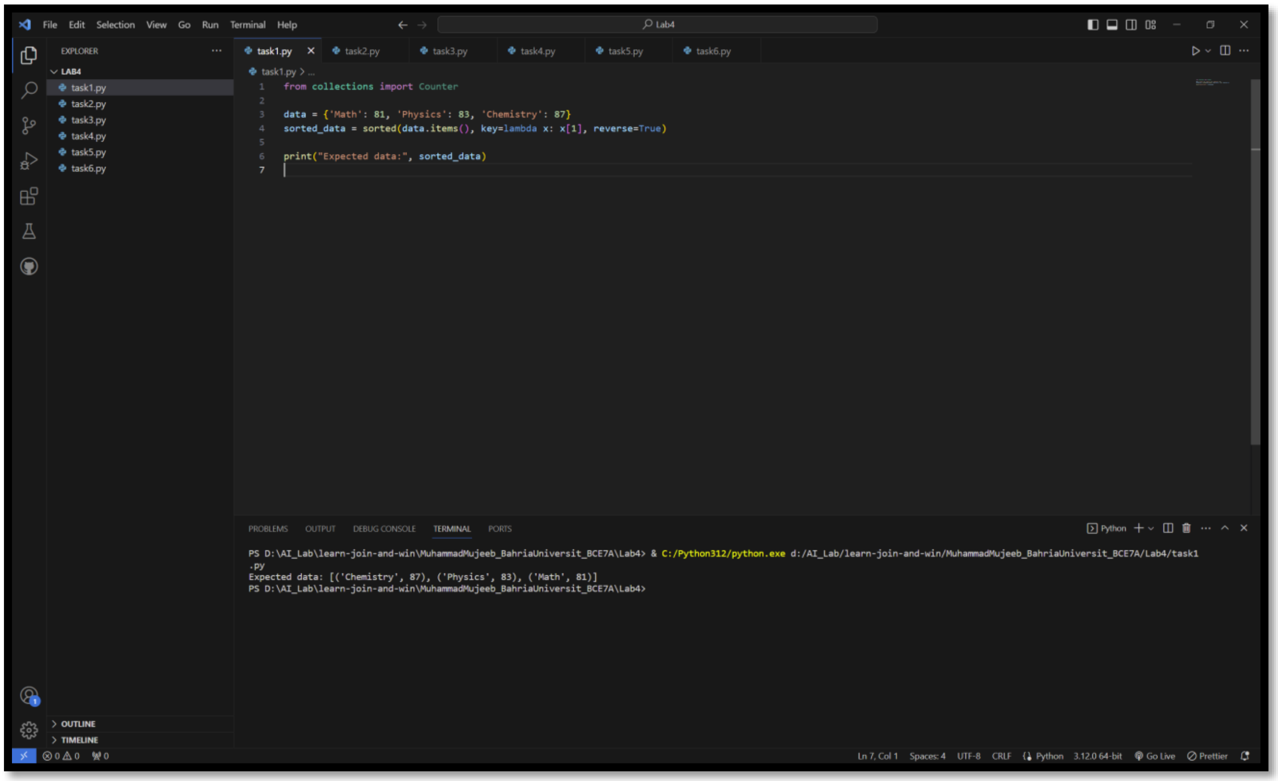
MUHAMMAD MUJEEB 02-132202-017

Lab # 4

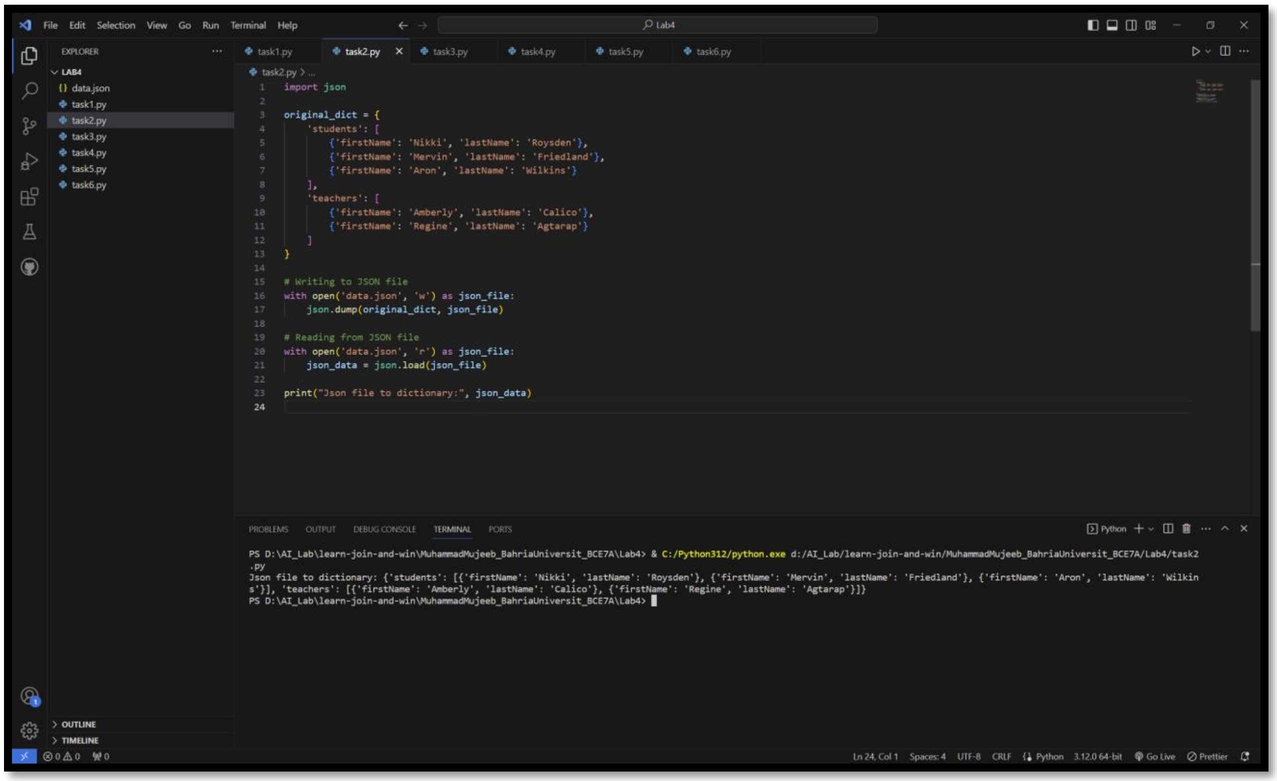
1. Write a Python program to sort Counter by value.
   Sample data: {'Math':81, 'Physics':83, 'Chemistry':87}
   Expected data: [('Chemistry', 87), ('Physics', 83), ('Math', 81)]



```python
from collections import Counter

data = {'Math': 81, 'Physics': 83, 'Chemistry': 87}
sorted_data = sorted(data.items(), key=lambda x: x[1], reverse=True)

print("Expected data:", sorted_data)
```

```
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4> & C:/Python312/python.exe d:/AI_Lab/learn-join-and-win/MuhammadMujeeb_BahriaUniversit_BCE7A/Lab4/task1
.py
Expected data: [('Chemistry', 87), ('Physics', 83), ('Math', 81)]
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4>
```

2. Write a Python program to store dictionary data in a JSON file.
   Original dictionary:
   {'students': [{'firstName': 'Nikki', 'lastName': 'Roysden'}, {'firstName': 'Mervin', 'lastName': 'Friedland'}, {'firstName': 'Aron ', 'lastName': 'Wilkins'}], 'teachers': [{'firstName': 'Amberly', 'lastName': 'Calico'}, {'firstName': 'Regine', 'lastName': 'Agtarap'}]}
   <class 'dict'>
   Json file to dictionary:
   {'students': [{'firstName': 'Nikki', 'lastName': 'Roysden'}, {'firstName': 'Mervin', 'lastName': 'Friedland'}, {'firstName': 'Aron ', 'lastName': 'Wilkins'}], 'teachers': [{'firstName': 'Amberly', 'lastName': 'Calico'}, {'firstName': 'Regine', 'lastName': 'Agtarap'}]}

