# Green University of Bangladesh
## Department of Computer Science and Engineering (CSE)
**Faculty of Sciences and Engineering**
**Semester: (Spring, Year:2025), B.Sc. in CSE (Day)**

**CLP- NO: 03**
**Course Title: Artificial Intelligence Lab**
**Course Code: CSE-316 Section: 221_D7**

**Student Details**

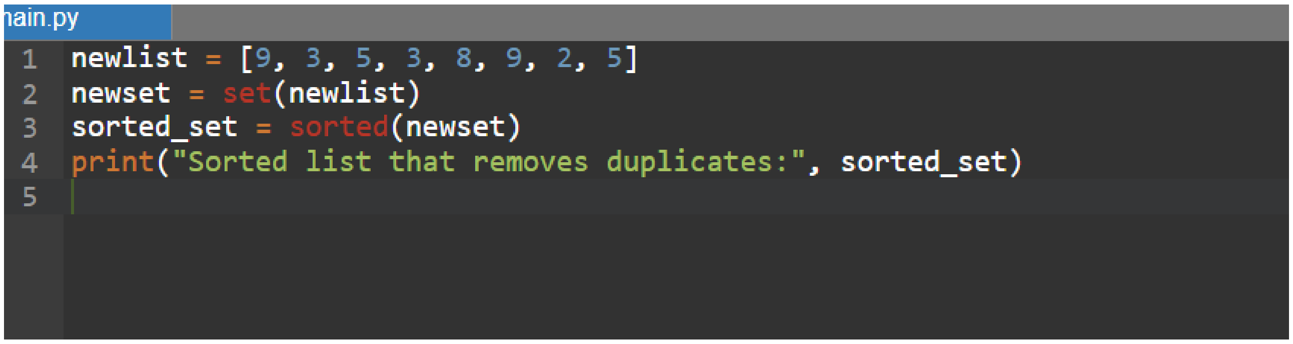
| | Name | ID |
|---|---|---|
| 1. | Nadia | 221902326 |

**Submission Date :10-02-2025**
**Course Teacher's Name :Md. Sabbir Hosen Mamun**

**Lab Report Status**
**Marks: ………………………………… Signature:.....................**
**Comments:............................................. Date:..............................**

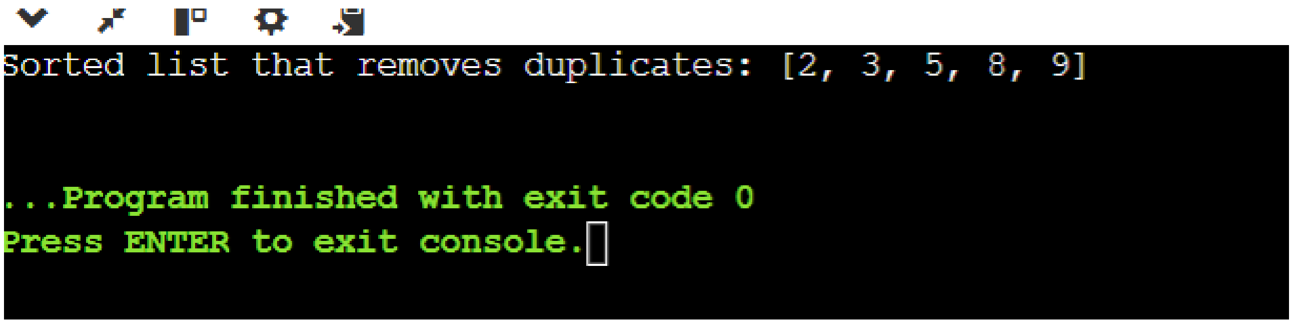
**Task-01:List: Given a list of numbers, remove duplicates and sort in ascending order.**

Code:



```
newlist = [9, 3, 5, 3, 8, 9, 2, 5]
newset = set(newlist)
sorted_set = sorted(newset)
print("Sorted list that removes duplicates:", sorted_set)
```

Output:

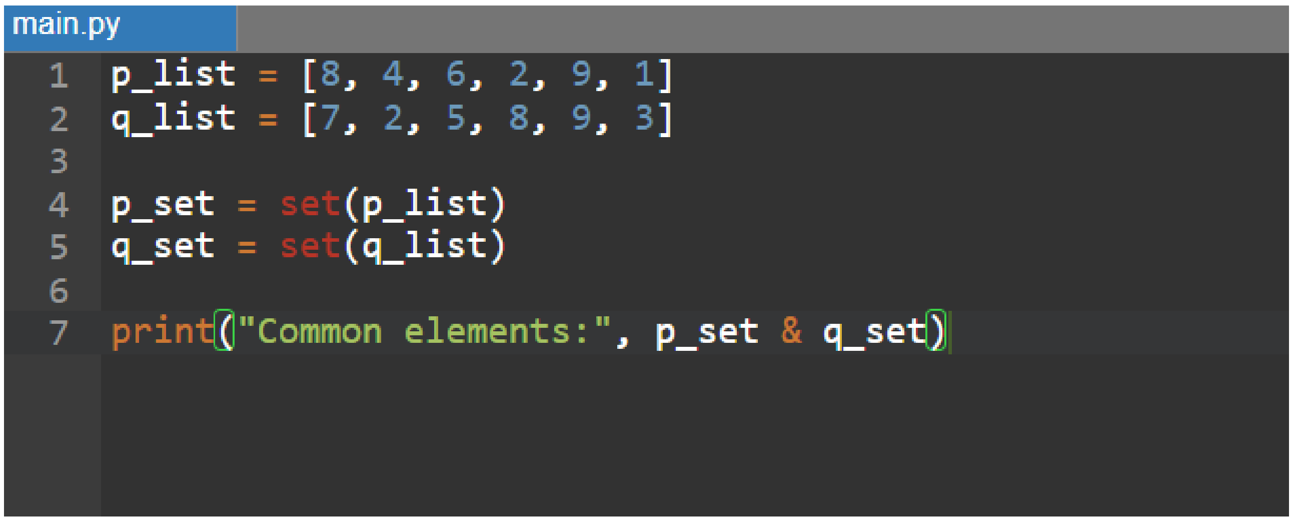


```
Sorted list that removes duplicates: [2, 3, 5, 8, 9]


...Program finished with exit code 0
Press ENTER to exit console.
```

