ASSIGNMENT-1.4

2303A51546

BATCH-10

TASK-1:

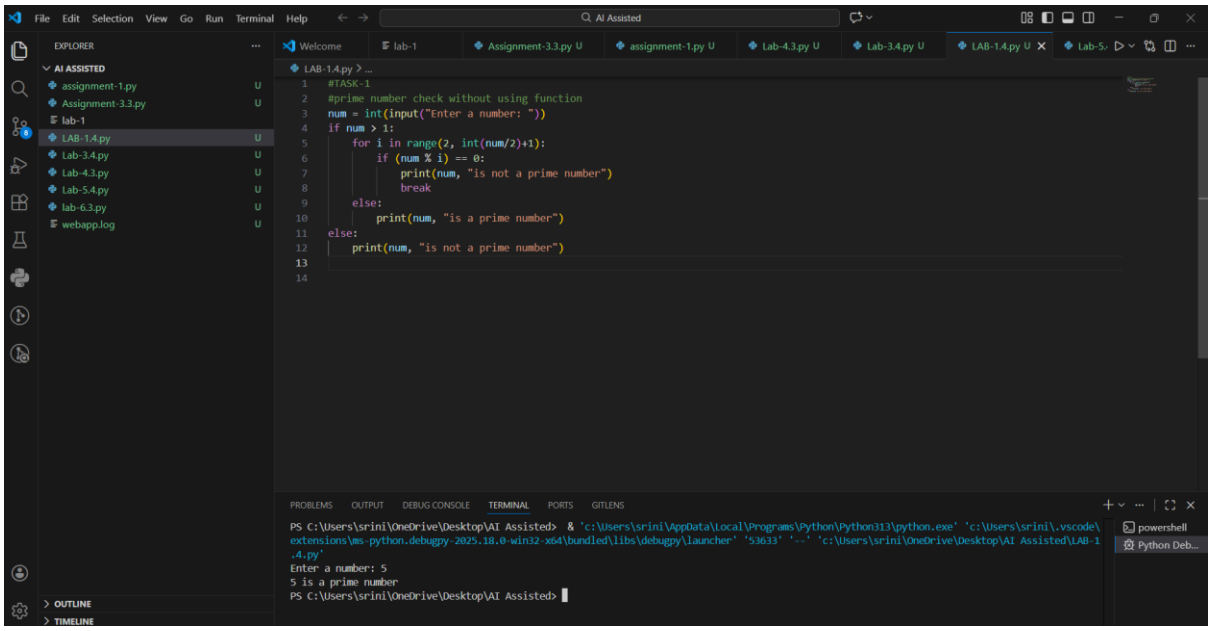
PROMPT:

prime number check without using function

CODE:

```
num = int(input("Enter a number: "))
if num > 1:
   for i in range(2, int(num/2)+1):
      if (num % i) == 0:
         print(num, "is not a prime number")
         break
   else:
      print(num, "is a prime number")
else:
   print(num, "is not a prime number")
```

OUTPUT:



Analysis:

The user enters a number.

The program checks if the number is greater than 1 (since prime numbers are > 1).

It runs a loop from 2 to num/2 to see if any number divides it exactly.

If a divisor is found, it prints not prime and stops the loop using break.

If the loop finishes without finding any divisor, the for-else prints prime.

If the number is ≤ 1, it directly prints not prime.

TASK-2:

