**ASSIGNMENT 3.4**

**LAB NAME** : AI Assisted Coding

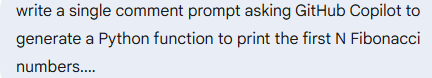
**LAB NUMBER** :3.4

**ROLL NUMBER** :2503a51l05

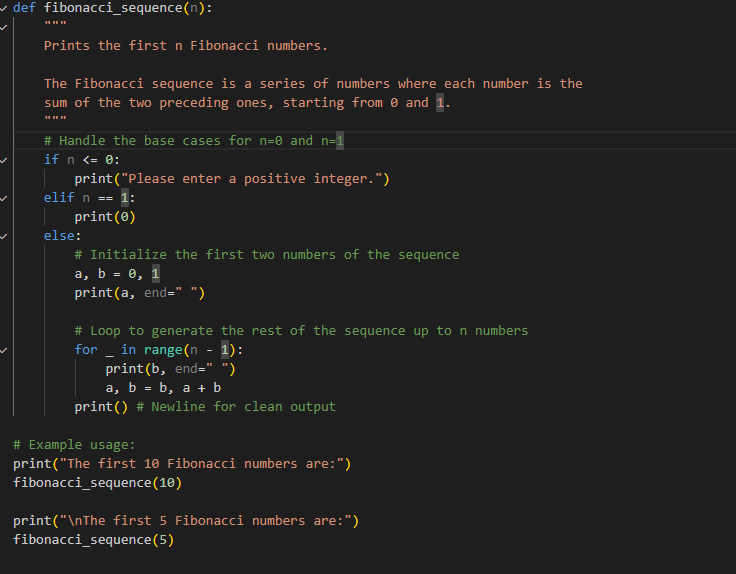
**BATCH NO**  :24BTCAICSB19

**NAME OF STUDENT**: N .SATYA SRI CHARAN

TASK 1

****

**PROMPT :**



CODE :

def fibonacci\_sequence(n):

    """

    Prints the first n Fibonacci numbers.

    The Fibonacci sequence is a series of numbers where each number is the

    sum of the two preceding ones, starting from 0 and 1.

    """

    # Handle the base cases for n=0 and n=1

    if n <= 0:

        print("Please enter a positive integer.")

    elif n == 1:

        print(0)

    else:

        # Initialize the first two numbers of the sequence

        a, b = 0, 1

        print(a, end=" ")

        # Loop to generate the rest of the sequence up to n numbers

        for \_ in range(n - 1):

            print(b, end=" ")

            a, b = b, a + b

        print() # Newline for clean output

# Example usage:

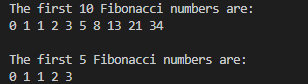
print("The first 10 Fibonacci numbers are:")

fibonacci\_sequence(10)

print("\nThe first 5 Fibonacci numbers are:")

fibonacci\_sequence(5)

**OUTPUT**:

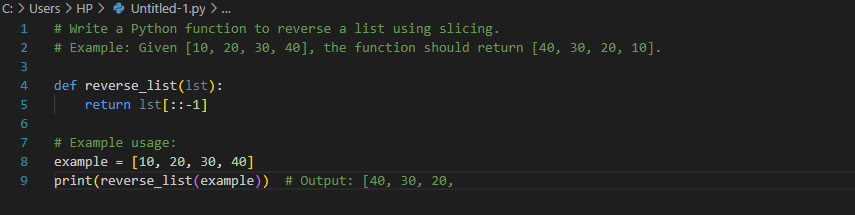


**OBSERVATIONS:**

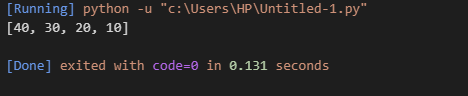
I observation The Fibonacci sequence is a list of numbers that starts with 0 and 1. To get the next number in the list, you just add the two previous numbers together.

**TASKS 2: Screenshot 2025-08-25 190936.png**

**PROMPT :**

****

**OUTPUT:**

****

**CODE:**

# Example: Given [10, 20, 30, 40], the function should return [40, 30, 20, 10].

def reverse\_list(lst):

    return lst[::-1]

# Example usage:

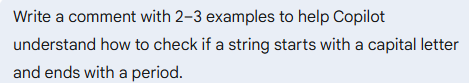
example = [10, 20, 30, 40]

print(reverse\_list(example))  # Output: [40, 30, 20,

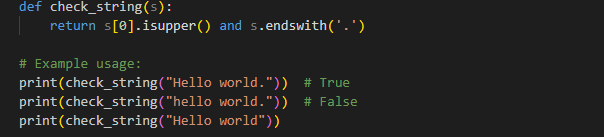
**OBSERVATIONS:**

The function is simple, efficient, and returns a new list in reverse order.