Fig 01: Remove Duplicates and Sort in ascending order.

**Task-02:Set: Find the common elements between two lists using sets.**

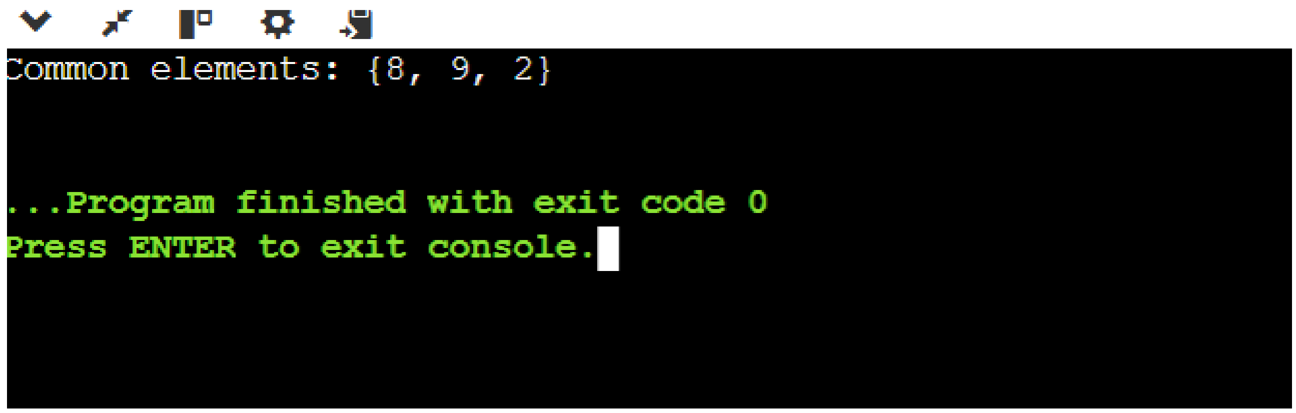
Code:



main.py

```python
p_list = [8, 4, 6, 2, 9, 1]
q_list = [7, 2, 5, 8, 9, 3]

p_set = set(p_list)
q_set = set(q_list)

print("Common elements:", p_set & q_set)
```

Output:

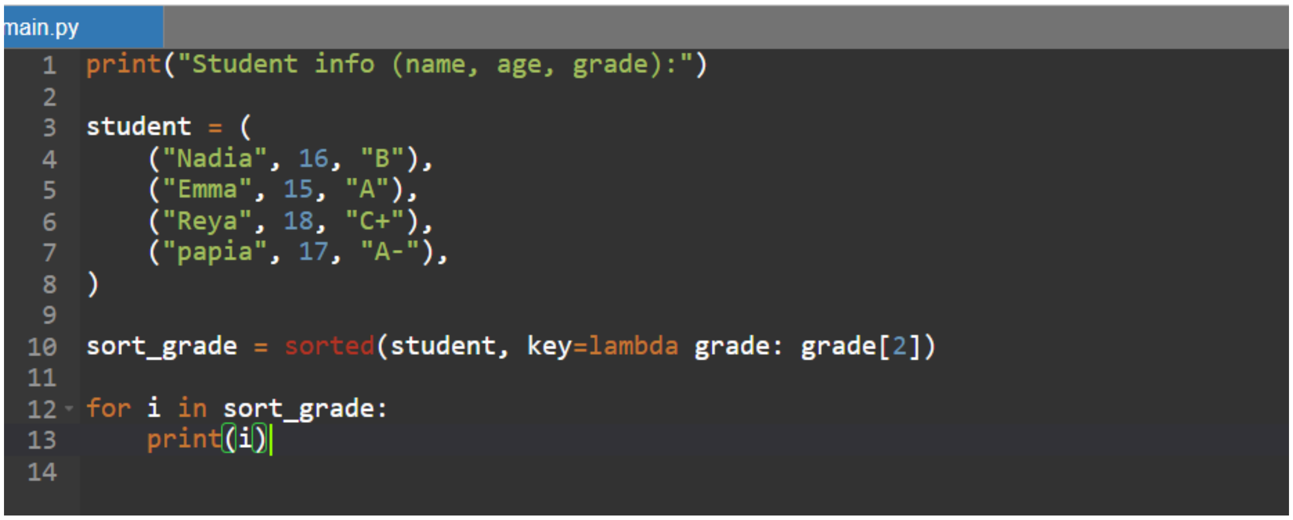


```
Common elements: {8, 9, 2}


...Program finished with exit code 0
Press ENTER to exit console.
```

Fig 02: Common Elements.