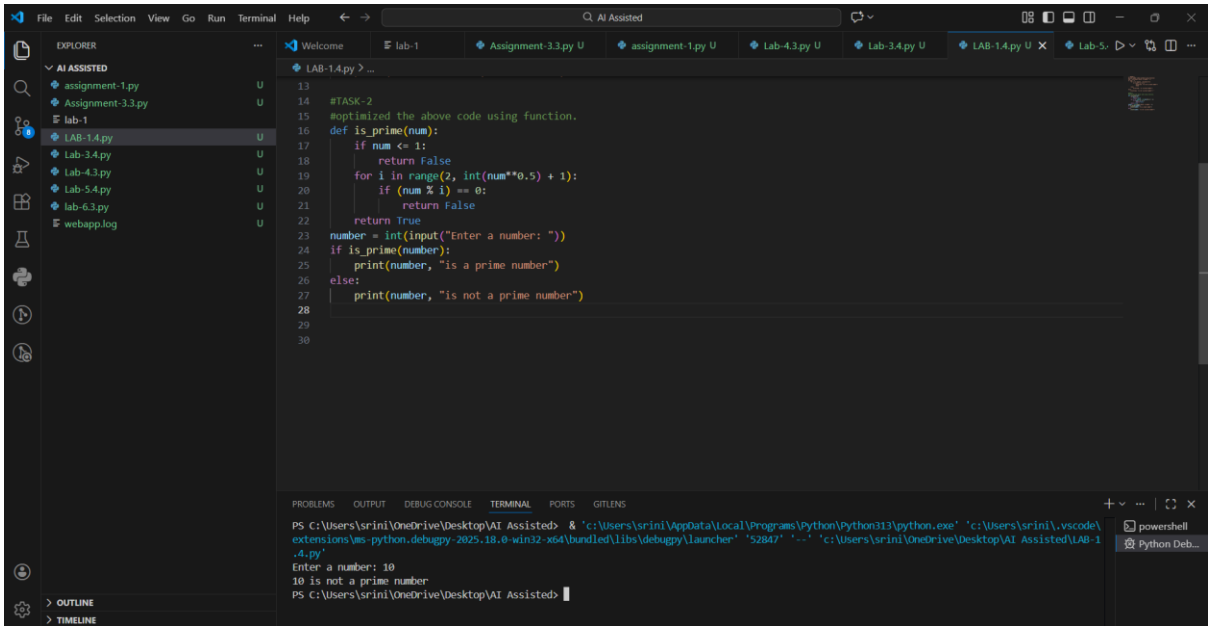
PROMPT:

optimized the above code using function.

CODE:

```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if (num % i) == 0:
            return False
    return True
number = int(input("Enter a number: "))
if is_prime(number):
    print(number, "is a prime number")
else:
    print(number, "is not a prime number")
```

OUTPUT:



Analysis:

A function is_prime(num) is defined.

If number ≤ 1 → returns False.

Loop runs from 2 to √num.

If any divisor is found → returns False.

If no divisor is found → returns True.

Main program takes input, calls the function, and prints result.

TASK-3:

PROMPT:

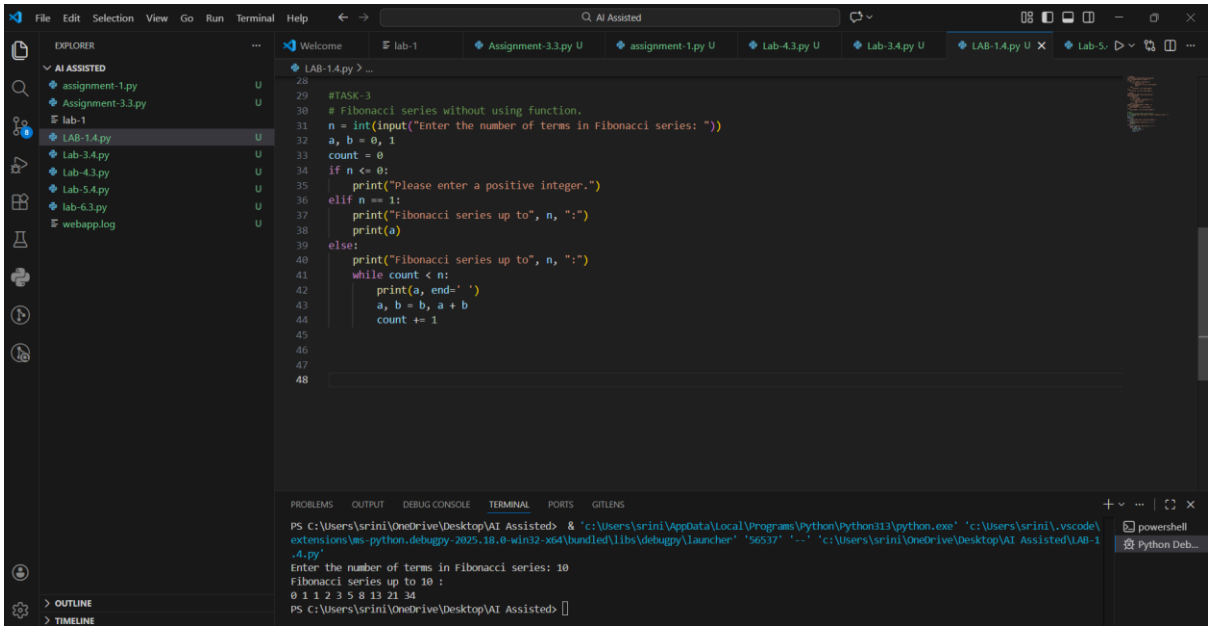
Fibonacci series without using function.