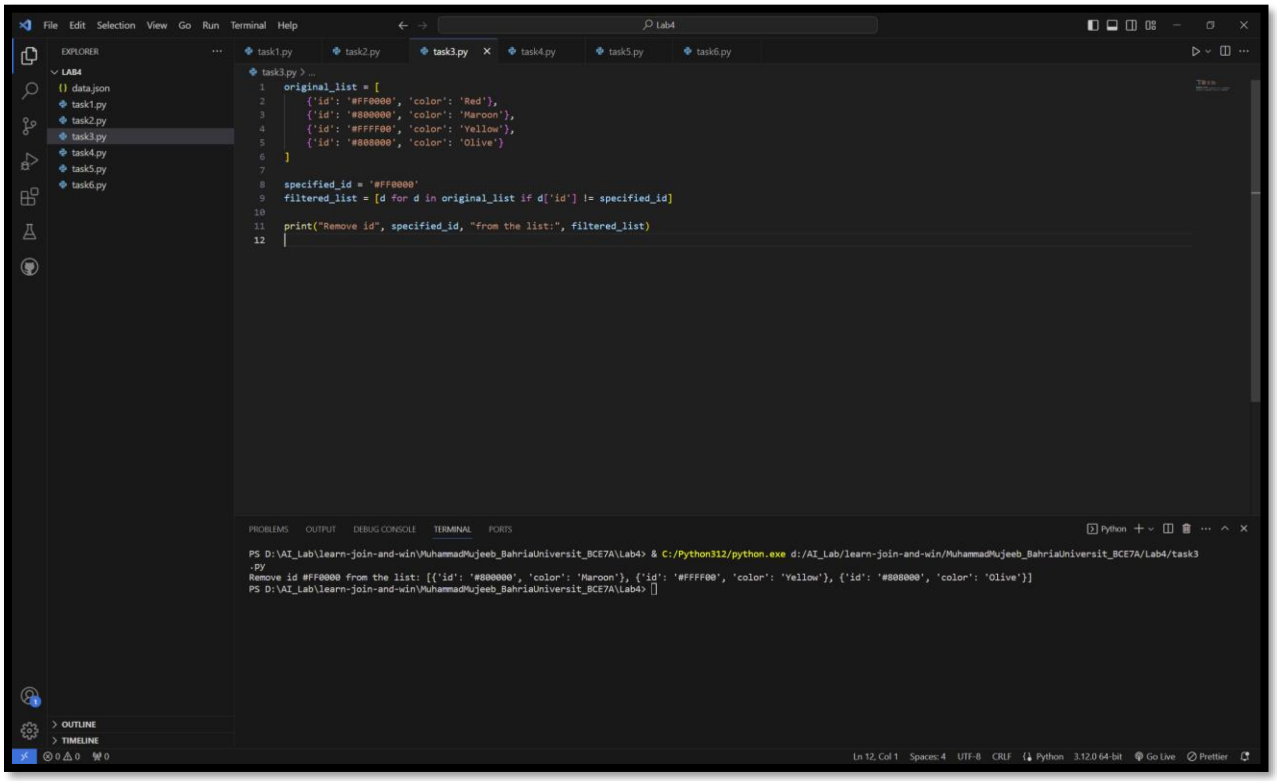


```python
import json

original_dict = {
    'students': [
        {'firstName': 'Nikki', 'lastName': 'Roysden'},
        {'firstName': 'Mervin', 'lastName': 'Friedland'},
        {'firstName': 'Aron', 'lastName': 'Wilkins'}
    ],
    'teachers': [
        {'firstName': 'Amberly', 'lastName': 'Calico'},
        {'firstName': 'Regine', 'lastName': 'Agtarap'}
    ]
}

# Writing to JSON file
with open('data.json', 'w') as json_file:
    json.dump(original_dict, json_file)

# Reading from JSON file
with open('data.json', 'r') as json_file:
    json_data = json.load(json_file)

print("Json file to dictionary:", json_data)
```

```
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4> & C:/Python312/python.exe d:/AI_Lab/learn-join-and-win/MuhammadMujeeb_BahriaUniversit_BCE7A/Lab4/task2
.py
Json file to dictionary: {'students': [{'firstName': 'Nikki', 'lastName': 'Roysden'}, {'firstName': 'Mervin', 'lastName': 'Friedland'}, {'firstName': 'Aron', 'lastName': 'Wilkin
s'}], 'teachers': [{'firstName': 'Amberly', 'lastName': 'Calico'}, {'firstName': 'Regine', 'lastName': 'Agtarap'}]}
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4>
```

MUHAMMAD MUJEEB 02-132202-017

3. **.** Write a Python program to remove a specified dictionary from a given list.
   Original list of dictionary:
   [{'id': '#FF0000', 'color': 'Red'}, {'id': '#800000', 'color': 'Maroon'}, {'id': '#FFFF00', 'color': 'Yellow'}, {'id': '#808000', 'color': 'Olive'}]
   Remove id #FF0000 from the said list of dictionary:
   [{'id': '#800000', 'color': 'Maroon'}, {'id': '#FFFF00', 'color': 'Yellow'}, {'id': '#808000', 'color': 'Olive'}]

