24.1, 23.5, 25.0, 22.8, 23.3]

# Računanje statistike
prosjeck = sum(temperature) / len(temperature)
maksimalna = max(temperature)
minimalna = min(temperature)

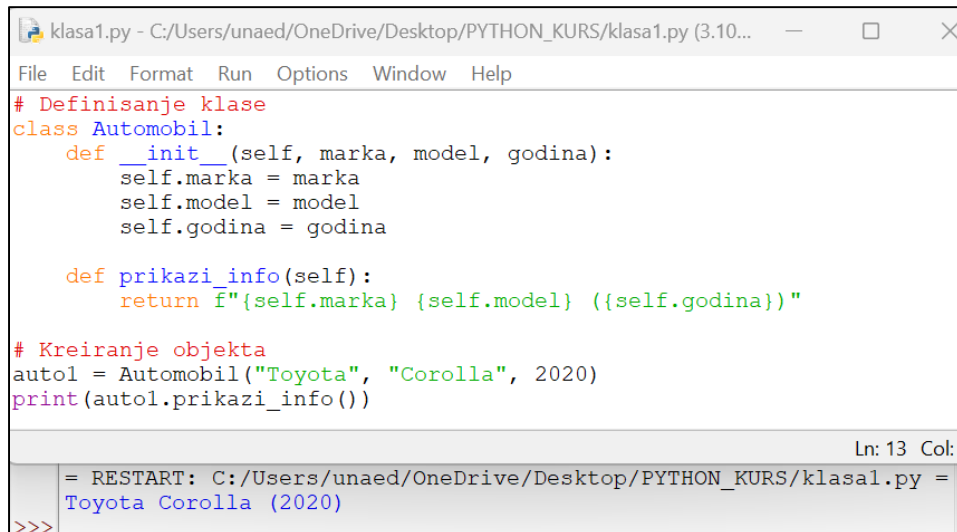
print(f"Prosječna temperatura: {prosjeck:.2f}°C")
print(f"Najviša temperatura: {maksimalna}°C")
print(f"Najniža temperatura: {minimalna}°C")

Ln: 12 Col:
File Edit Shell Debug Options Window Help
>>> a': 1}
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/list
a18.py =====
Prosječna temperatura: 23.25°C
Najviša temperatura: 25.0°C
```

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.



```
klasa1.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa1.py (3.10...
File Edit Format Run Options Window Help
# Definisanje klase
class Automobil:
    def __init__(self, marka, model, godina):
        self.marka = marka
        self.model = model
        self.godina = godina

    def prikazi_info(self):
        return f"{self.marka} {self.model} ({self.godina})"

# Kreiranje objekta
auto1 = Automobil("Toyota", "Corolla", 2020)
print(auto1.prikazi_info())

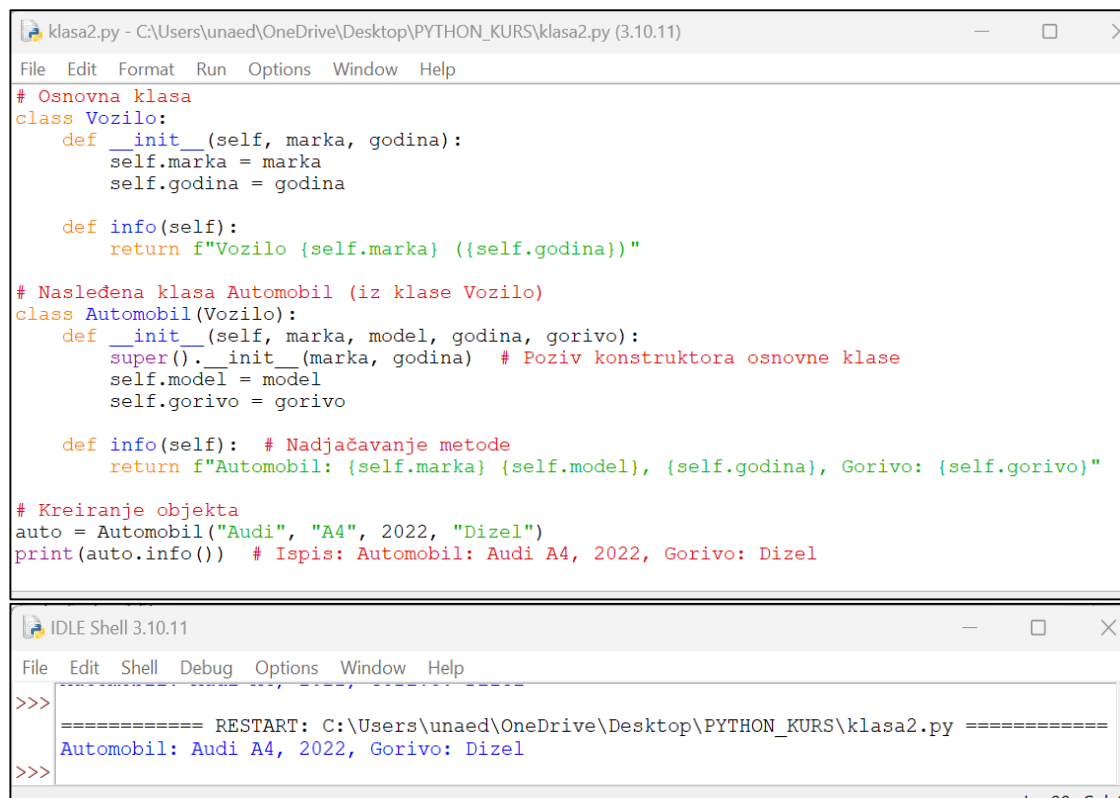
Ln: 13 Col: 1
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa1.py =
Toyota Corolla (2020)
>>>
```

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

Slika 45. *Primjer kreiranja objekta*

6. Klase

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



```
klasa2.py - C:\Users\unaed\OneDrive\Desktop\PYTHON_KURS\klasa2.py (3.10.11)
File Edit Format Run Options Window Help
# Osnovna klasa
class Vozilo:
    def __init__(self, marka, godina):
        self.marka = marka
        self.godina = godina

    def info(self):
        return f"Vozilo {self.marka} ({self.godina})"

# Nasledena klasa Automobil (iz klase Vozilo)
class Automobil(Vozilo):
    def __init__(self, marka, model, godina, gorivo):
        super().__init__(marka, godina) # Poziv konstruktora osnovne klase
        self.model = model
        self.gorivo = gorivo

    def info(self): # Nadjačavanje metode
        return f"Automobil: {self.marka} {self.model}, {self.godina}, Gorivo: {self.gorivo}"

# Kreiranje objekta
auto = Automobil("Audi", "A4", 2022, "Dizel")
print(auto.info()) # Ispis: Automobil: Audi A4, 2022, Gorivo: Dizel

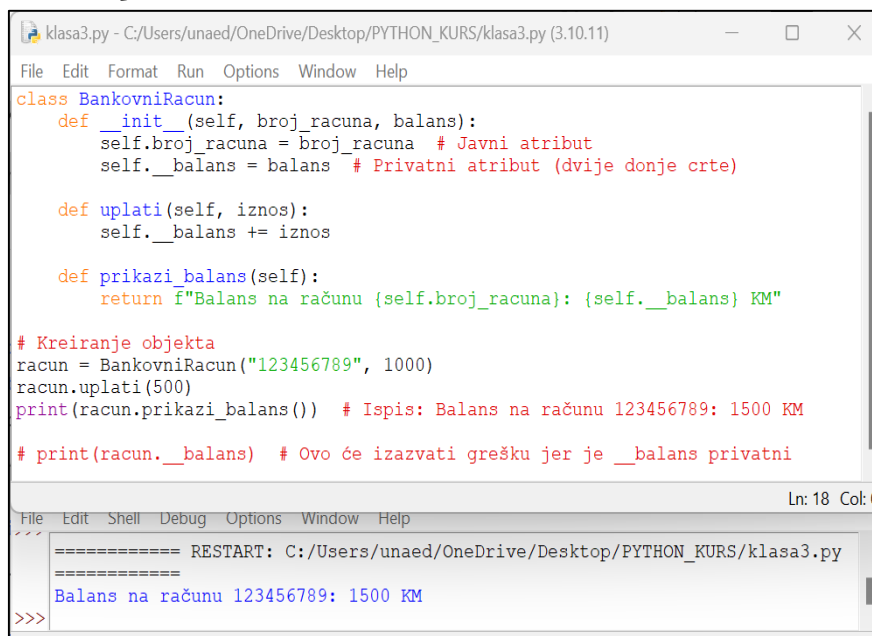
IDLE Shell 3.10.11
File Edit Shell Debug Options Window Help
>>>
===== RESTART: C:\Users\unaed\OneDrive\Desktop\PYTHON_KURS\klasa2.py =====
Automobil: Audi A4, 2022, Gorivo: Dizel
>>>
```

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

Slika 46. Primjer nasljeđivanja

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.