CODE:

```
n = int(input("Enter the number of terms in Fibonacci series: "))
a, b = 0, 1
count = 0
if n <= 0:
   print("Please enter a positive integer.")
elif n == 1:
```

```
    print("Fibonacci series up to", n, ":")
    print(a)
else:
    print("Fibonacci series up to", n, ":")
    while count < n:
        print(a, end=' ')
        a, b = b, a + b
        count += 1
```

OUTPUT:



Analysis:

User enters number of terms n.

If n ≤ 0 → error message.

If n == 1 → prints only 0.

Otherwise, a while loop runs n times:

- Prints current number a
- Updates values using a, b = b, a + b

TASK-4:

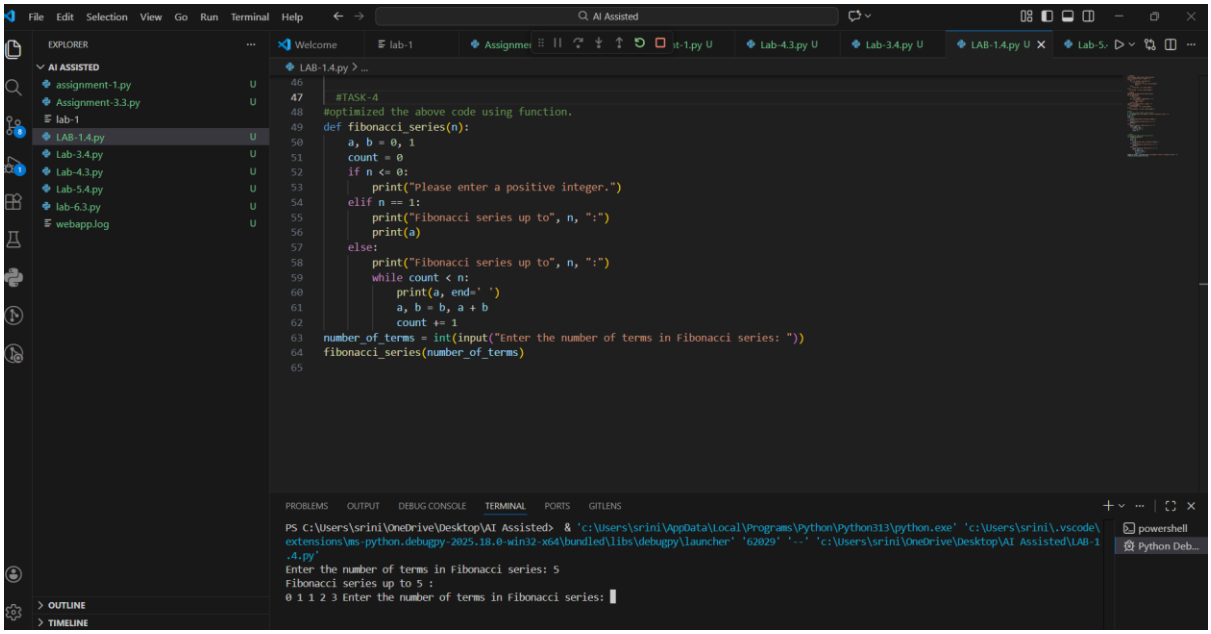
PROMPT:

optimized the above code using function

CODE:

```
def fibonacci_series(n):
    a, b = 0, 1
    count = 0
    if n <= 0:
       print("Please enter a positive integer.")
    elif n == 1:
       print("Fibonacci series up to", n, ":")
       print(a)
    else:
       print("Fibonacci series up to", n, ":")
       while count < n:
          print(a, end=' ')
          a, b = b, a + b
          count += 1
number_of_terms = int(input("Enter the number of terms in Fibonacci series: "))
fibonacci_series(number_of_terms)
```

OUTPUT:



Analysis:

Function fibonacci_series(n) is defined.

It checks conditions for invalid, single-term, or multiple-term cases.

Prints the series inside the function.

Main program takes input and calls the function.

TASK-5:

PROMPT:

#Write a function to find the longest common prefix string amongst an array of strings.

#If there is no common prefix, return an empty string "".

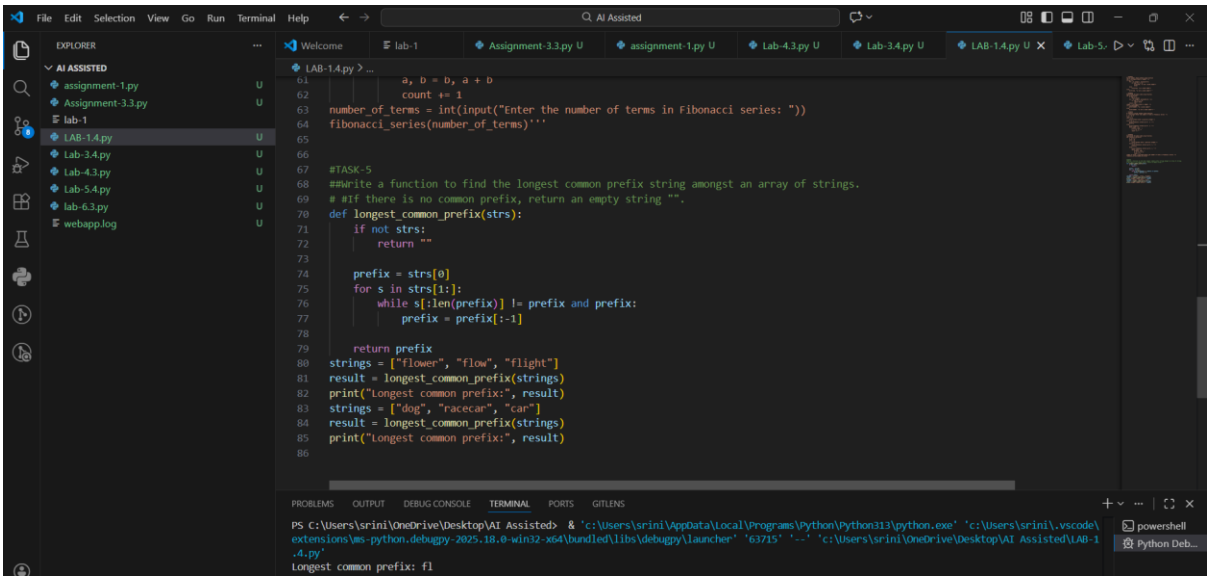
CODE:

```
def longest_common_prefix(strs):
    if not strs:
        return ""

    prefix = strs[0]
    for s in strs[1:]:
        while s[:len(prefix)] != prefix and prefix:
            prefix = prefix[:-1]
```

```
    return prefix
strings = ["flower", "flow", "flight"]
result = longest_common_prefix(strings)
print("Longest common prefix:", result)
strings = ["dog", "racecar", "car"]
result = longest_common_prefix(strings)
print("Longest common prefix:", result)
```

OUTPUT:



Analysis:

If the list is empty → return empty string.

Assume the first word is the prefix.

Compare it with each word in the list.

While the current word does not start with the prefix:

- Remove the last character of prefix (prefix = prefix[:-1])

When done, return the remaining prefix.