AI ASSISTED CODING LAB

ASSIGNMENT 1.2

ENROLLMENT NO :2503A51L15

BATCH NO: 19

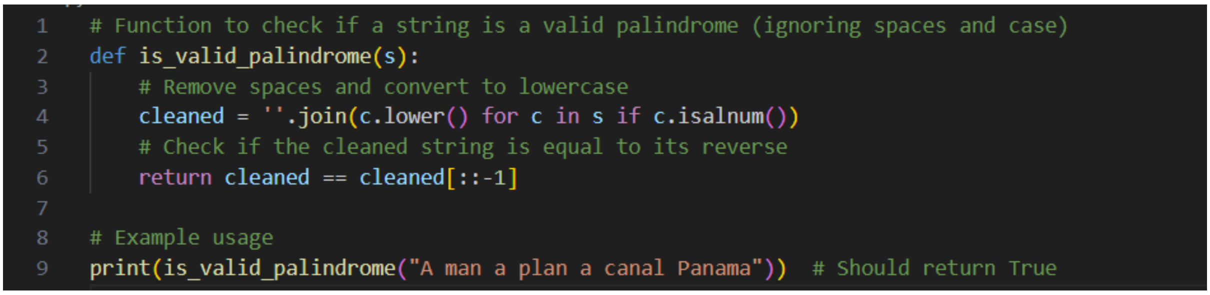
NAME: MOHAMMAD KHAJA AFZALUDDIN

TASK1

TASK1 DESCRIPTION:- Write a comment: # Function to check if a string is a valid palindrome (ignoring spaces and case) and allow Copilot to complete it

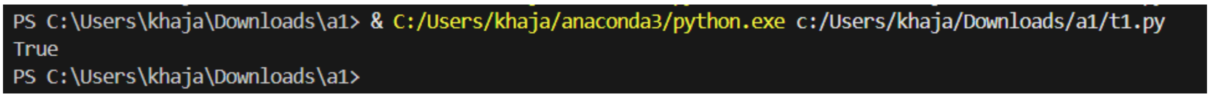
PROMPT:- Generate a comment: # Function to check if a string is a valid palindrome (ignoring spaces and case) and complete it.

CODE:-



```python
# Function to check if a string is a valid palindrome (ignoring spaces and case)
def is_valid_palindrome(s):
    # Remove spaces and convert to lowercase
    cleaned = ''.join(c.lower() for c in s if c.isalnum())
    # Check if the cleaned string is equal to its reverse
    return cleaned == cleaned[::-1]

# Example usage
print(is_valid_palindrome("A man a plan a canal Panama"))  # Should return True
```

OUTPUT:-



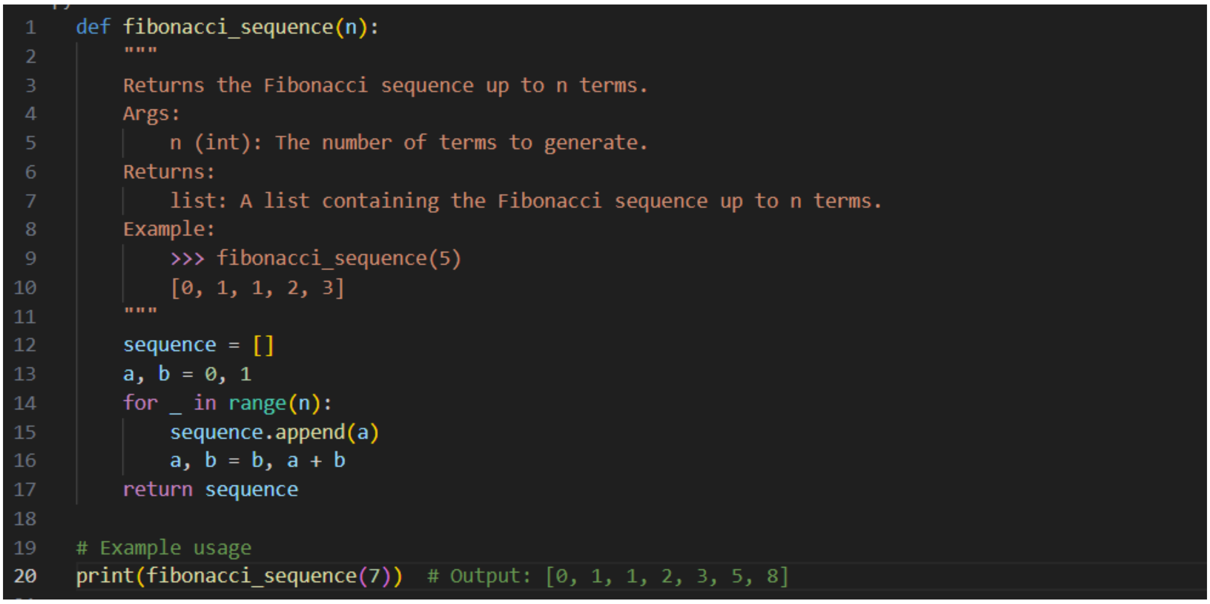
```
PS C:\Users\khaja\Downloads\a1> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/a1/t1.py
True
PS C:\Users\khaja\Downloads\a1>
```