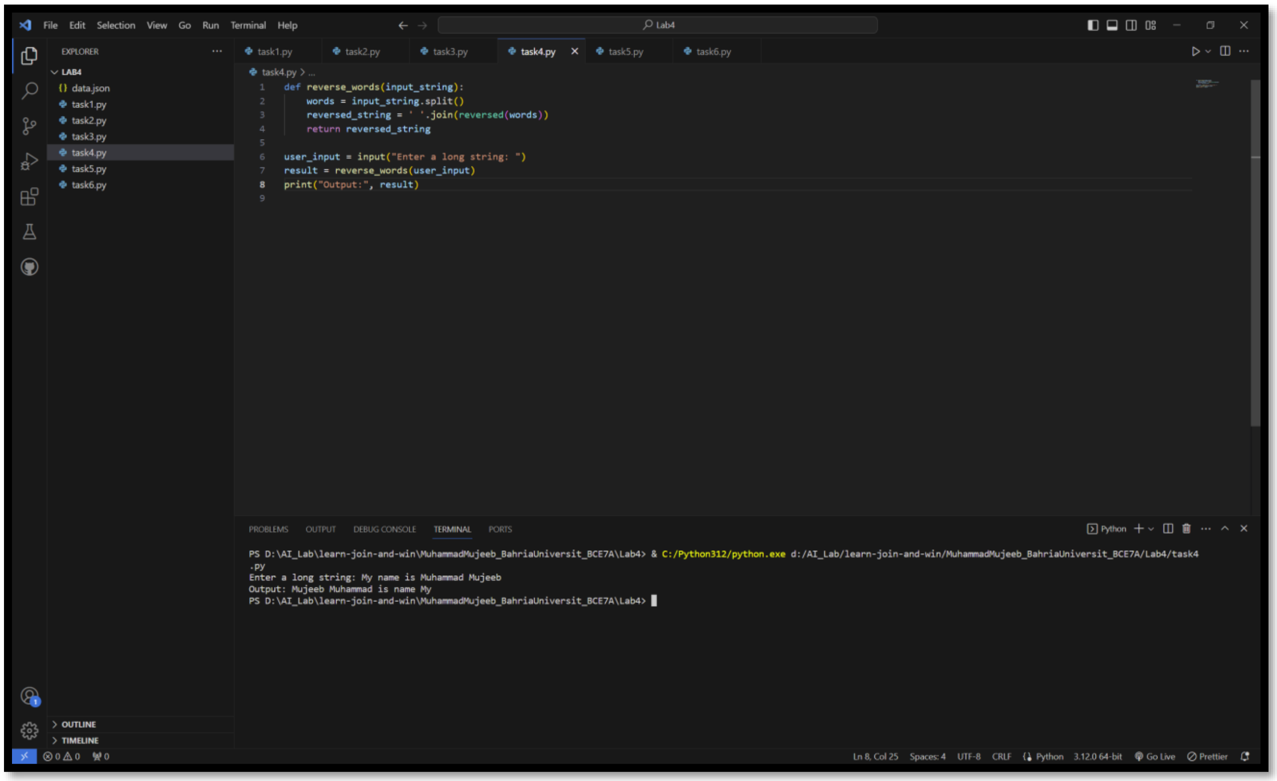


```python
original_list = [
    {'id': '#FF0000', 'color': 'Red'},
    {'id': '#800000', 'color': 'Maroon'},
    {'id': '#FFFF00', 'color': 'Yellow'},
    {'id': '#808000', 'color': 'Olive'}
]

specified_id = '#FF0000'
filtered_list = [d for d in original_list if d['id'] != specified_id]

print("Remove id", specified_id, "from the list:", filtered_list)
```

```
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4> & C:/Python312/python.exe d:/AI_Lab/learn-join-and-win/MuhammadMujeeb_BahriaUniversit_BCE7A/Lab4/task3
.py
Remove id #FF0000 from the list: [{'id': '#800000', 'color': 'Maroon'}, {'id': '#FFFF00', 'color': 'Yellow'}, {'id': '#808000', 'color': 'Olive'}]
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4>
```

<u>**MUHAMMAD MUJEEB 02-132202-017**</u>

4. Write a program (using functions!) that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order. For example, Input: I live in Pakistan. Output: Pakistan in live I.