**Task-03:**Tuple: Create a tuple of student records (name, age, grade) and sort by grade.

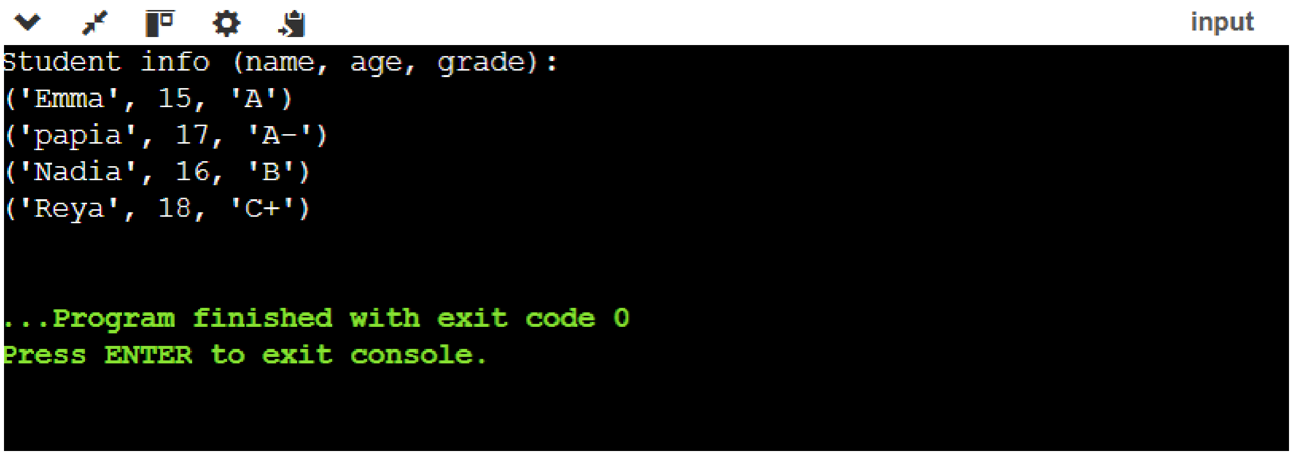
Code:



main.py

```python
print("Student info (name, age, grade):")

student = (
    ("Nadia", 16, "B"),
    ("Emma", 15, "A"),
    ("Reya", 18, "C+"),
    ("papia", 17, "A-"),
)

sort_grade = sorted(student, key=lambda grade: grade[2])

for i in sort_grade:
    print(i)

```

Output:

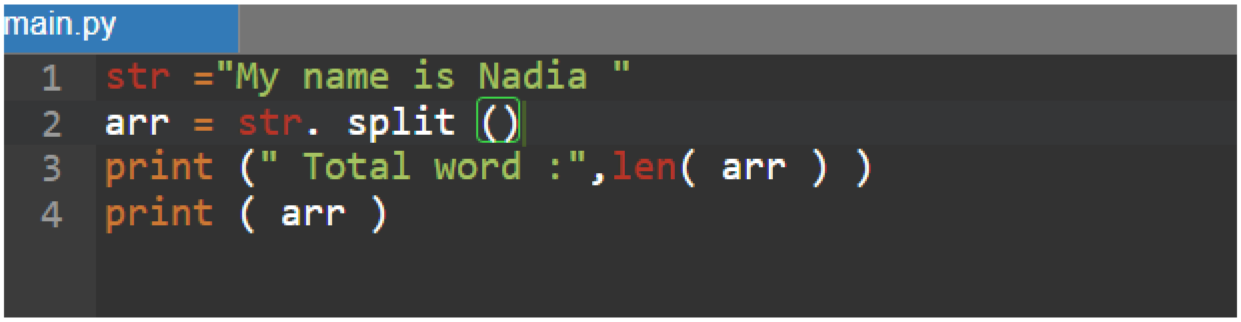


```
Student info (name, age, grade):
('Emma', 15, 'A')
('papia', 17, 'A-')
('Nadia', 16, 'B')
('Reya', 18, 'C+')


...Program finished with exit code 0
Press ENTER to exit console.
```