TASK2

TASK1 DESCRIPTION:- Generate a Python function that returns the Fibonacci sequence up to n terms. Prompt with only a function header and docstring

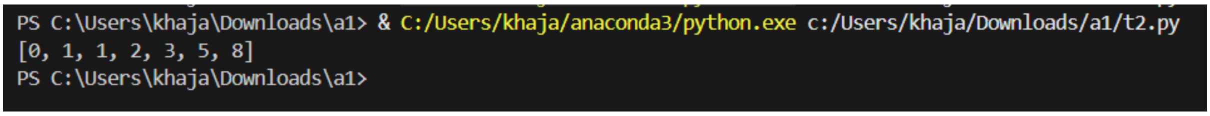
PROMPT:-Generate a Python function that returns the Fibonacci sequence up to n terms. The function should include only the function header and a detailed docstring and should implement with example

CODE:-



```python
def fibonacci_sequence(n):
    """
    Returns the Fibonacci sequence up to n terms.
    Args:
        n (int): The number of terms to generate.
    Returns:
        list: A list containing the Fibonacci sequence up to n terms.
    Example:
        >>> fibonacci_sequence(5)
        [0, 1, 1, 2, 3]
    """
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence

# Example usage
print(fibonacci_sequence(7))  # Output: [0, 1, 1, 2, 3, 5, 8]
```

OUTPUT:-



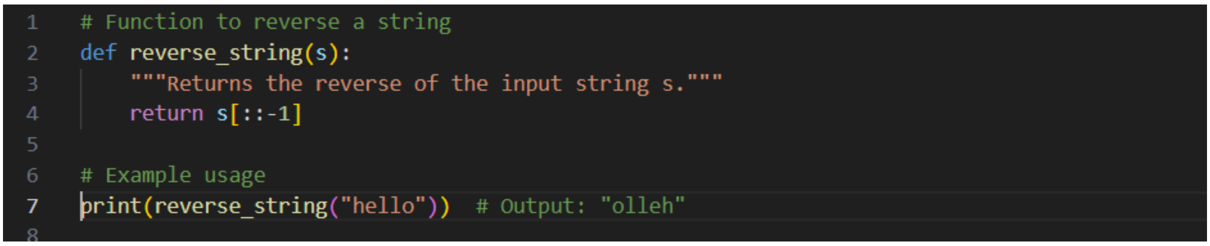
```
PS C:\Users\khaja\Downloads\a1> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/a1/t2.py
[0, 1, 1, 2, 3, 5, 8]
PS C:\Users\khaja\Downloads\a1>
```

TASK3

TASK3 DESCRIPTION:- Write a comment like # Function to reverse a string and use Copilot to generate the function.