```
klasa3.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa3.py (3.10.11)
File Edit Format Run Options Window Help

class BankovniRacun:
    def __init__(self, broj_racuna, balans):
        self.broj_racuna = broj_racuna # Javni atribut
        self.__balans = balans # Privatni atribut (dviije donje crte)

    def uplati(self, iznos):
        self.__balans += iznos

    def prikazi_balans(self):
        return f"Balans na računu {self.broj_racuna}: {self.__balans} KM"

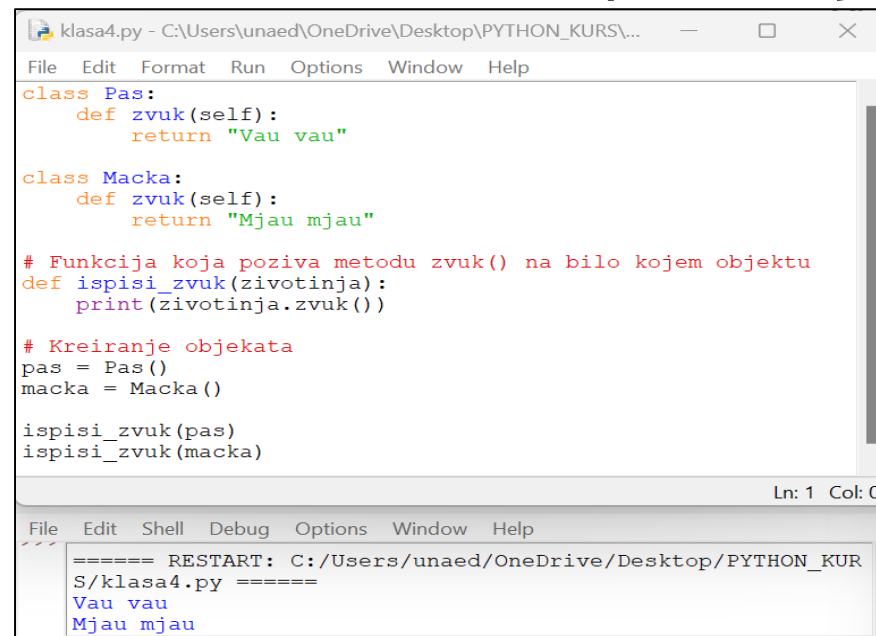
# Kreiranje objekta
racun = BankovniRacun("123456789", 1000)
racun.uplati(500)
print(racun.prikazi_balans()) # Ispis: Balans na računu 123456789: 1500 KM

# print(racun.__balans) # Ovo će izazvati grešku jer je __balans privatni

Ln: 18 Col: 0

File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa3.py
=====
Balans na računu 123456789: 1500 KM
>>>
```

Slika 47. Primjer enkapsulacije



```
klasa4.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/...
File Edit Format Run Options Window Help

class Pas:
    def zvuk(self):
        return "Vau vau"

class Macka:
    def zvuk(self):
        return "Mjau mjau"

# Funkcija koja poziva metodu zvuk() na bilo kojem objektu
def ispisi_zvuk(zivotinja):
    print(zivotinja.zvuk())

# Kreiranje objekata
pas = Pas()
macka = Macka()

ispisi_zvuk(pas)
ispisi_zvuk(macka)

Ln: 1 Col: 0

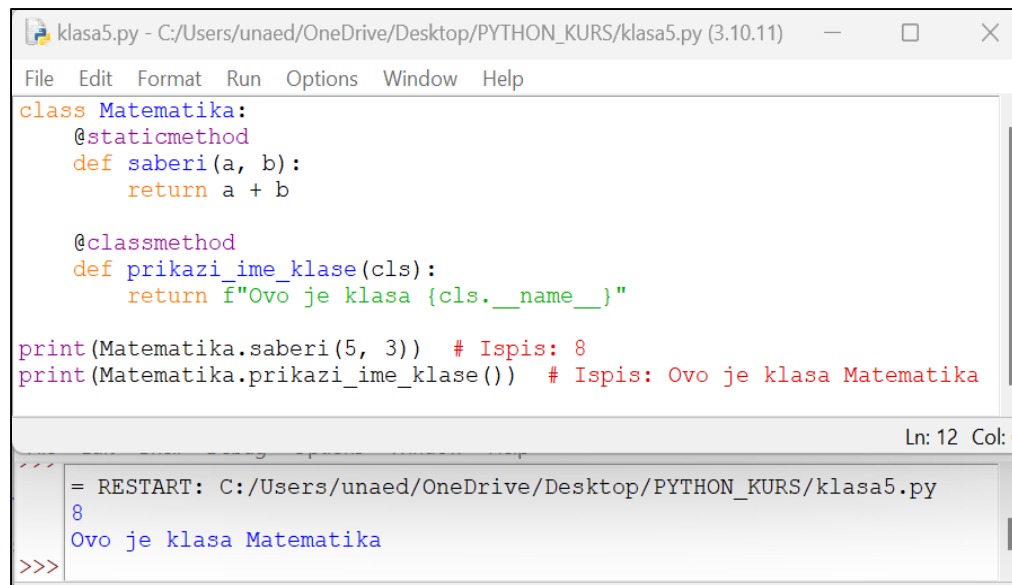
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa4.py =====
Vau vau
Mjau mjau
```

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.



```
klasa5.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa5.py (3.10.11)
File Edit Format Run Options Window Help
class Matematika:
    @staticmethod
    def saberi(a, b):
        return a + b

    @classmethod
    def prikazi_ime_klase(cls):
        return f'Ovo je klasa {cls.__name__}'

print(Matematika.saberi(5, 3)) # Ispis: 8
print(Matematika.prikazi_ime_klase()) # Ispis: Ovo je klasa Matematika

Ln: 12 Col: 0

= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa5.py
8
Ovo je klasa Matematika
>>>
```

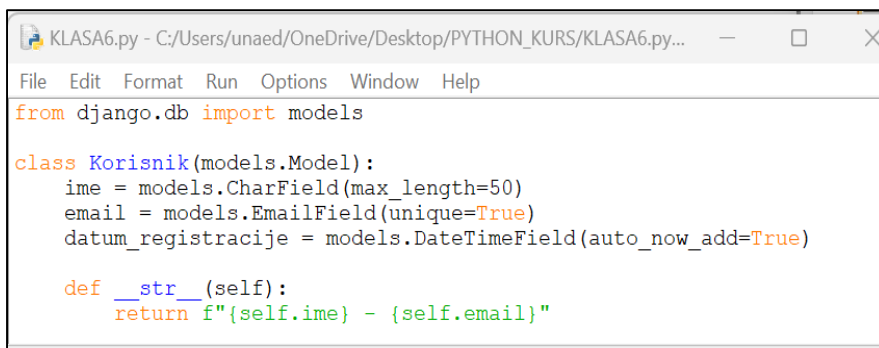
- @staticmethod označava metodu koja ne zavisi od instance (self);
- @classmethod koristi cls, što omogućava rad s klasom, a ne samo instancom.

Slika 49. *Primjer statičke metode i metode klase*



6. Klase

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

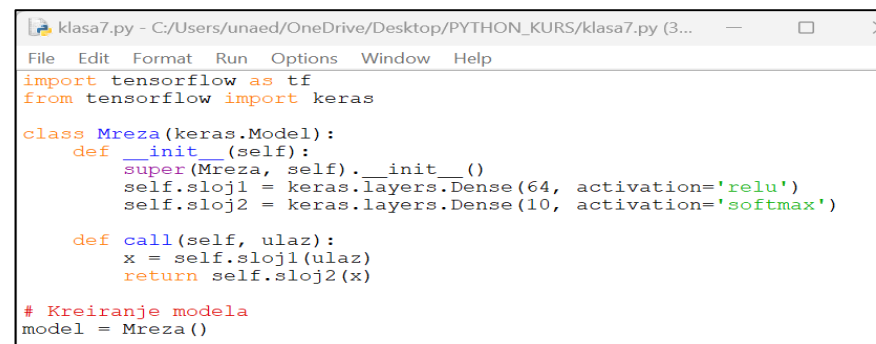


```
KLASA6.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/KLASA6.py...
File Edit Format Run Options Window Help
from django.db import models