Fig 03:Student Records

**Task-04:Dictionary: Count word occurrences in a given text and store them in a dictionary.**

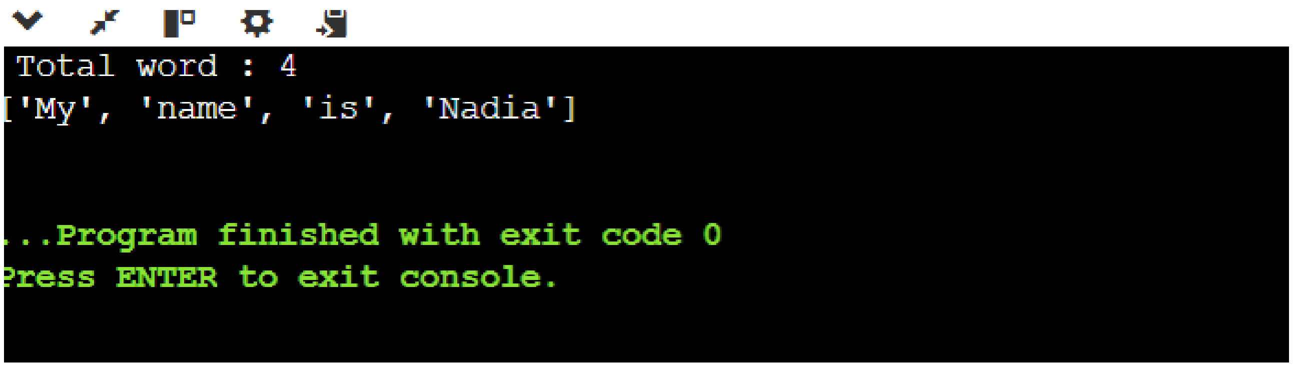
Code:



main.py

```python
str ="My name is Nadia "
arr = str. split ()
print (" Total word :",len( arr ) )
print ( arr )
```

Output:

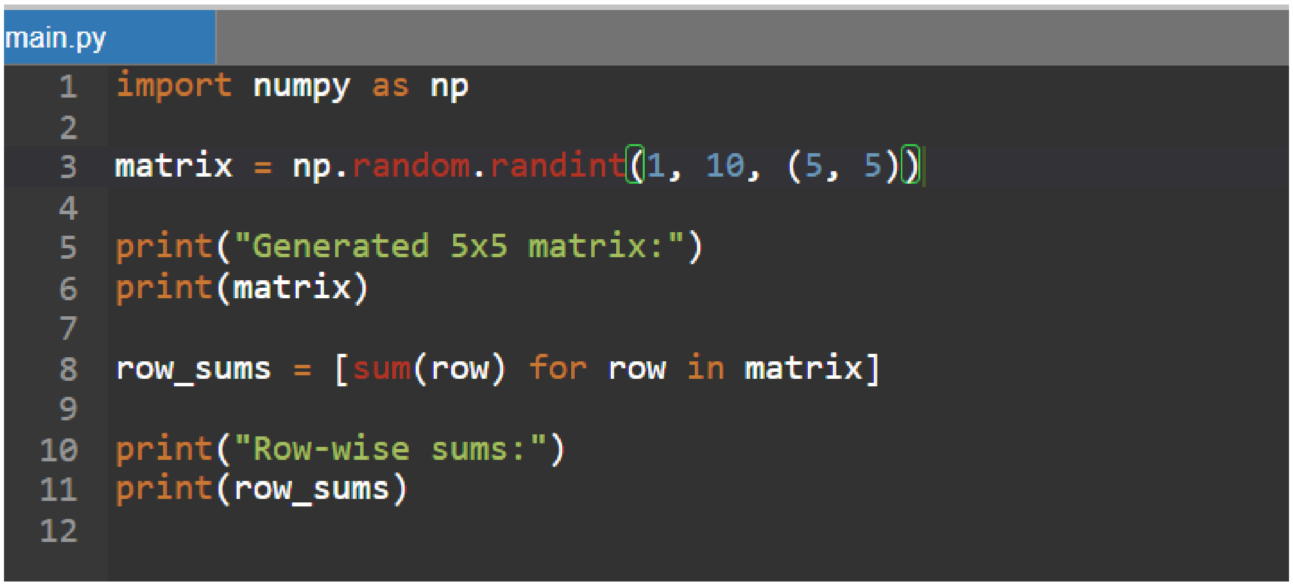


```
 Total word : 4
['My', 'name', 'is', 'Nadia']


...Program finished with exit code 0
Press ENTER to exit console.
```

Fig 04 : Count word.

**Task-05:**NumPy#1: Generate a 5x5 matrix of random integers and compute row-wise sums.

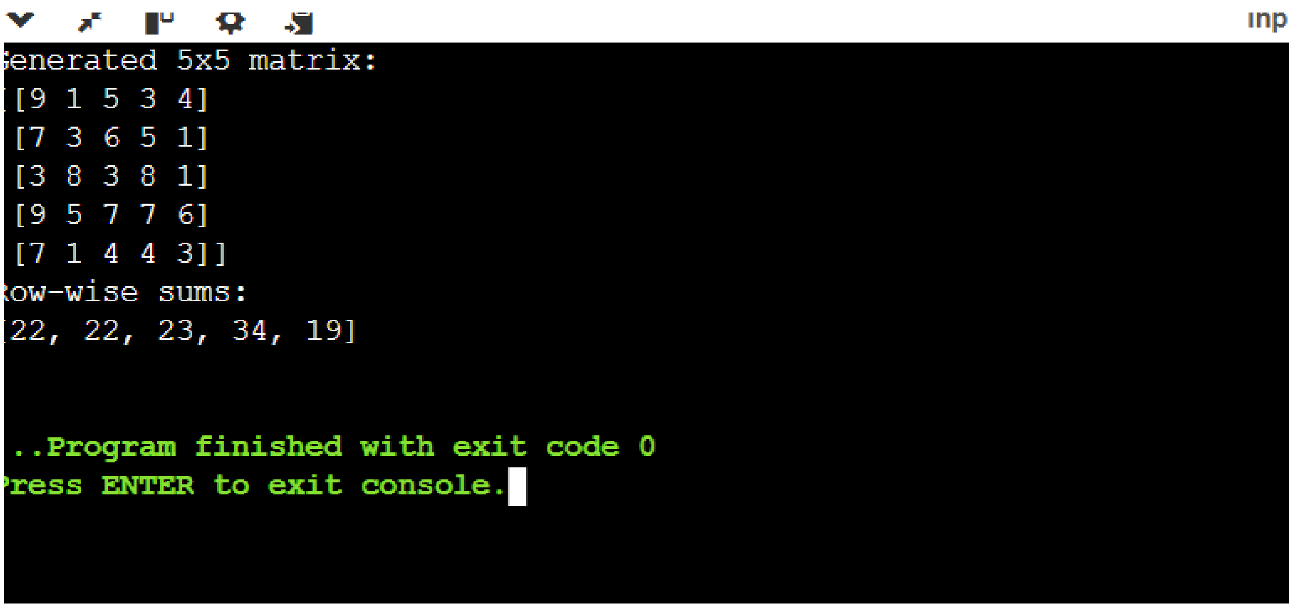
Code:



