



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)

Course Code: CSE-316 Section: D14

Student Details

Name	ID
Ashraful Islam Miraj	221902231

Lab Date : 05-02-2025

Submission Date : 11-02-2025

Course Teacher's Name :Md. Sabbir Hosen Mamun

[For Teachers use only: **Don't Write Anything inside this box]**

Lab Report Status

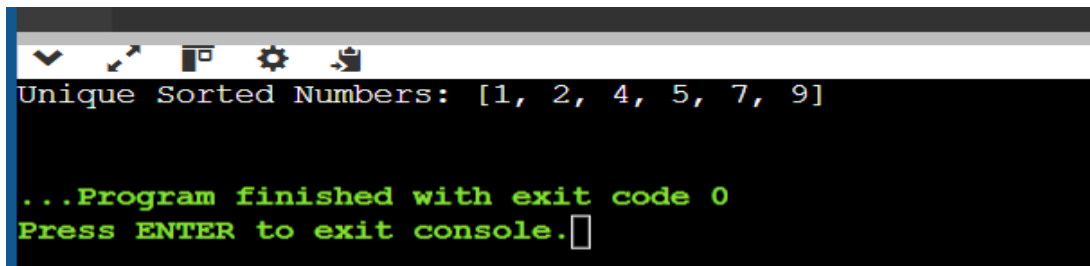
Marks: Signature:.....

Comments:..... Date:.....

1. List: Remove duplicates and sort in ascending order

CODE:

```
numbers = [4, 2, 7, 2, 5, 7, 9, 1]
unique_sorted_numbers = sorted(set(numbers))
print("Unique Sorted Numbers:", unique_sorted_numbers)
```

A screenshot of a terminal window with a dark background. The top toolbar shows icons for a dropdown menu, a cursor, a window, a gear, and a clipboard. The terminal text shows the output of the first code block: "Unique Sorted Numbers: [1, 2, 4, 5, 7, 9]" in a light blue font. Below this, in green text, it says "...Program finished with exit code 0" and "Press ENTER to exit console." with a cursor at the end of the line.

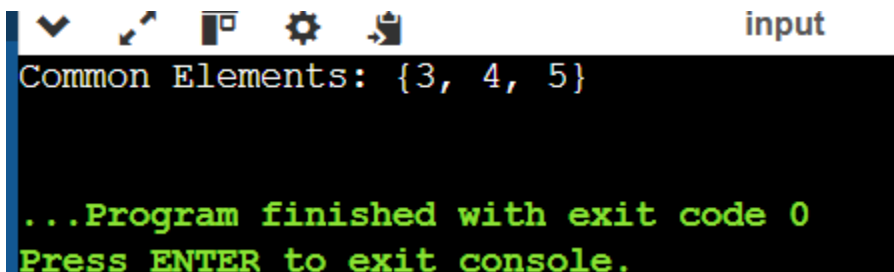
```
Unique Sorted Numbers: [1, 2, 4, 5, 7, 9]

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

2. Set: Find common elements between two lists using sets

CODE:

```
list1 = {1, 2, 3, 4, 5}
list2 = {3, 4, 5, 6, 7}
common_elements = list1.intersection(list2)
print("Common Elements:", common_elements)
```

A screenshot of a terminal window with a dark background. The top toolbar shows icons for a dropdown menu, a cursor, a window, a gear, and a clipboard. The word "input" is written in blue above the toolbar. The terminal text shows the output of the second code block: "Common Elements: {3, 4, 5}" in a light blue font. Below this, in green text, it says "...Program finished with exit code 0" and "Press ENTER to exit console." with a cursor at the end of the line.

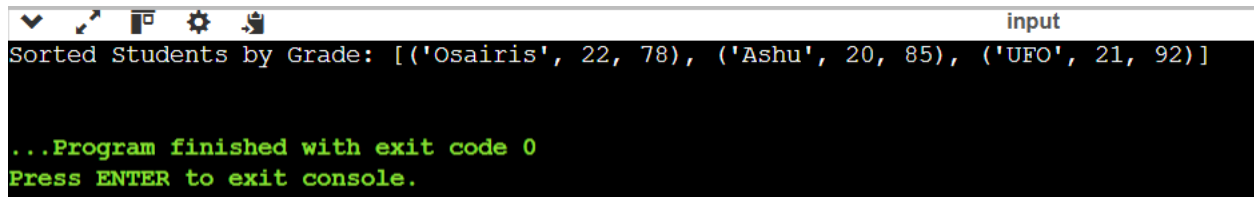
```
Common Elements: {3, 4, 5}