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



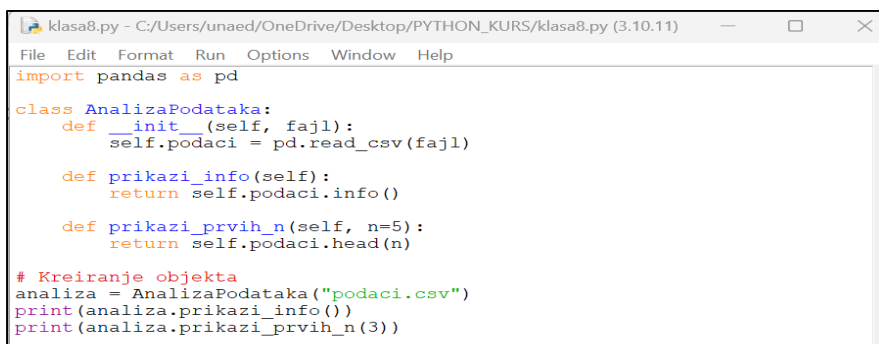
```
klasa7.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klasa7.py (3...
File Edit Format Run Options Window Help
import tensorflow as tf
from tensorflow import keras

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

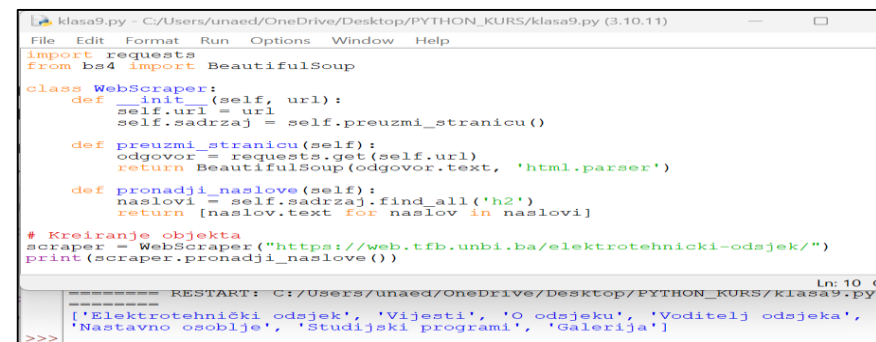
class AnalizaPodataka:
    def __init__(self, fajl):
        self.podaci = pd.read_csv(fajl)

    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.sadržaj = self.preuzmi_stranicu()

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

    def pronadji_naslove(self):
        naslovi = self.sadržaj.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())

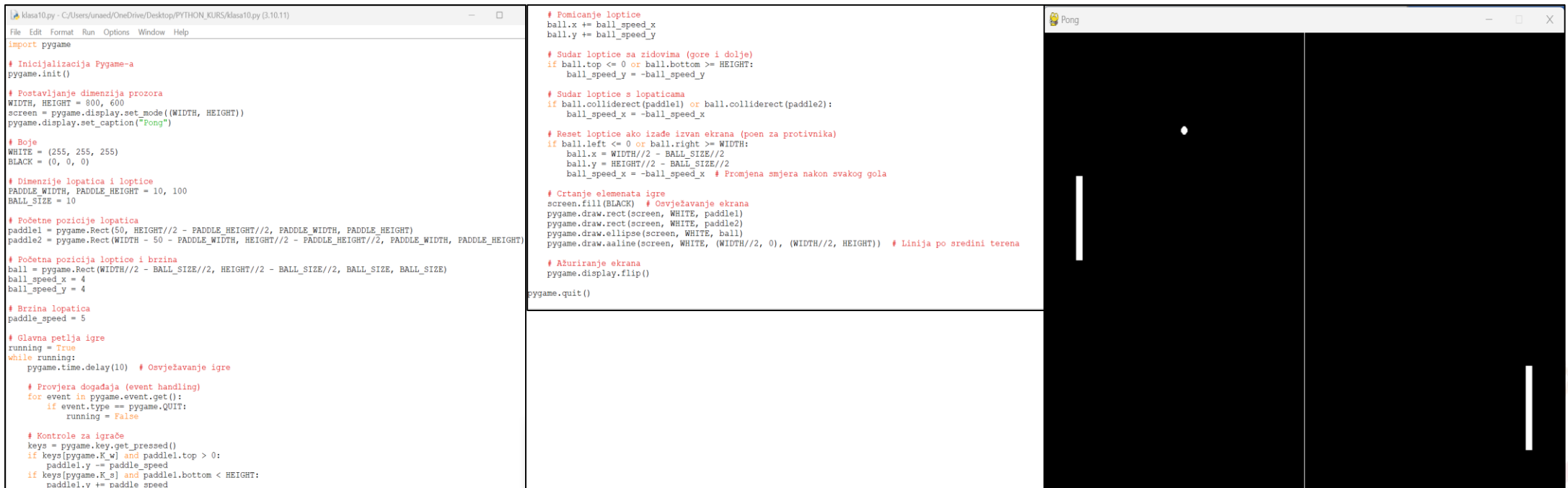
In: 10 Col
['Elektrotehnički odsjek', 'Vijesti', 'O odsjeku', 'Voditelj odsjeka',
'Nastavno osoblje', 'Studijski programi', 'Galerija']
>>>
```

Slika 53. Primjer preuzimanja podataka



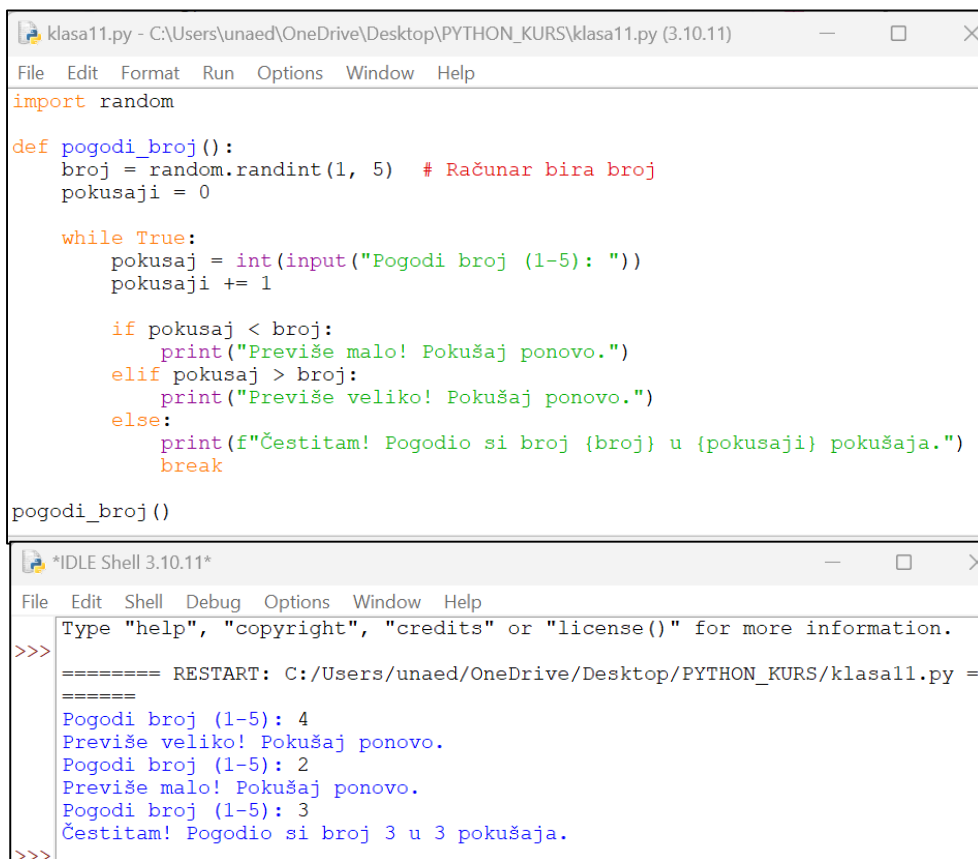
6. Klase

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



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

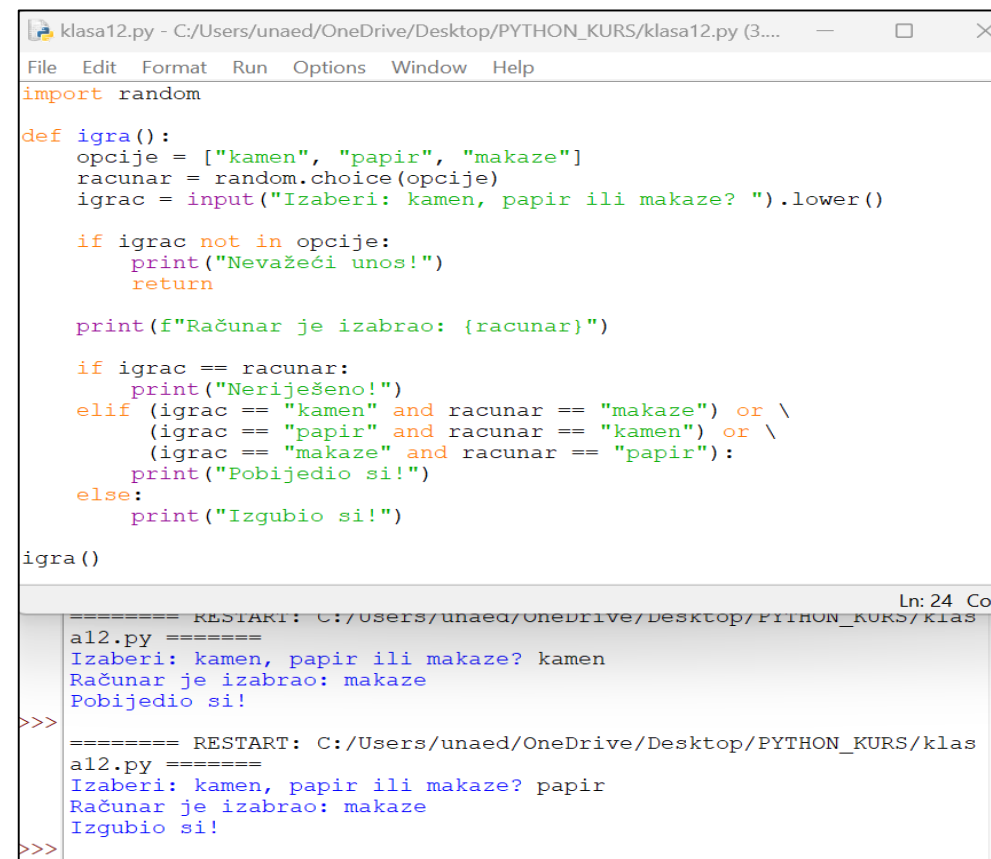
    while True:
        pokusaj = int(input("Pogodi broj (1-5): "))
        pokusaji += 1

        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!")