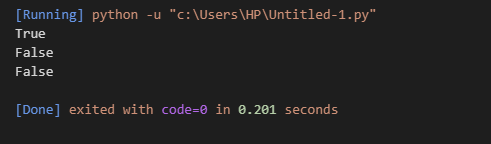
**TASKS 3:**

****

**PROMPT :**

****

**OUTPUT:**

****

**CODE:**

def check\_string(s):

return s[0].isupper() and s.endswith('.')

print(check\_string("Hello world.")) # True

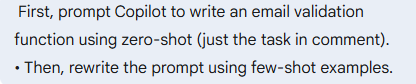
print(check\_string("hello world.")) # False

print(check\_string("Hello world"))

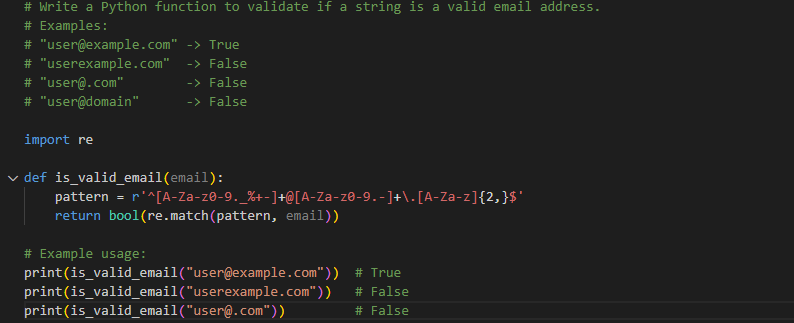
**OBSERVATIONS:**

The function check\_string(s) checks if the first character of the string is uppercase and if the string ends with a period.

**TASKS 4:**



**PROMPT:**



**CODE:**

# Write a Python function to validate if a string is a valid email address.

# Examples:

# "user@example.com" -> True

# "userexample.com" -> False

# "user@.com" -> False

# "user@domain" -> False

import re

def is\_valid\_email(email):

pattern = r'^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'

return bool(re.match(pattern, email))

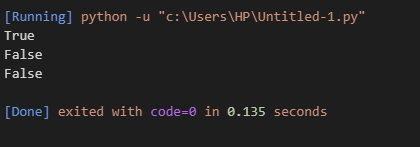
# Example usage:

print(is\_valid\_email("user@example.com")) # True

print(is\_valid\_email("userexample.com")) # False

print(is\_valid\_email("user@.com")) # False

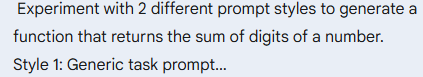
**OUTPUT:**



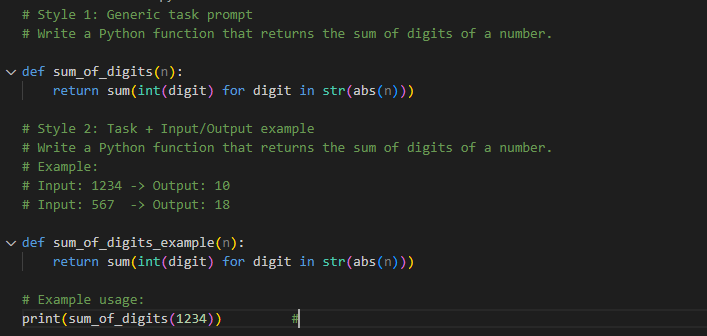
**OBSERVATIONS:**

The function is\_valid\_email(email) uses a regular expression to validate email addresses.

**TASKS 5:**

****

**PROMPT:**

****

**CODE:**

**# Style 1: Generic task prompt**

**# Write a Python function that returns the sum of digits of a number.**

**def sum\_of\_digits(n):**

**return sum(int(digit) for digit in str(abs(n)))**

**# Style 2: Task + Input/Output example**

**# Write a Python function that returns the sum of digits of a number.**

**# Example:**

**# Input: 1234 -> Output: 10**

**# Input: 567 -> Output: 18**

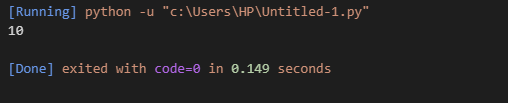
**def sum\_of\_digits\_example(n):**

**return sum(int(digit) for digit in str(abs(n)))**

**# Example usage:**

**print(sum\_of\_digits(1234)) #**

**OUTPUT:**



**OBSERVATIONS:**

Two prompt styles are used to guide Copilot:

Style 1: A generic task prompt asks for a function to return the sum of digits of a number.

Style 2: The prompt includes input/output examples to clarify expected behavior.