main.py

```python
import numpy as np

matrix = np.random.randint(1, 10, (5, 5))

print("Generated 5x5 matrix:")
print(matrix)

row_sums = [sum(row) for row in matrix]

print("Row-wise sums:")
print(row_sums)

```

Output:

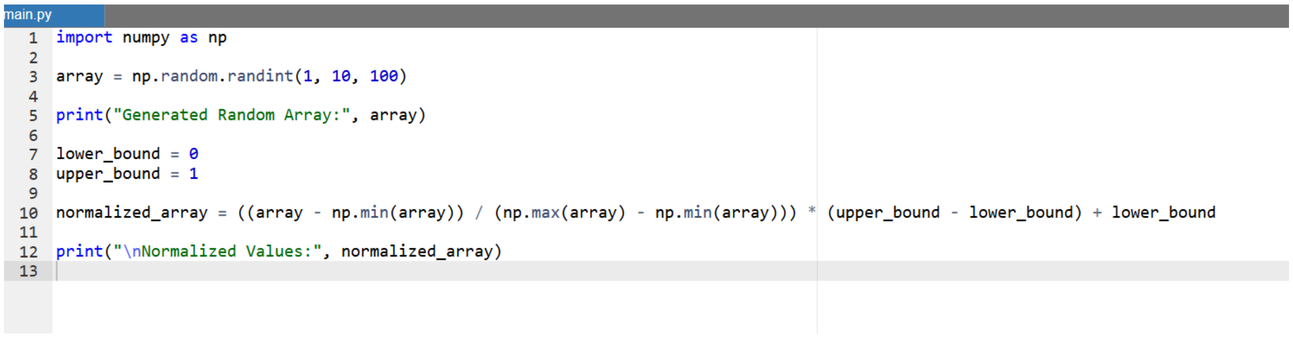


```
enerated 5x5 matrix:
[9 1 5 3 4]
 [7 3 6 5 1]
 [3 8 3 8 1]
 [9 5 7 7 6]
 [7 1 4 4 3]]
ow-wise sums:
22, 22, 23, 34, 19]


..Program finished with exit code 0
ress ENTER to exit console.
```

Fig 05: Create random matrix and compute row-wise

## Task-06:NumPy#2: Create an array of 100 random values and normalize them between 0 and 1.

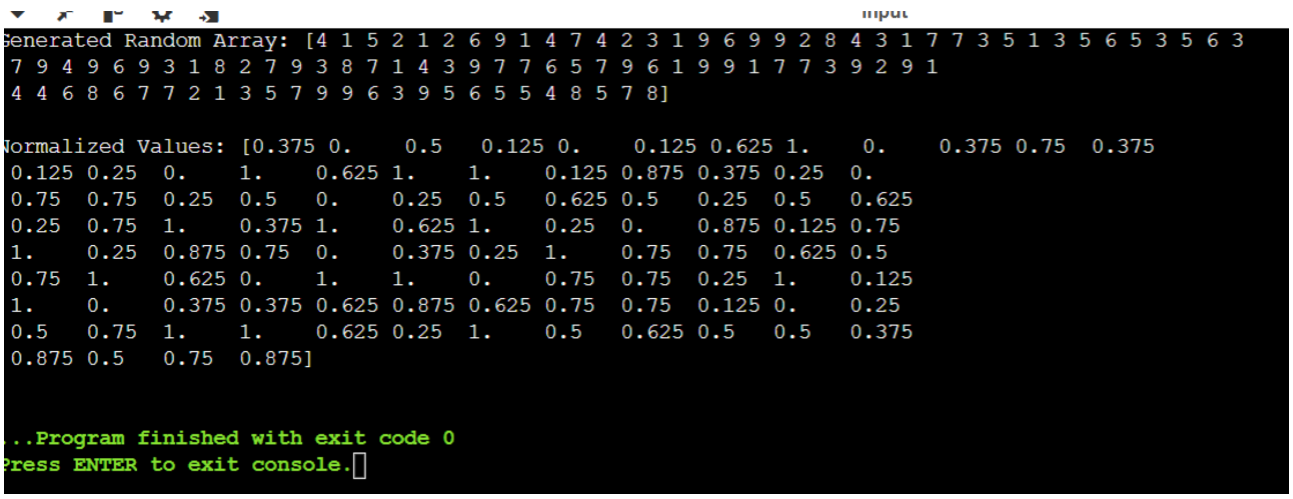
Code:



main.py

```python
import numpy as np

array = np.random.randint(1, 10, 100)

print("Generated Random Array:", array)

lower_bound = 0
upper_bound = 1

normalized_array = ((array - np.min(array)) / (np.max(array) - np.min(array))) * (upper_bound - lower_bound) + lower_bound

print("\nNormalized Values:", normalized_array)