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

3. Tuple: Create a tuple of student records and sort by grade

CODE:

```
students = [("Ashu", 20, 85), ("Osairis", 22, 78), ("UFO", 21, 92)]
sorted_students = sorted(students, key=lambda x: x[2])
print("Sorted Students by Grade:", sorted_students)
```

A screenshot of a terminal window with a dark background. The title bar at the top shows standard window controls and the word "input". The terminal displays the output of a program: "Sorted Students by Grade: [('Osairis', 22, 78), ('Ashu', 20, 85), ('UFO', 21, 92)]". Below this, it shows "...Program finished with exit code 0" and "Press ENTER to exit console." in green text.

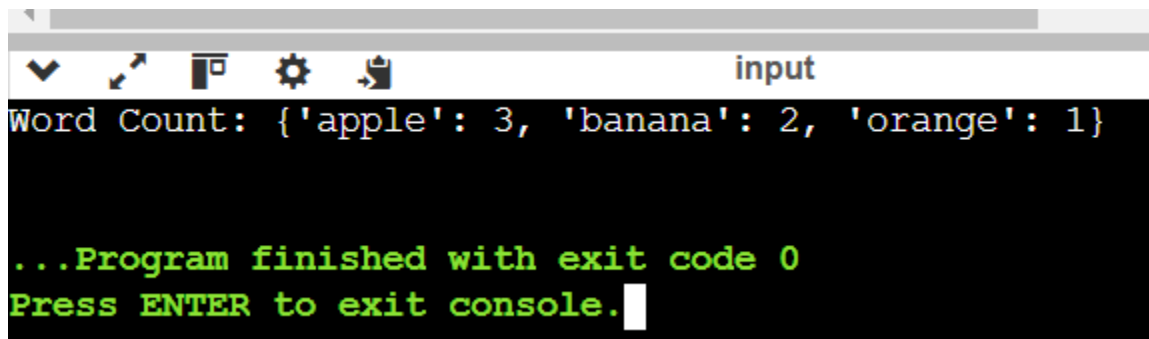
```
Sorted Students by Grade: [('Osairis', 22, 78), ('Ashu', 20, 85), ('UFO', 21, 92)]

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

4. Dictionary: Count word occurrences in a given text

CODE:

```
text = "apple banana apple orange banana apple"
words = text.split()
word_count = {word: words.count(word) for word in set(words)}
print("Word Count:", word_count)
```

A screenshot of a terminal window with a dark background. The title bar at the top shows standard window controls and the word "input". The terminal displays the output of a program: "Word Count: {'apple': 3, 'banana': 2, 'orange': 1}". Below this, it shows "...Program finished with exit code 0" and "Press ENTER to exit console." in green text, with a white cursor at the end of the last line.

```
Word Count: {'apple': 3, 'banana': 2, 'orange': 1}

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

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

CODE:

```
import numpy as np

matrix = np.random.randint(1, 100, (5, 5))
row_sums = matrix.sum(axis=1)

print("Random Matrix:\n", matrix)
print("Row-wise Sums:", row_sums)
```

```
input
Random Matrix:
[[99 20 93 46 43]
 [44 32 13 50 93]
 [14 91 77 66  6]
 [61 16 14 74 91]
 [52 65 42 94  7]]
Row-wise Sums: [301 232 254 256 260]

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

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

CODE:

```
import numpy as np
random_values = np.random.rand(100)
normalized_values = (random_values - np.min(random_values)) /
(np.max(random_values) - np.min(random_values))

print("Original Random Values:\n", random_values)
print("\nNormalized Values:\n", normalized_values)
```