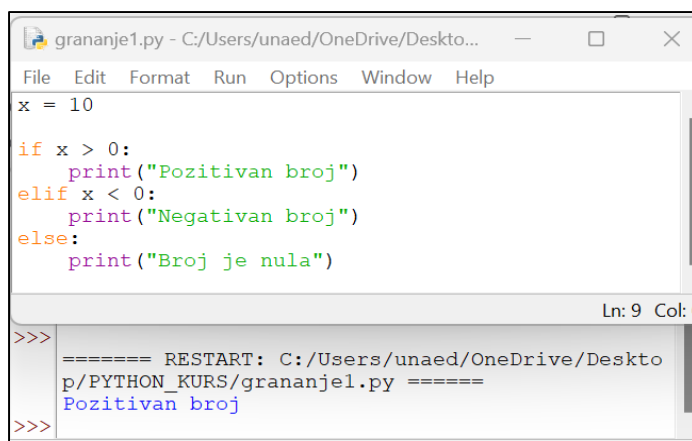
igra()

===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klas
al2.py =====
Izaberi: kamen, papir ili makaze? kamen
Računar je izabrao: makaze
Pobijedio si!
>>>
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/klas
al2.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.

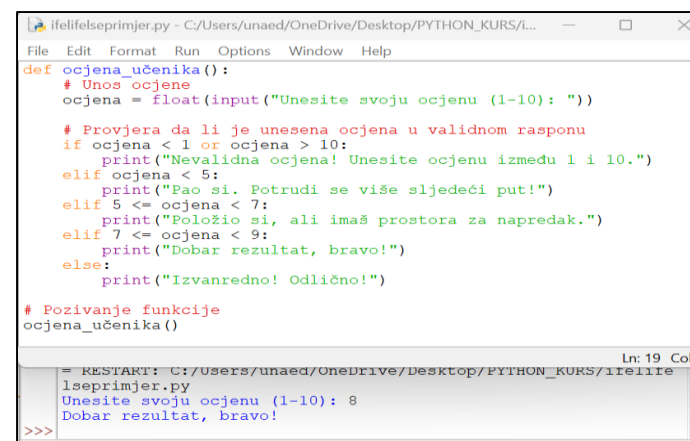


```
File Edit Format Run Options Window Help
x = 10

if x > 0:
    print("Pozitivan broj")
elif x < 0:
    print("Negativan broj")
else:
    print("Broj je nula")

Ln: 9 Col: 0

>>>
===== RESTART: C:/Users/unaed/OneDrive/Deskto
p/PYTHON_KURS/granjanje1.py =====
Pozitivan broj
>>>
```



```
File Edit Format Run Options Window Help
def ocjena_učenika():
    # Unos ocjene
    ocjena = float(input("Unesite svoju ocjenu (1-10): "))

    # Provjera da li je unesena ocjena u validnom rasponu
    if ocjena < 1 or ocjena > 10:
        print("Nevalidna ocjena! Unesite ocjenu između 1 i 10.")
    elif ocjena < 5:
        print("Pao si. Potrudi se više sljedeći put!")
    elif 5 <= ocjena < 7:
        print("Položio si, ali imaš prostora za napredak.")
    elif 7 <= ocjena < 9:
        print("Dobar rezultat, bravo!")
    else:
        print("Izvanredno! Odlično!")

# Pozivanje funkcije
ocjena_učenika()

Ln: 19 Col: 0

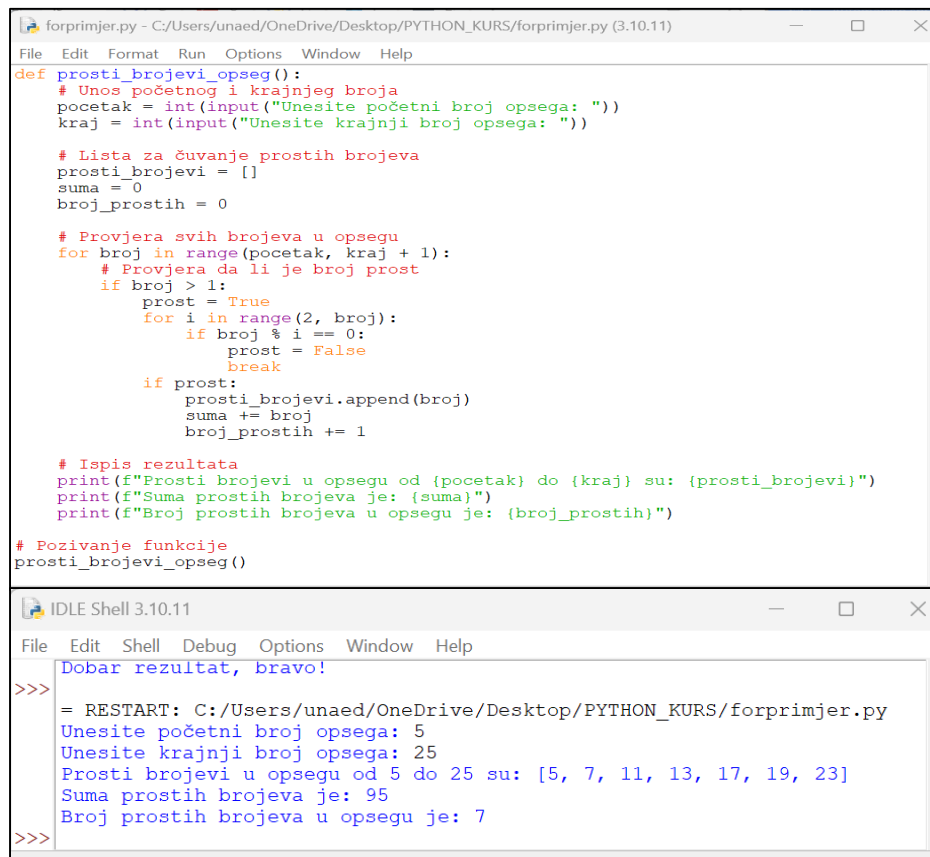
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/ifelife
lseprimjer.py
Unesite svoju ocjenu (1-10): 8
Dobar rezultat, Bravo!
>>>
```

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

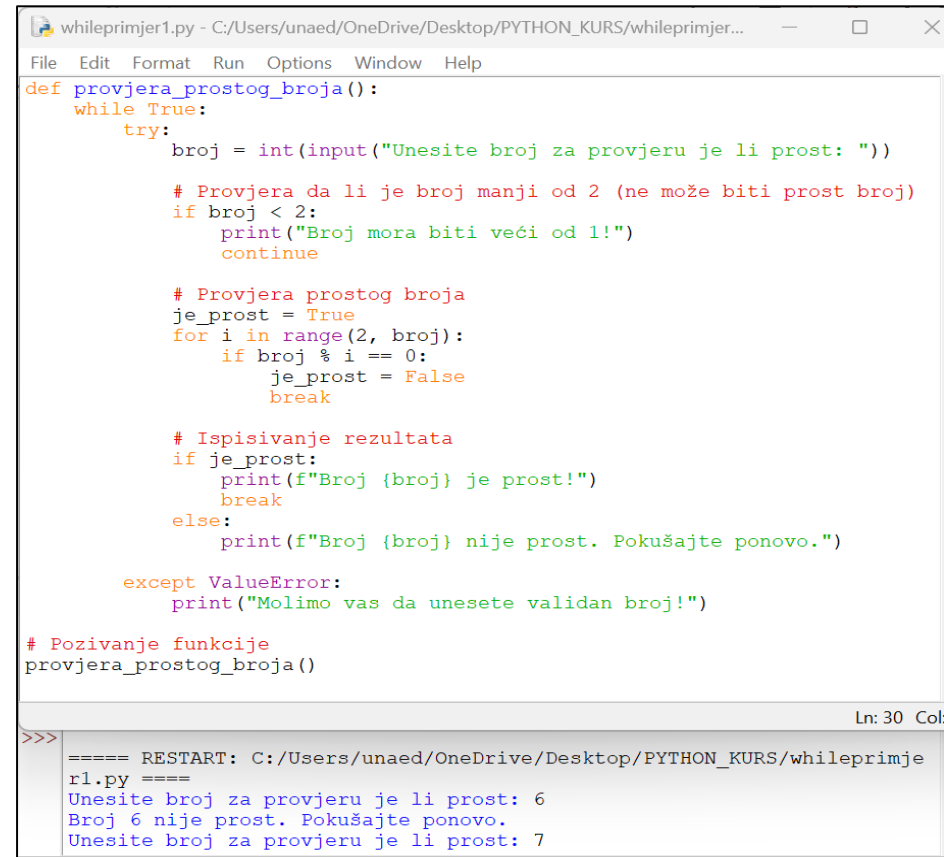
    # Provjera svih brojeva u opsegu
    for broj in range(pocetak, kraj + 1):
        # Provjera da li je broj prost
        if broj > 1:
            prost = True
            for i in range(2, broj):
                if broj % i == 0:
                    prost = False
                    break
         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

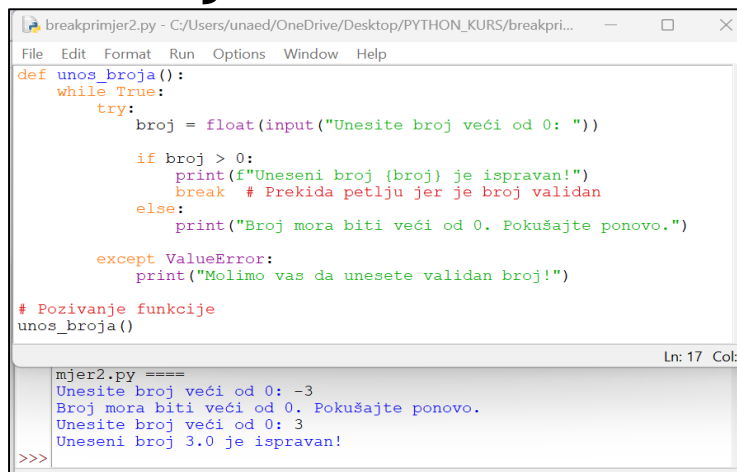
# Pozivanje funkcije
provjera_prostog_broja()

Ln: 30 Col: 1
>>>
===== 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.



```
def unos_broja():
    while True:
        try:
            broj = float(input("Unesite broj veći od 0: "))

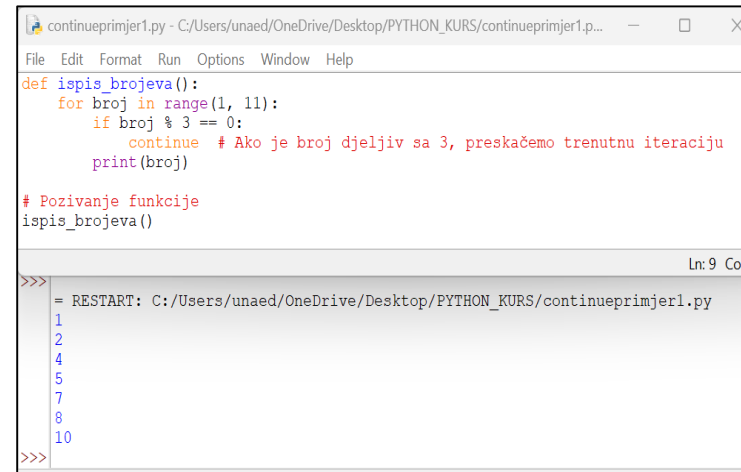
            if broj > 0:
                print(f"Uneseni broj {broj} je ispravan!")
                break # Prekida petlju jer je broj validan
            else:
                print("Broj mora biti veći od 0. Pokušajte ponovo.")

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

# Pozivanje funkcije
unos_broja()

mjer2.py ====
Unesite broj veći od 0: -3
Broj mora biti veći od 0. Pokušajte ponovo.
Unesite broj veći od 0: 3
Uneseni broj 3.0 je ispravan!
```

Slika 60. Primjer primjene naredbe `break`



```
def ispis_brojeva():
    for broj in range(1, 11):
        if broj % 3 == 0:
            continue # Ako je broj djeljiv sa 3, preskačemo trenutnu iteraciju
        print(broj)

# Pozivanje funkcije
ispis_brojeva()

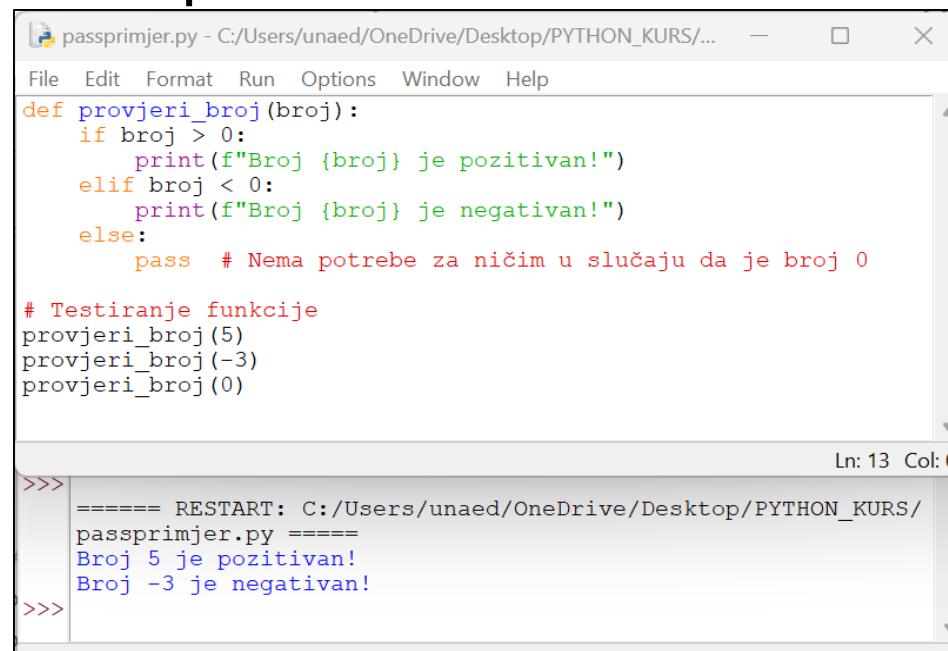
= RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/continueprimjer1.py
1
2
4
5
7
8
10
```

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.



```
passprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/...
File Edit Format Run Options Window Help
def provjeri_broj(broj):
    if broj > 0:
        print(f"Broj {broj} je pozitivan!")
    elif broj < 0:
        print(f"Broj {broj} je negativan!")
    else:
        pass # Nema potrebe za ničim u slučaju da je broj 0

# Testiranje funkcije
provjeri_broj(5)
provjeri_broj(-3)
provjeri_broj(0)

Ln: 13 Col: 0

>>>
===== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/
passprimjer.py =====
Broj 5 je pozitivan!
Broj -3 je negativan!
>>>
```

Slika 62. Primjer primjene naredbe `pass`

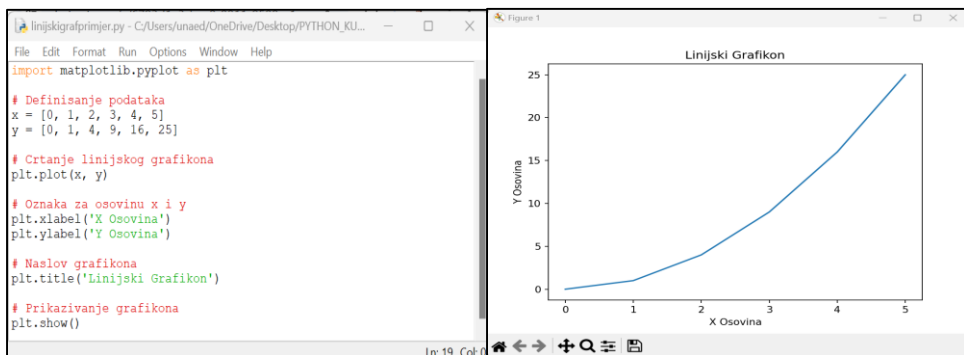


8. Funkcije za grafički prikaz

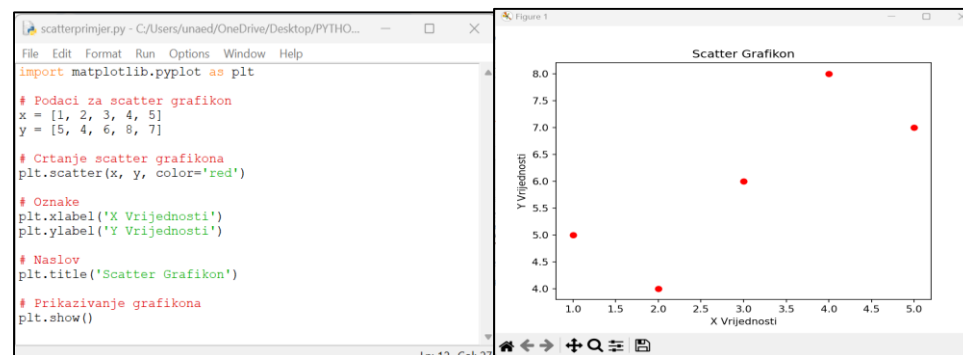
- U Pythonu, za vizualizaciju podataka u 2D i 3D prostoru, najčešće se koristi paket Matplotlib. Matplotlib omogućava jednostavan i fleksibilan način za generiranje širokog spektra grafova i dijagrama, uključujući 2D i 3D prikaze podataka. Osim toga, za napredniji grafički prikaz i interaktivne 3D grafike, 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.



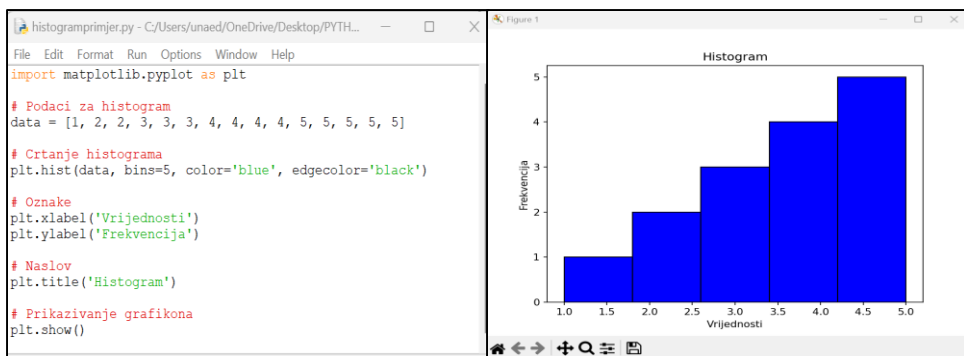
8. Funkcije za grafički prikaz



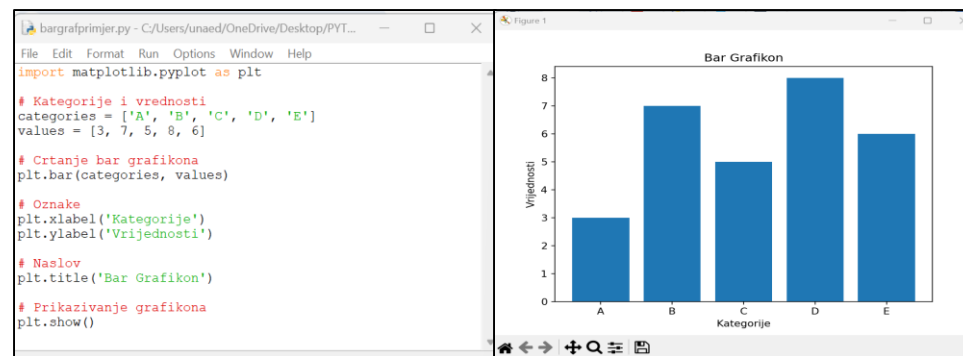
Slika 63. *Primjer linijskog grafa*



Slika 64. *Primjer scatter grafa*



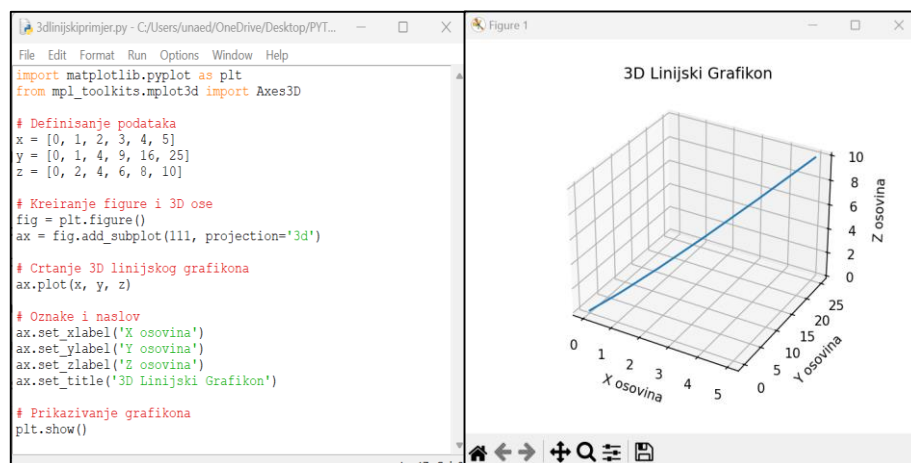
Slika 65. *Primjer histograma*



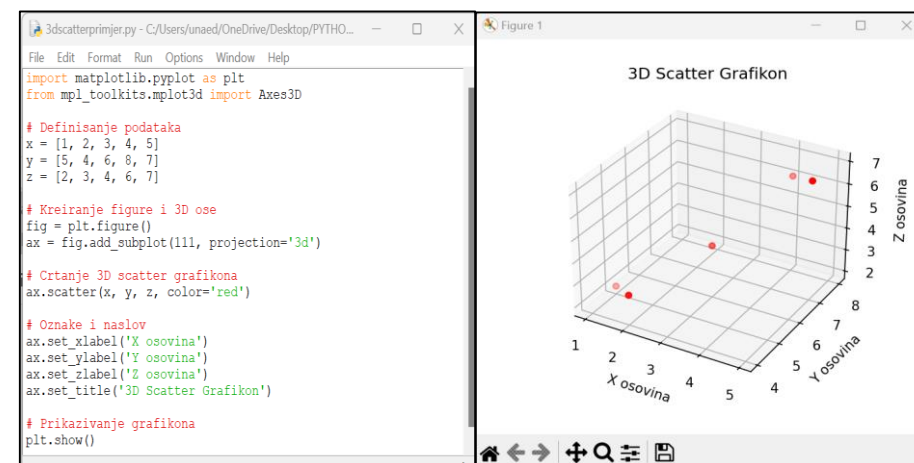
Slika 66. *Primjer bar grafa*

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.

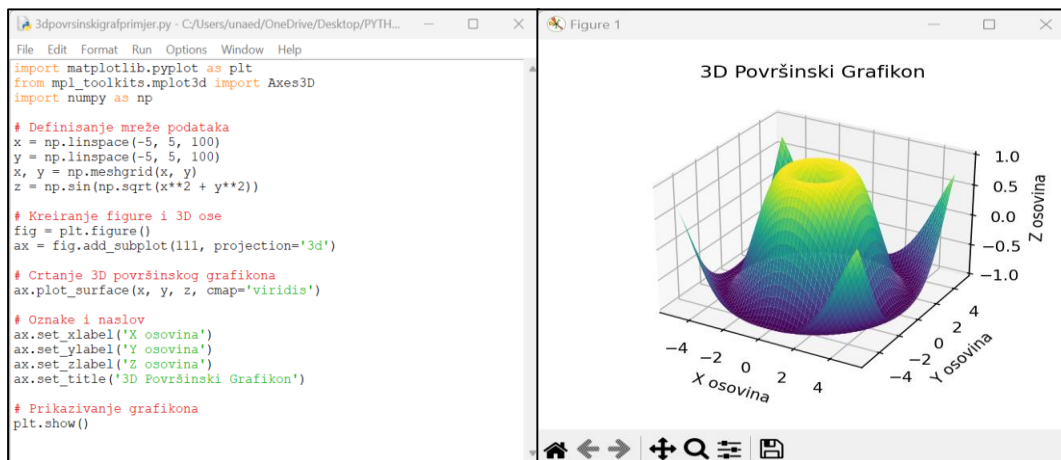


Slika 67. Primjer 3D linijskog grafa

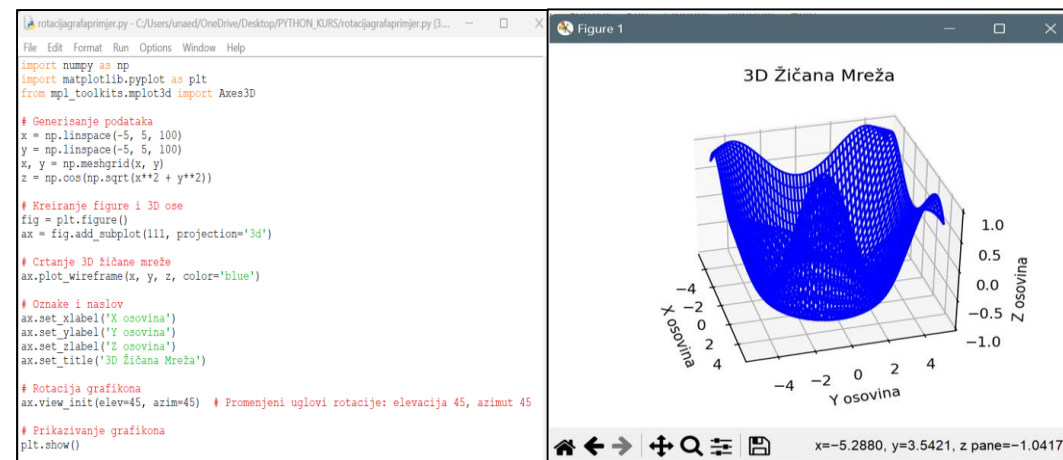


Slika 68. Primjer 3D scatter grafa

8. Funkcije za grafički prikaz



Slika 69. Primjer 3D površinskog grafa



Slika 70. Primjer rotacije 3D grafa

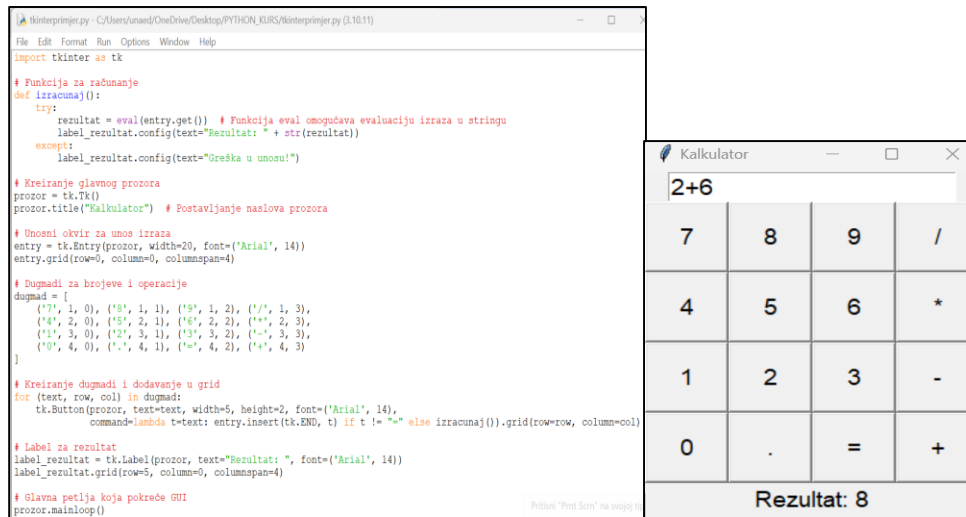


9. Grafički interfejs

- 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 - Takođe 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.



Slika 71. Primjer primjene Tkinter biblioteke

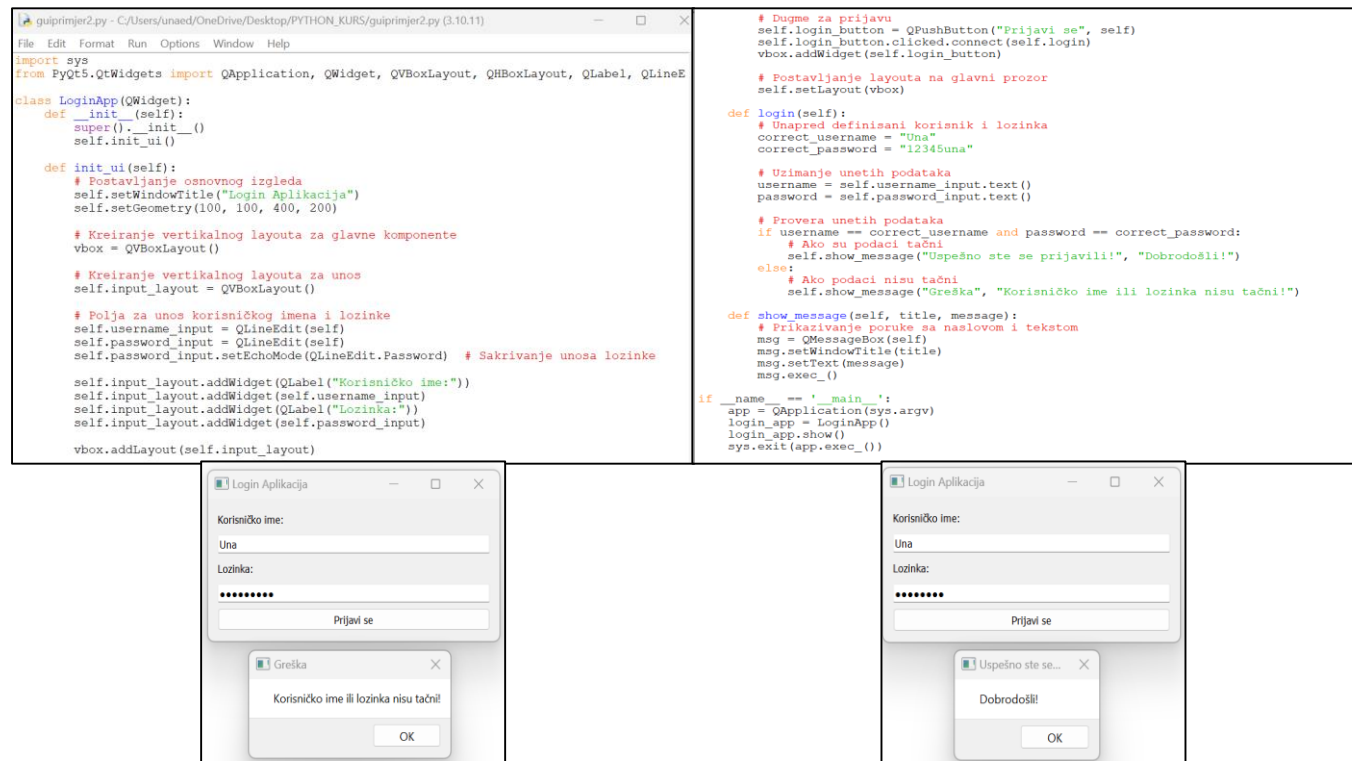
Osnovni elementi Tkinter-a:

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

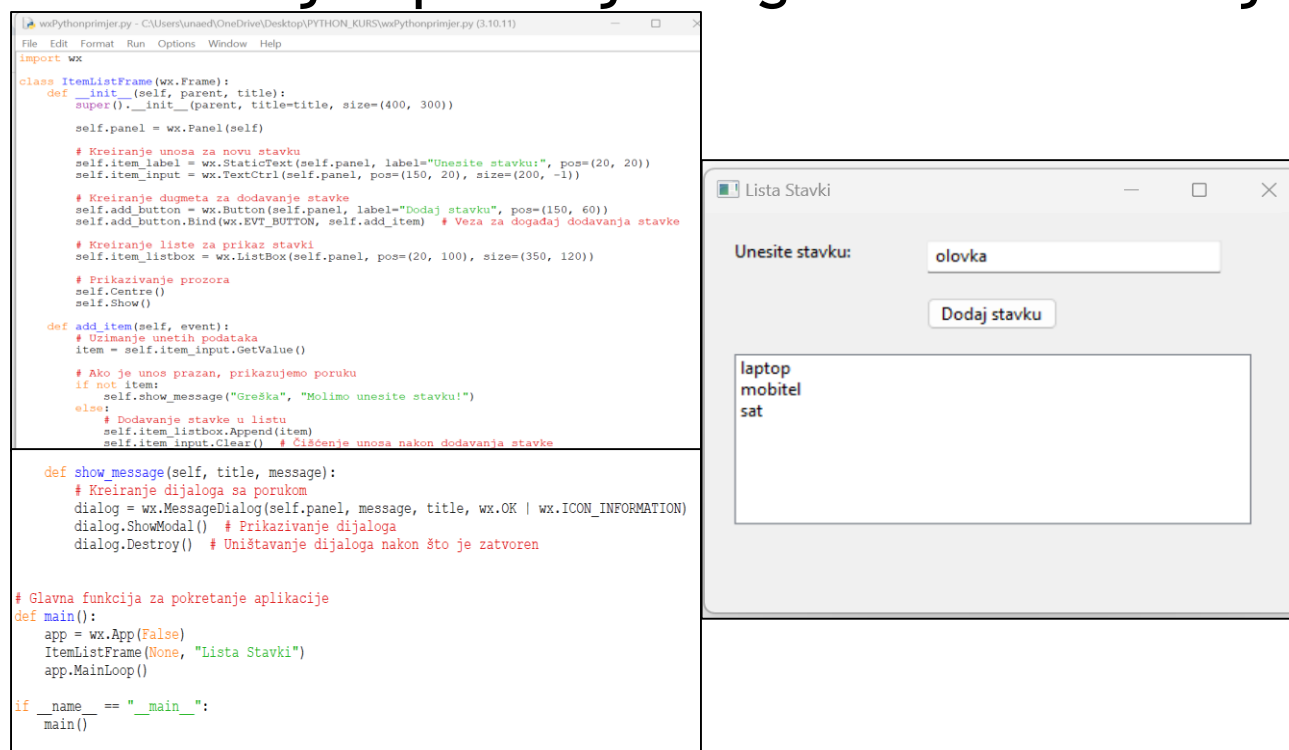
- PyQt5 je biblioteka koja se temelji na Qt frameworku, koja pruža veće mogućnosti i fleksibilnost.



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

9. Grafički interfejs

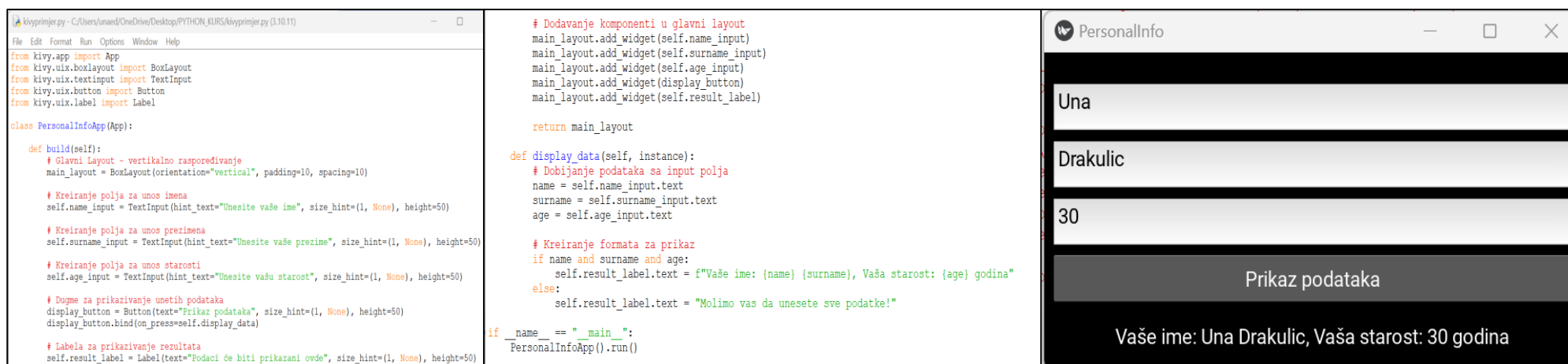
- wxPython je biblioteka za razvoj grafičkog korisničkog interfejsa (GUI) u Pythonu. To je Python wrapper za wxWidgets, koji je C++ biblioteka koja omogućava kreiranje aplikacija sa grafičkim interfejsom.



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



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)

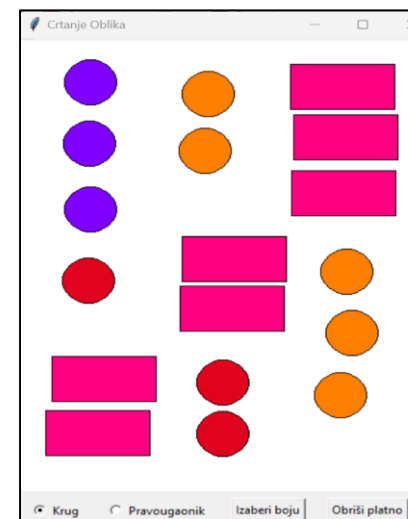
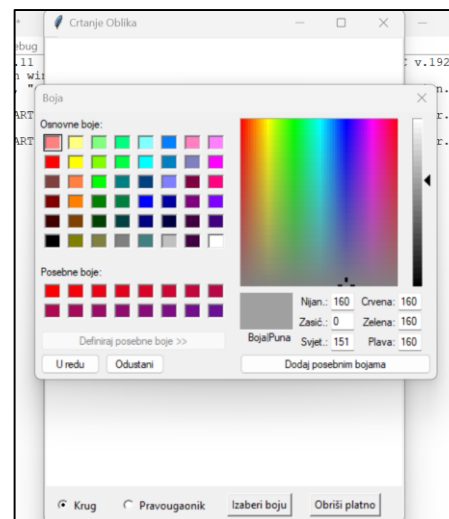
rectangle_button = tk.Radiobutton(root, text="Pravougaonik", variable=shape_var, value="Rectangle")
rectangle_button.pack(side=tk.LEFT, padx=10)

# 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

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

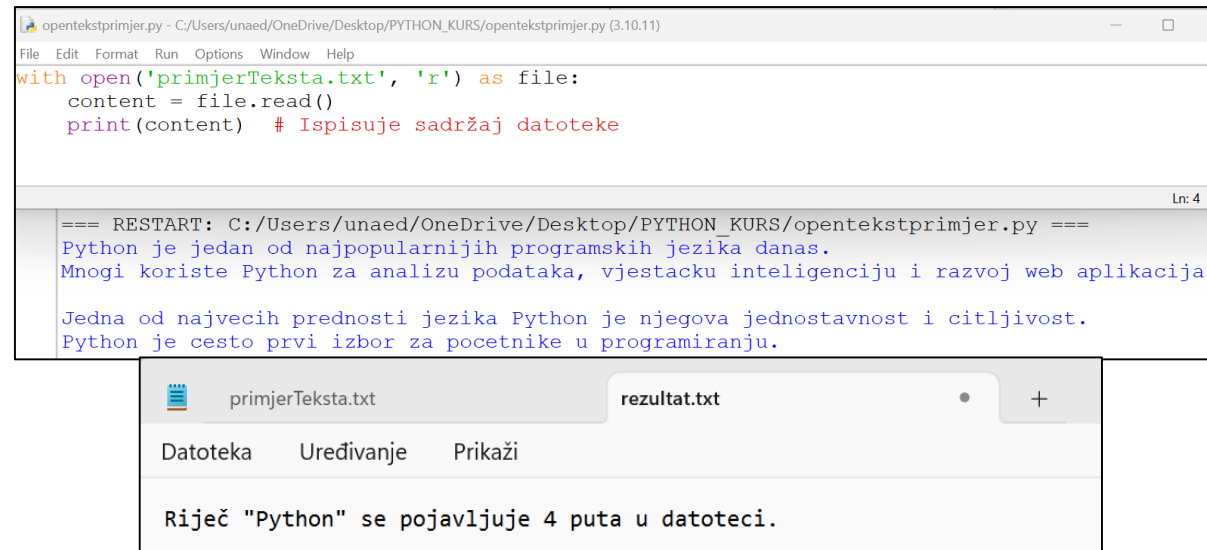
- Umjesto da pozivamo `file.close()` eksplicitno, možemo koristiti `with` izjavu koja automatski zatvara datoteku kada se završi rad sa njom. Ovo je bolji način jer osigurava da će datoteka biti zatvorena, čak iako dođe do greške u toku obrade.

```
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.



```
opentekstprimjer.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/opentekstprimjer.py (3.10.11)
File Edit Format Run Options Window Help
with open('primjerTeksta.txt', 'r') as file:
    content = file.read()
    print(content) # Ispisuje sadržaj datoteke

Ln: 4 Co

=== RESTART: C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/opentekstprimjer.py ===
Python je jedan od najpopularnijih programskih jezika danas.
Mnogi koriste Python za analizu podataka, vjestacku inteligenciju i razvoj web aplikacija.

Jedna od najvećih prednosti jezika Python je njegova jednostavnost i citljivost.
Python je cesto prvi izbor za pocetnike u programiranju.

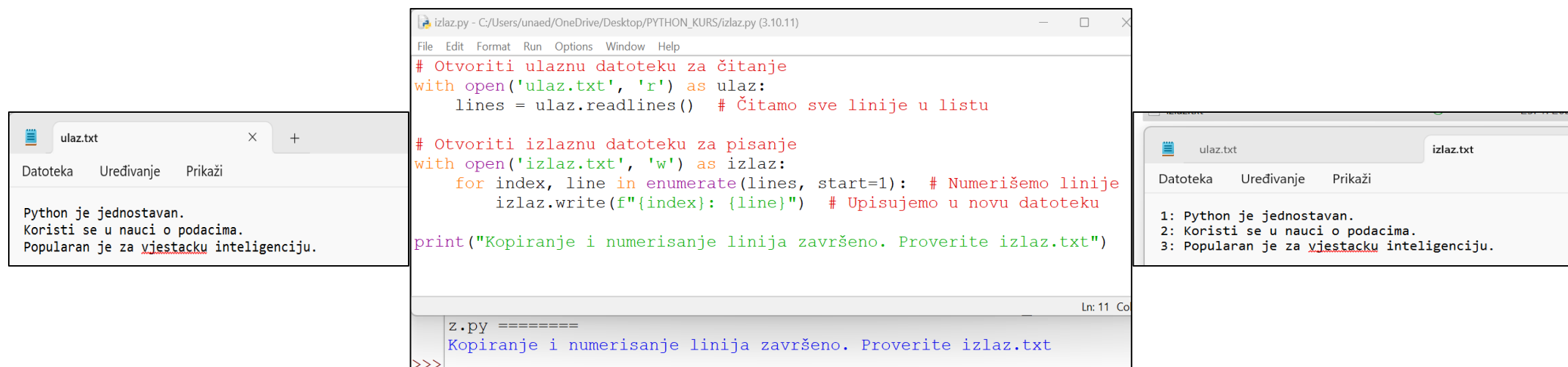
primjerTeksta.txt rezultat.txt
Datoteka Uređivanje Prikaži
Riječ "Python" se pojavljuje 4 puta u datoteci.
```

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.



```
izlaz.py - C:/Users/unaed/OneDrive/Desktop/PYTHON_KURS/izlaz.py (3.10.11)
File Edit Format Run Options Window Help
# Otvoriti ulaznu datoteku za čitanje
with open('ulaz.txt', 'r') as ulaz:
    lines = ulaz.readlines() # Čitamo sve linije u listu

# Otvoriti izlaznu datoteku za pisanje
with open('izlaz.txt', 'w') as izlaz:
    for index, line in enumerate(lines, start=1): # Numerišemo linije
        izlaz.write(f"{index}: {line}") # Upisujemo u novu datoteku

print("Kopiranje i numerisanje linija završeno. Proverite izlaz.txt")

z.py =====
Kopiranje i numerisanje linija završeno. Proverite izlaz.txt
>>>
```

ulaz.txt

Datoteka Uređivanje Prikaži

Python je jednostavan.
Koristi se u nauci o podacima.
Popularan je za vjestacku inteligenciju.

ulaz.txt izlaz.txt

Datoteka Uređivanje Prikaži

1: Python je jednostavan.
2: Koristi se u nauci o podacima.
3: Popularan je za vjestacku inteligenciju.

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



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



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.

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