```

Output:

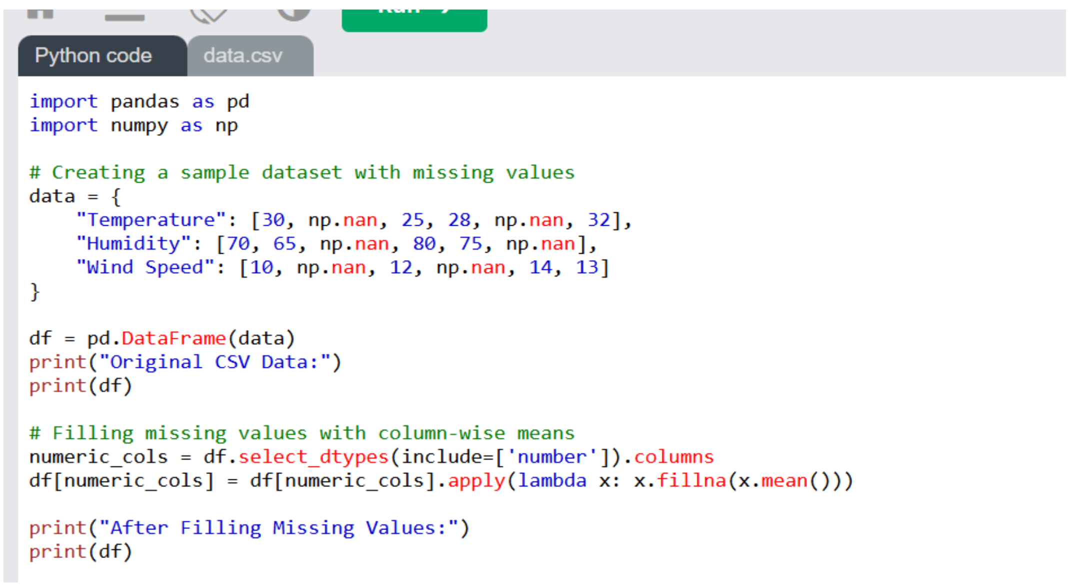


```
enerated Random Array: [4 1 5 2 1 2 6 9 1 4 7 4 2 3 1 9 6 9 9 2 8 4 3 1 7 7 3 5 1 3 5 6 5 3 5 6 3
 7 9 4 9 6 9 3 1 8 2 7 9 3 8 7 1 4 3 9 7 7 6 5 7 9 6 1 9 9 1 7 7 3 9 2 9 1
 4 4 6 8 6 7 7 2 1 3 5 7 9 9 6 3 9 5 6 5 5 4 8 5 7 8]

ormalized Values: [0.375 0.    0.5   0.125 0.    0.125 0.625 1.    0.    0.375 0.75  0.375
 0.125 0.25  0.    1.    0.625 1.    1.    0.125 0.875 0.375 0.25  0.
 0.75  0.75  0.25  0.5   0.    0.25  0.5   0.625 0.5   0.25  0.5   0.625
 0.25  0.75  1.    0.375 1.    0.625 1.    0.25  0.    0.875 0.125 0.75
 1.    0.25  0.875 0.75  0.    0.375 0.25  1.    0.75  0.75  0.625 0.5
 0.75  1.    0.625 0.    1.    1.    0.    0.75  0.75  0.25  1.    0.125
 1.    0.    0.375 0.375 0.625 0.875 0.625 0.75  0.75  0.125 0.    0.25
 0.5   0.75  1.    1.    0.625 0.25  1.    0.5   0.625 0.5   0.5   0.375
 0.875 0.5   0.75  0.875]


..Program finished with exit code 0
ress ENTER to exit console.
```

Fig 06: Random 100 values Normalization

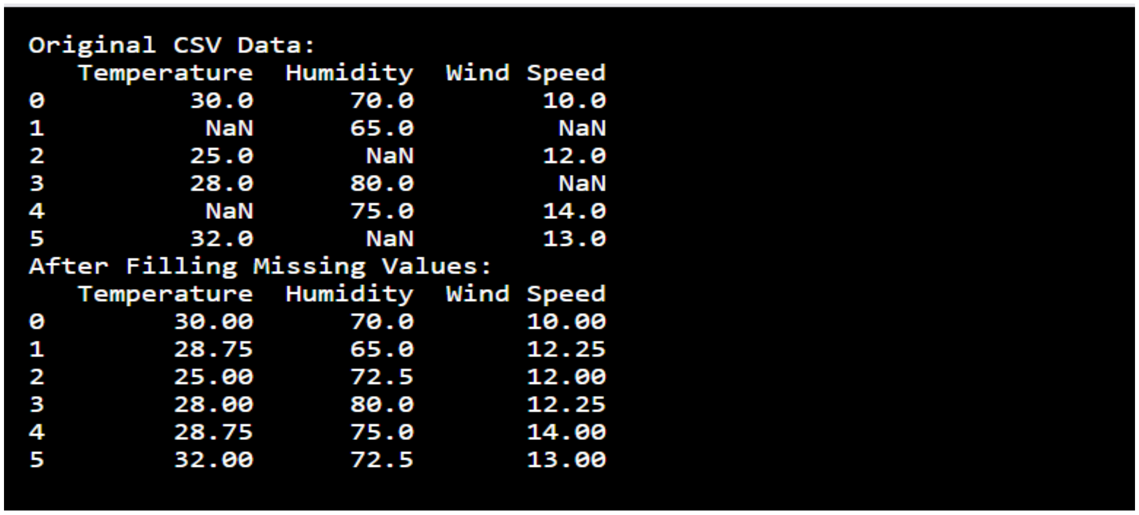
## Task-08:Pandas#2: Fill missing values in a dataset with column-wise means.