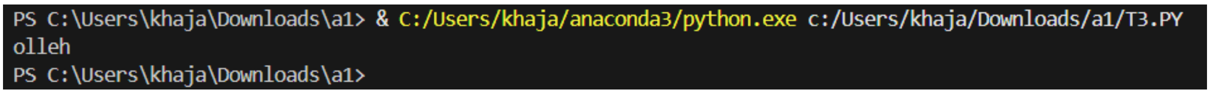
PROMPT:-Write a comment like # Function to reverse a string and generate the function

CODE:-



```
# Function to reverse a string
def reverse_string(s):
    """Returns the reverse of the input string s."""
    return s[::-1]

# Example usage
print(reverse_string("hello"))  # Output: "olleh"
```

OUTPUT:-



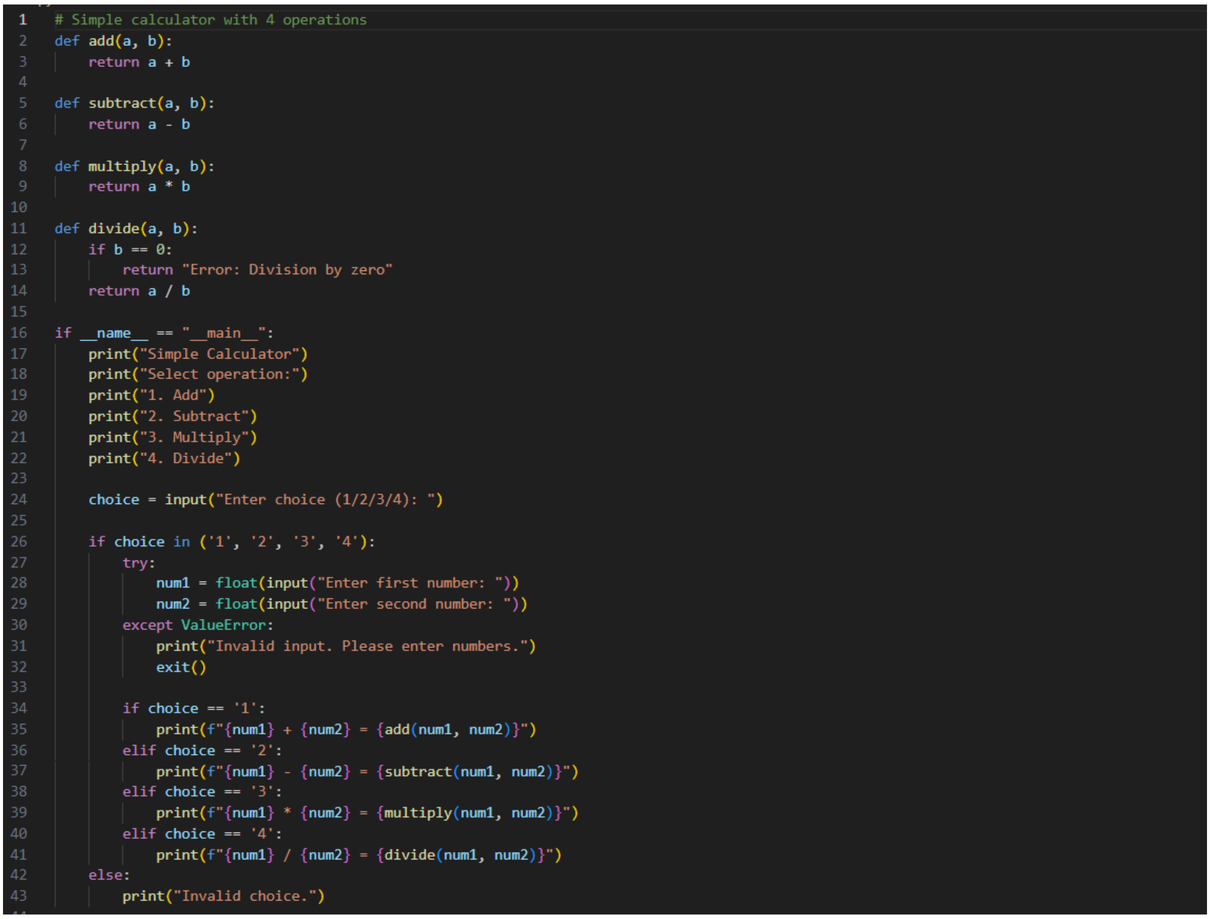
```
PS C:\Users\khaja\Downloads\a1> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/a1/T3.PY
olleh
PS C:\Users\khaja\Downloads\a1>
```