```
Original Random Values:
[0.33879906 0.35500013 0.24412896 0.06058485 0.31319658 0.55478577
 0.35859621 0.30940349 0.2295498  0.56272426 0.30625823 0.44756832
 0.81156355 0.97799738 0.38380803 0.10023212 0.74280624 0.27353293
 0.8014956  0.99429853 0.09139166 0.27786343 0.4430724  0.86268526
 0.36351528 0.20891112 0.83342192 0.47066888 0.49059173 0.43502135
 0.37480371 0.81665869 0.56339837 0.68886948 0.04335421 0.52069415
 0.38269013 0.56335946 0.21431815 0.705583  0.90444268 0.87967516
 0.59117945 0.30988519 0.75579262 0.17918995 0.91683912 0.45441816
 0.80653001 0.41793734 0.05798517 0.32593623 0.3448017  0.46335937
 0.90547506 0.26107423 0.21647919 0.4193479  0.30192282 0.86840056
 0.60498194 0.07800397 0.46551826 0.24757923 0.36397415 0.04926081
 0.30365049 0.57307916 0.3514198  0.64969306 0.05319933 0.53986146
 0.32619784 0.94839975 0.59266521 0.31702736 0.30589996 0.07129516
 0.64483048 0.40928918 0.64645851 0.75048632 0.80050433 0.34801679
 0.57910239 0.90777122 0.99992608 0.06544866 0.85002797 0.20279937
 0.75725493 0.38827211 0.53483072 0.36909921 0.8412479  0.02045214
 0.03123611 0.00790747 0.31236486 0.11332893]

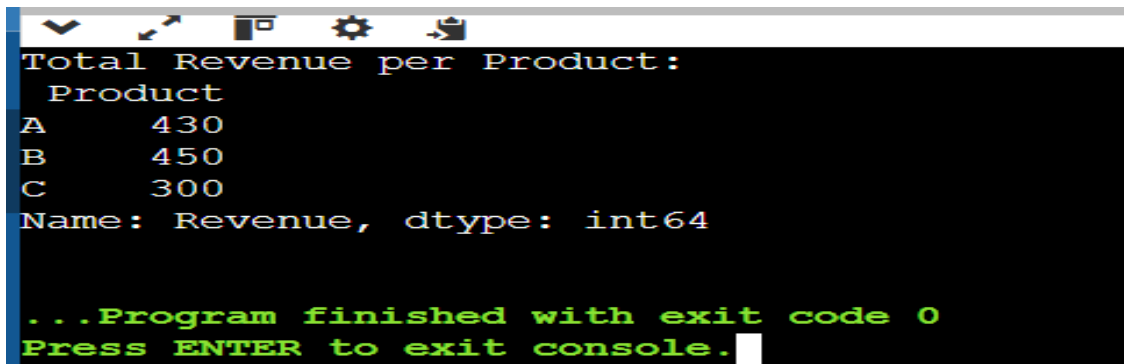
Normalized Values:
[0.33355382 0.34988523 0.23812204 0.05310121 0.30774535 0.55127827
 0.35351025 0.30392174 0.22342558 0.55928063 0.30075118 0.44319819
 0.81012198 0.97789487 0.37892491 0.09306746 0.74081148 0.26776258
 0.79997303 0.99432717 0.08415587 0.27212792 0.4386661  0.861655
 0.3584689  0.20262085 0.83215621 0.46648461 0.48656775 0.43055027
 0.36984815 0.81525812 0.55996016 0.68644076 0.03573193 0.51691236
 0.37779802 0.55992094 0.20807138 0.70328875 0.90374838 0.87878159
 0.58796475 0.30440732 0.75390234 0.17266055 0.91624455 0.45010315
 0.80504795 0.41332881 0.0504806  0.3205875  0.33960476 0.45911629
 0.90478907 0.25520364 0.21024981 0.41475072 0.29638089 0.86741628
 0.6018783  0.07066047 0.46129254 0.24160007 0.35893145 0.04168605
 0.29812246 0.56971884 0.3462761  0.64694915 0.04565626 0.53623388
 0.32085121 0.94805912 0.58946248 0.31160695 0.30039002 0.06389768
 0.64204744 0.40461107 0.64368857 0.74855335 0.79897379 0.34284571
 0.57579053 0.9071037  1.         0.05800415 0.84889587 0.19645992
 0.75537642 0.38342491 0.53116267 0.36409774 0.84004516 0.0126456
 0.02351633 0.         0.30690694 0.10626965]
```

7. Pandas#1: Load a CSV file of sales data and compute total revenue per product

CODE:

