Lab Assignment – 1

G. YESHASHWINI
2303A52023
BATCH-38

Task 1: AI-Generated Logic Without Modularization (Fibonacci Sequence Without Functions)

Use GitHub Copilot to generate a Python program that:

- ➢ **Prints the Fibonacci sequence up to *n* terms**
- ➢ **Accepts user input for *n***
- ➢ **Implements the logic directly in the main code ➢ Does not use any user-defined functions**

Prompt:

write a code to print fibonacci series upto n without using a function

Code:

```
# write a code to print fibonacci series upto n without using a function
n = int(input("Enter a number: "))
a = 0
b = 1
for i in range(n):
    print(a, end=" ")
    c = a + b
    a = b
    b = c
print()
```

Output:

```
Enter a number: 5
0 1 1 2 3
```

Explanation:

This program prints the Fibonacci series.
The user enters a number n. The program starts with 0 and 1. In each loop, it prints the current number and adds the previous two numbers to get the next one. This repeats n times and prints the Fibonacci series.

Task 2: AI Code Optimization & Cleanup (Improving Efficiency)

➢ Examine the Copilot-generated code from Task 1 and improve it by:
➢ Removing redundant variables
➢ Simplifying loop logic
➢ Avoiding unnecessary computations ➢ Use Copilot prompts such as: ▪ "Optimize this Fibonacci code"
➢ "Simplify variable usage"

Prompt:
Optimize the above code by removing unnecessary variables, simplifying the loop, and improving readability.

Code:

```
# Optimize the above code by removing unnecessary variables, simplifying the loop, and improving readability.

n = int(input("Enter the number of terms: "))

if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print(f"Fibonacci series up to {n} term:")
    print(0)
else:
    print(f"Fibonacci series up to {n} terms:")
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
```

Output:

```
Enter the number of terms: 5
Fibonacci series up to 5 terms:
0 1 1 2 3
```

Explanation:

This program prints the Fibonacci series up to a given number of terms.
The user enters a number n. If the number is not valid, the program shows an error message. Otherwise, it starts with 0 and 1 and prints the Fibonacci numbers using a loop. In each step, the next number is found by adding the previous two numbers. The series is printed until n terms are displayed.

Task 3: Modular Design Using AI Assistance (Fibonacci Using Functions) Use GitHub Copilot to generate a function-based Python program that:

- ➢ **Uses a user-defined function to generate Fibonacci numbers**
- ➢ **Returns or prints the sequence up to n**
- ➢ **Includes meaningful comments (AI-assisted)**

Prompt:

Write a code for printing the Fibonacci series up to n terms using a function and include meaningful comments.

Code:

```
# Write a code for printing the Fibonacci series up to n terms using a functionand include meaningful comments.
def fibonacci_series(n):
    """
    This function prints the Fibonacci series up to n terms.

    Parameters:
    n (int): The number of terms in the Fibonacci series to be printed.
    """
    a, b = 0, 1  # Initialize the first two terms of the Fibonacci series
    count = 0    # Initialize a counter to keep track of the number of terms printed

    # Check if the number of terms is valid
    if n <= 0:
        print("Please enter a positive integer.")
    elif n == 1:
        print("Fibonacci series up to", n, "term:")
        print(a)  # Print the first term
    else:
        print("Fibonacci series up to", n, "terms:")
        while count < n:
            print(a, end=' ')  # Print the current term
            a, b = b, a + b   # Update values of a and b to the next two terms
            count += 1        # Increment the counter
# Example usage:
fibonacci_series(10)  # Print the first 10 terms of the Fibonacci serie
```

Output:

```
Fibonacci series up to 10 terms:
0 1 1 2 3 5 8 13 21 34
```

Explanation:

This program prints the Fibonacci series using a function.

The user enters a number n. The function print_fibonacci checks whether the number is valid. If it is valid, the program starts with 0 and 1 and prints Fibonacci numbers by adding the previous two numbers each time. The series is printed up to n terms

Task 4: Comparative Analysis – Procedural vs Modular Fibonacci Code

Compare the Copilot-generated Fibonacci programs:

- **Without functions (Task 1)**
- **With functions (Task 3)**
- **Analyze them in terms of:**
  - **Code clarity**
  - **Reusability**
  - **Debugging ease**
  - **Suitability for larger systems**

Prompt:

Non-Modular: Write a code for printing a Fibonacci series up to n terms without using a function.

Modular: Write a code for printing the Fibonacci series up to n terms using a function and include meaningful comments.

Code:

Procedural:

```
# Optimize the above code by removing unnecessary variables, simplifying the loop,and improving readability.
n = int(input("Enter the number of terms: "))
if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print(f"Fibonacci series up to {n} term:")
    print(0)
else:
    print(f"Fibonacci series up to {n} terms:")
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
```

Output:

```
Enter the number of terms: 8
Fibonacci series up to 8 terms:
0 1 1 2 3 5 8 13
```

Modular:

```python
# Write a code for printing the Fibonacci series up to n terms using a function
# and include meaningful comments.

def print_fibonacci(n):
    """
    Prints the Fibonacci series up to n terms.

    Args:
        n (int): Number of terms to print in the Fibonacci series

    The Fibonacci series starts with 0 and 1, and each subsequent number
    is the sum of the previous two numbers.
    """

    if n <= 0:
        print("Please enter a positive integer")
        return

    elif n == 1:
        print(f"Fibonacci series up to {n} term:")
        print(0)
        return

    print(f"Fibonacci series up to {n} terms:")

    # Initialize the first two terms of the series
    a, b = 0, 1

    # Generate and print each term
    for _ in range(n):
        print(a, end=" ")
        # Update values: a becomes b, b becomes sum of previous a and b
        a, b = b, a + b

    print()  # Print newline at the end


# Get input from user
n = int(input("Enter the number of terms: "))
print_fibonacci(n)
```

Output:

```
Enter the number of terms: 6
Fibonacci series up to 6 terms:
0 1 1 2 3 5
```

Explanation:

In the **procedural approach**, the Fibonacci series code is written directly in the main program without using any function. The steps run one after another, and this method is simple but not reusable.

In the **modular approach**, the Fibonacci logic is written inside a function. The main program only calls the function. This makes the code more organized, easy to understand, and reusable.

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches for Fibonacci Series)

Prompt GitHub Copilot to generate:
An iterative Fibonacci implementation
A recursive Fibonacci implementation

Prompt:

Write a code for printing a Fibonacci series up to n terms without using a function.

Write a code for printing the Fibonacci series up to n terms using recursion.

Code:

```
# Write a code for printing a Fibonacci series up to n terms without using a function

n = int(input("Enter number of terms: "))
a, b = 0, 1
for _ in range(n):
    print(a, end=" ")
    a, b = b, a + b
print()

# Write a code for printing the Fibonacci series up to n terms using recursion

def fibonacci(n, a=0, b=1, count=0):
    if count < n:
        print(a, end=" ")
        fibonacci(n, b, a + b, count + 1)

n = int(input("Enter number of terms: "))
fibonacci(n)
print()
```

Output:

```
Enter number of terms: 5
0 1 1 2 3
```

Explanation:
This program prints the Fibonacci series in two ways .
In the first part, the Fibonacci series is printed **without using a function**. The user enters the number of terms, and the program uses a loop to print the series by adding the previous two numbers each time.
In the second part, the Fibonacci series is printed **using recursion**. A function is defined that prints one Fibonacci number and then calls itself to print the next one. This process continues until the required number of terms is printed. Both methods produce the same Fibonacci series but use different approaches.