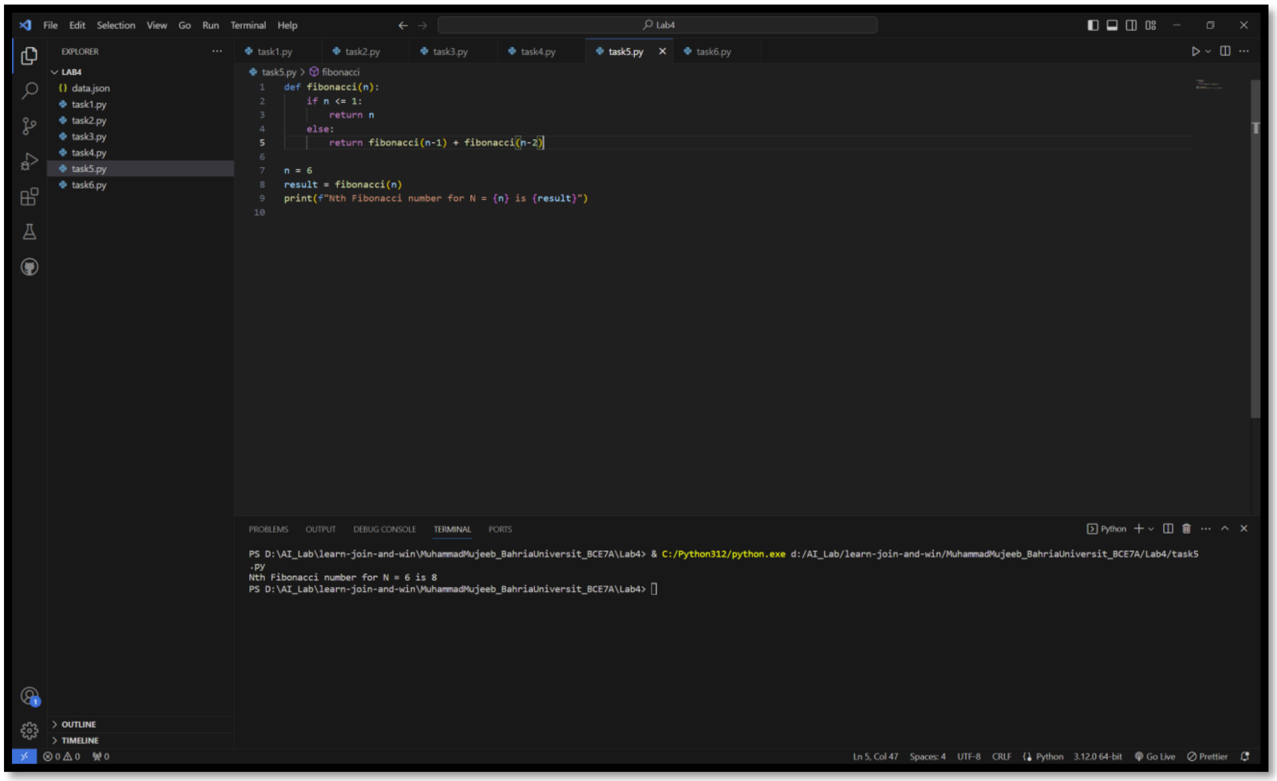


```python
def reverse_words(input_string):
    words = input_string.split()
    reversed_string = ' '.join(reversed(words))
    return reversed_string

user_input = input("Enter a long string: ")
result = reverse_words(user_input)
print("Output:", result)
```

```
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4> & C:/Python312/python.exe d:/AI_Lab/learn-join-and-win/MuhammadMujeeb_BahriaUniversit_BCE7A/Lab4/task4
.py
Enter a long string: My name is Muhammad Mujeeb
Output: Mujeeb Muhammad is name My
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4>
```

5. Write a recursive function to compute Ntn Fibonacci number. Test and trace for N = 6 is 8. We remember that a Fibonacci number can be recursively defined as:

   $F(n) = F(n-1) + F(n-2)$ for $n \geq 2$, where $F(0) = 0, F(1) = 1$.