TASK4

TASK4 DESCRIPTION:- Generate a program that simulates a basic calculator (add, subtract, multiply, divide). Write the comment: # Simple calculator with 4 operations and let AI complete it.

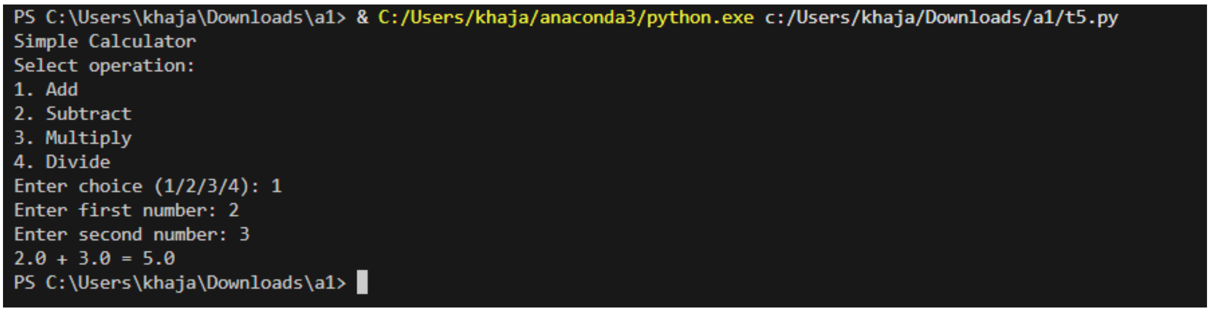
PROMPT:-Write the comment: "# Simple calculator with 4 operations" and let AI complete it. The program should be a fully working calculator with input/output and operator selection logic, supporting add, subtract, multiply, and divide operations.

CODE:-



```python
# Simple calculator with 4 operations
def add(a, b):
    return a + b

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

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

def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b

if __name__ == "__main__":
    print("Simple Calculator")
    print("Select operation:")
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")

    choice = input("Enter choice (1/2/3/4): ")

    if choice in ('1', '2', '3', '4'):
        try:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
        except ValueError:
            print("Invalid input. Please enter numbers.")
            exit()

        if choice == '1':
            print(f"{num1} + {num2} = {add(num1, num2)}")
        elif choice == '2':
            print(f"{num1} - {num2} = {subtract(num1, num2)}")
        elif choice == '3':
            print(f"{num1} * {num2} = {multiply(num1, num2)}")
        elif choice == '4':
            print(f"{num1} / {num2} = {divide(num1, num2)}")
    else:
        print("Invalid choice.")
```

OUTPUT:-



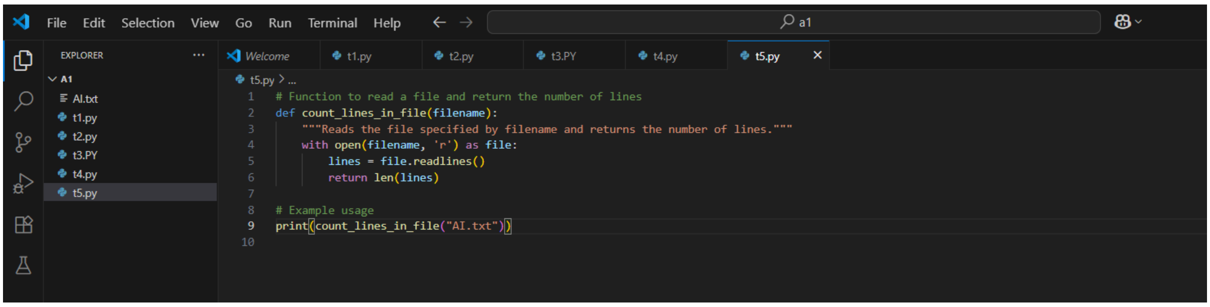
```
PS C:\Users\khaja\Downloads\a1> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/a1/t5.py
Simple Calculator
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice (1/2/3/4): 1
Enter first number: 2
Enter second number: 3
2.0 + 3.0 = 5.0
PS C:\Users\khaja\Downloads\a1> 
```