```
import pandas as pd
data = {
    'Product': ['A', 'B', 'A', 'C', 'B', 'A'],
    'Revenue': [100, 200, 150, 300, 250, 180]
}
sales_df = pd.DataFrame(data)
total_revenue = sales_df.groupby('Product')['Revenue'].sum()

print("Total Revenue per Product:\n", total_revenue)
```



```
Total Revenue per Product:
Product
A      430
B      450
C      300
Name: Revenue, dtype: int64

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

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

CODE:

```
import pandas as pd
import numpy as np
data = {'A': [1, np.nan, 3], 'B': [4, 5, np.nan]}
df = pd.DataFrame(data)
df_filled = df.fillna(df.mean())
print("Original DataFrame with Missing Values:\n", df)
print("\nDataFrame after Filling Missing Values:\n", df_filled)
```

```

Original DataFrame with Missing Values:
   A    B
0  1.0  4.0
1  NaN  5.0
2  3.0  NaN

DataFrame after Filling Missing Values:
   A    B
0  1.0  4.0
1  2.0  5.0
2  3.0  4.5

...Program finished with exit code 0

```

9. Matplotlib#1: Plot a line graph showing temperature variations over a week

code:

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

```
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
```

```
temperatures = [22, 24, 19, 23, 25, 21, 20]
```

```
plt.figure(figsize=(8,5))
```

```
plt.plot(days, temperatures, marker='o', linestyle='-', color='b')
```

```
plt.xlabel("Days")
```

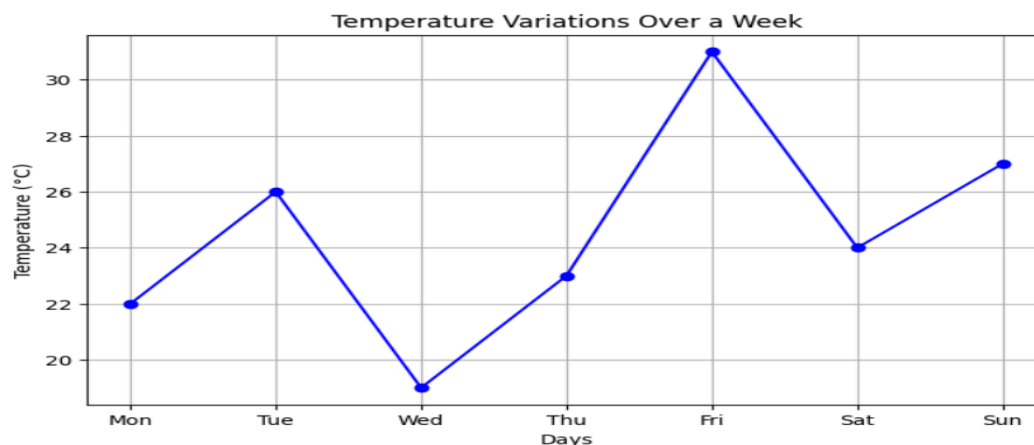
```
plt.ylabel("Temperature (°C)")
```

```
plt.title("Temperature Variations Over a Week")
```

```
plt.grid(True)
```

```
plt.show()
```

→



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

CODE:

```
import matplotlib.pyplot as plt
regions = ["North", "South", "East", "West"]
sales_revenue = [25000, 30000, 22000, 27000]
plt.figure(figsize=(8,5))
plt.bar(regions, sales_revenue, color=['blue', 'red', 'green',
'purple'])
plt.xlabel("Regions")
plt.ylabel("Sales Revenue ($)")
plt.title("Sales Revenue Across Regions")
plt.grid(axis='y', linestyle='--')
plt.show()
```