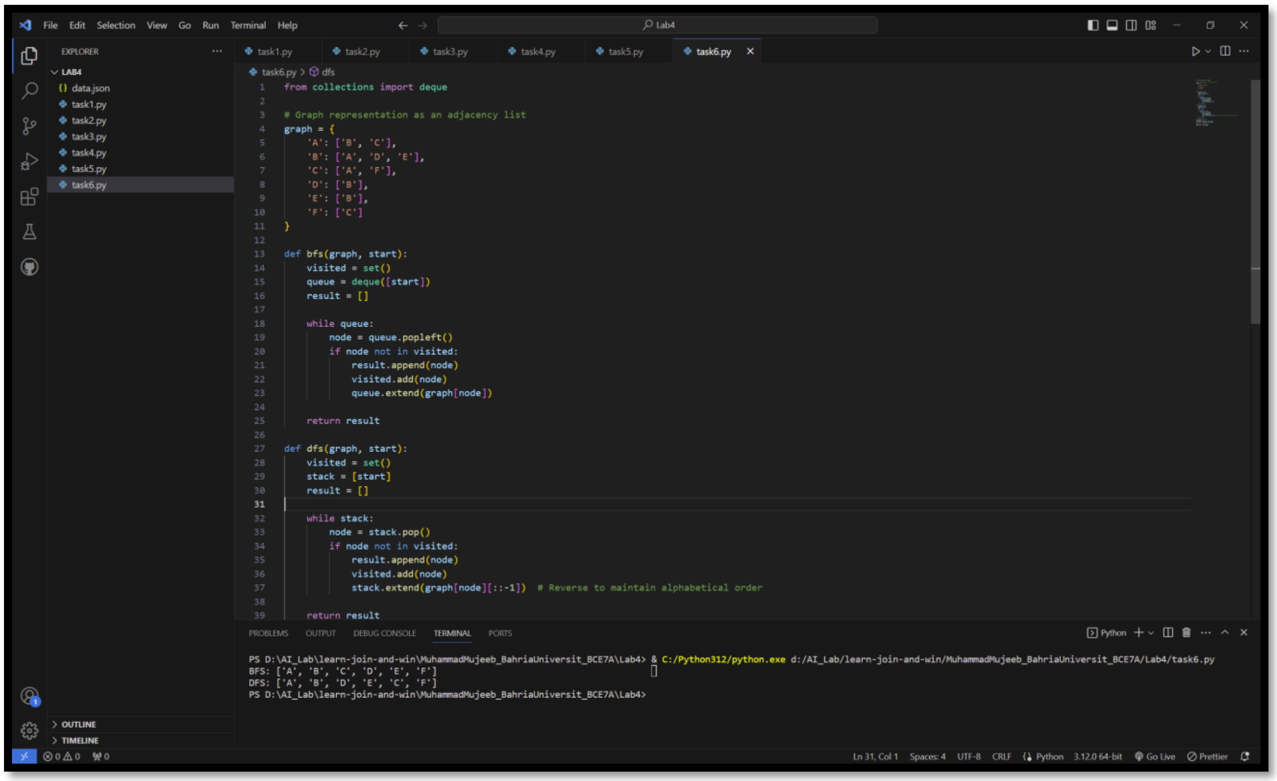


```python
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

n = 6
result = fibonacci(n)
print(f"Nth Fibonacci number for N = {n} is {result}")
```

```
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4> & C:/Python312/python.exe d:/AI_Lab/learn-join-and-win/MuhammadMujeeb_BahriaUniversit_BCE7A/Lab4/task5
.py
Nth Fibonacci number for N = 6 is 8
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4> 
```

MUHAMMAD MUJEEB 02-132202-017

6. Implement BFS & DFS in python as describes in the class.



```python
from collections import deque

# Graph representation as an adjacency list
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B'],
    'F': ['C']
}

def bfs(graph, start):
    visited = set()
    queue = deque([start])
    result = []

    while queue:
        node = queue.popleft()
        if node not in visited:
            result.append(node)
            visited.add(node)
            queue.extend(graph[node])

    return result

def dfs(graph, start):
    visited = set()
    stack = [start]
    result = []

    while stack:
        node = stack.pop()
        if node not in visited:
            result.append(node)
            visited.add(node)
            stack.extend(graph[node][::-1])  # Reverse to maintain alphabetical order

    return result
```

```
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4> & C:/Python312/python.exe d:/AI_Lab/learn-join-and-win/MuhammadMujeeb_BahriaUniversit_BCE7A/Lab4/task6.py
BFS: ['A', 'B', 'C', 'D', 'E', 'F']
DFS: ['A', 'B', 'D', 'E', 'C', 'F']
PS D:\AI_Lab\learn-join-and-win\MuhammadMujeeb_BahriaUniversit_BCE7A\Lab4>
```