TASK5

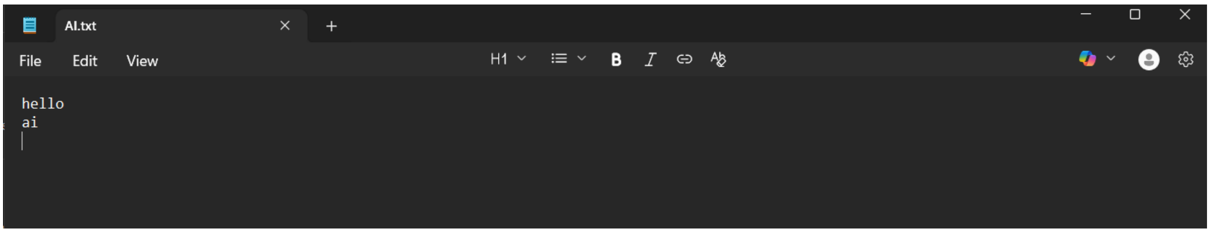
TASK5 DESCRIPTION:- Use a comment to instruct AI to write a function that reads a file and returns the number of lines.

PROMPT:-Write a comment instructing AI to write a function that reads a file and returns the number of lines.

CODE:-

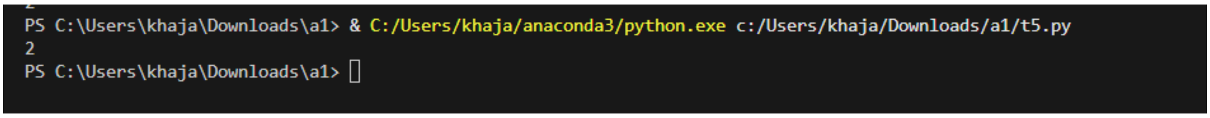


```python
# Function to read a file and return the number of lines
def count_lines_in_file(filename):
    """Reads the file specified by filename and returns the number of lines."""
    with open(filename, 'r') as file:
        lines = file.readlines()
        return len(lines)

# Example usage
print(count_lines_in_file("AI.txt"))
```



```
hello
ai
```

OUTPUT:-



```
PS C:\Users\khaja\Downloads\a1> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/a1/t5.py
2
PS C:\Users\khaja\Downloads\a1> 
```

OBSERVATION:- I explored how **GitHub Copilot** can generate complete and functional Python programs from minimal prompts such as comments, function headers, or docstrings. Each task demonstrated the usefulness of AI-assisted coding in reducing effort, saving time, and improving productivity.

- In **Task 1**, I observed that a simple comment describing the functionality (checking if a string is a palindrome) was enough for Copilot to generate a correct implementation with case and space handling.
- In **Task 2**, providing only a function header and docstring for generating the Fibonacci sequence highlighted how Copilot understands documentation and converts it into working logic.
- In **Task 3**, a short comment about reversing a string allowed Copilot to create a concise and accurate function, showing its ability to handle basic operations efficiently.
- In **Task 4**, using a comment to describe a “simple calculator with four operations” led Copilot to produce a fully functional program with input handling and operator selection, proving its capability to generate interactive applications.
- In **Task 5**, Copilot was able to generate file-handling logic from a single comment, showing its usefulness in automating repetitive tasks like counting lines in a file.