Code:



```python
import pandas as pd
import numpy as np

# Creating a sample dataset with missing values
data = {
    "Temperature": [30, np.nan, 25, 28, np.nan, 32],
    "Humidity": [70, 65, np.nan, 80, 75, np.nan],
    "Wind Speed": [10, np.nan, 12, np.nan, 14, 13]
}

df = pd.DataFrame(data)
print("Original CSV Data:")
print(df)

# Filling missing values with column-wise means
numeric_cols = df.select_dtypes(include=['number']).columns
df[numeric_cols] = df[numeric_cols].apply(lambda x: x.fillna(x.mean()))

print("After Filling Missing Values:")
print(df)
```

Output:

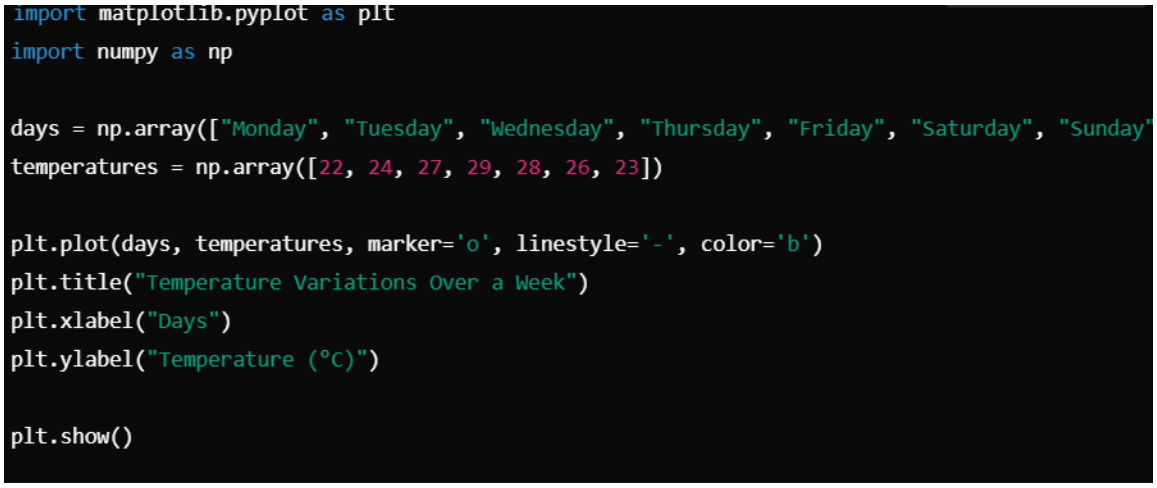


```
Original CSV Data:
   Temperature  Humidity  Wind Speed
0         30.0      70.0        10.0
1          NaN      65.0         NaN
2         25.0       NaN        12.0
3         28.0      80.0         NaN
4          NaN      75.0        14.0
5         32.0       NaN        13.0
After Filling Missing Values:
   Temperature  Humidity  Wind Speed
0        30.00      70.0       10.00
1        28.75      65.0       12.25
2        25.00      72.5       12.00
3        28.00      80.0       12.25
4        28.75      75.0       14.00
5        32.00      72.5       13.00
```

Fig 0 8: Fill missing values

**Task-09:Matplotlib#1: Plot a line graph showing temperature variations over a week.**

Code:



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

days = np.array(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"
temperatures = np.array([22, 24, 27, 29, 28, 26, 23])

plt.plot(days, temperatures, marker='o', linestyle='-', color='b')
plt.title("Temperature Variations Over a Week")
plt.xlabel("Days")
plt.ylabel("Temperature (°C)")

plt.show()
```

Output:

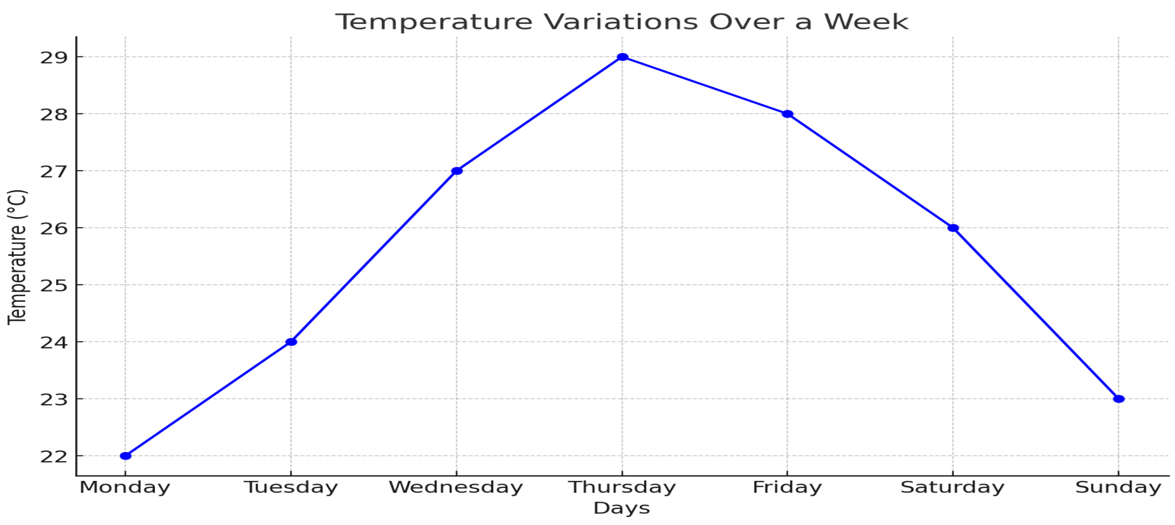
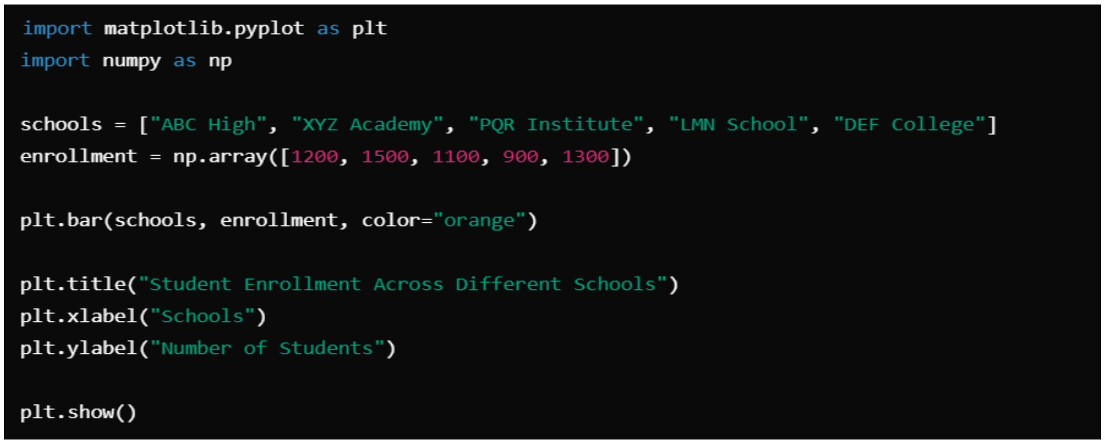


Fig 09: Plot a Line Graph

## Task-10:Matplotlib#2: Create a bar chart comparing sales revenue across different regions.

Code:



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

schools = ["ABC High", "XYZ Academy", "PQR Institute", "LMN School", "DEF College"]
enrollment = np.array([1200, 1500, 1100, 900, 1300])

plt.bar(schools, enrollment, color="orange")

plt.title("Student Enrollment Across Different Schools")
plt.xlabel("Schools")
plt.ylabel("Number of Students")

plt.show()
```

Output:

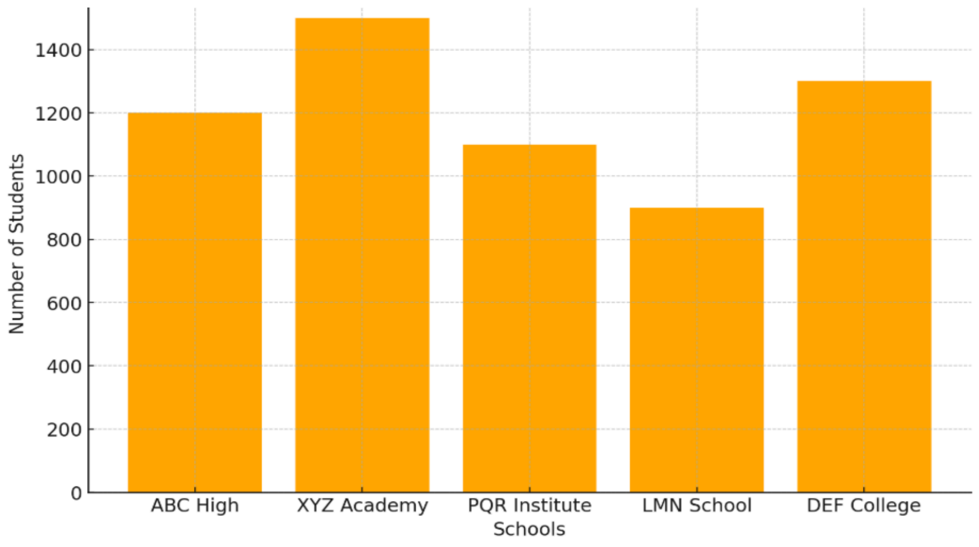


Fig 10: Create a Bar Chart

Github Link: