

ASSIGNMENT-1.4

2303A51337

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**0.5) + 1):
```

```
        if (num % i) == 0:
```

```
            print(f"{num} is not a prime number")
```

```
            break
```

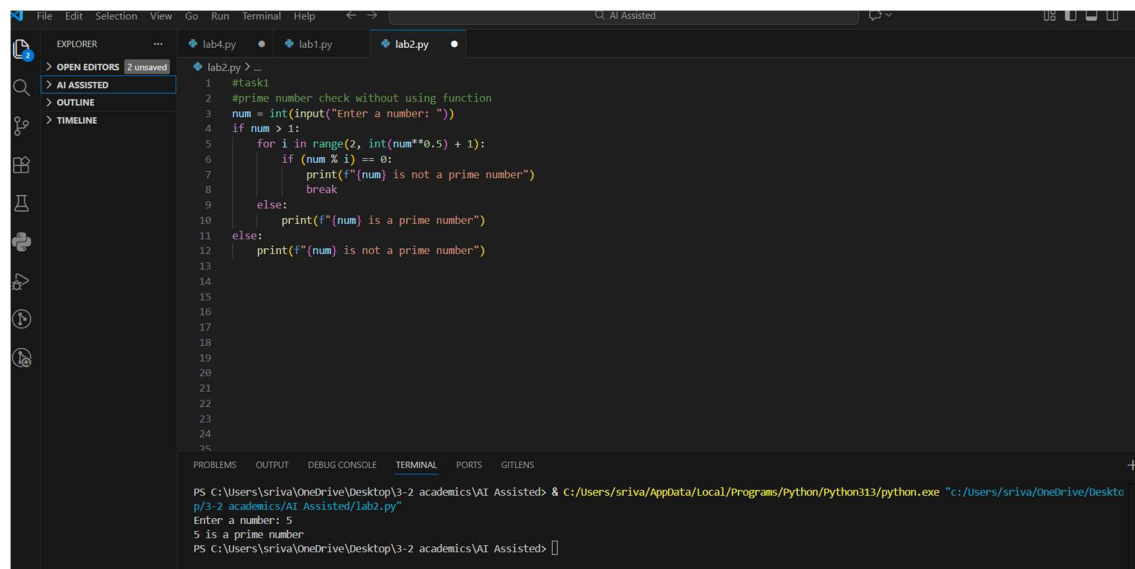
```
    else:
```

```
        print(f"{num} is a prime number")
```

```
else:
```

```
    print(f"{num} is not a prime number")
```

OUTPUT:



The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a file named 'lab2.py'. The main editor area displays the Python code for checking a prime number. The code is as follows:

```
1 #task1
2 #prime number check without using function
3 num = int(input("Enter a number: "))
4 if num > 1:
5     for i in range(2, int(num**0.5) + 1):
6         if (num % i) == 0:
7             print(f"{num} is not a prime number")
8             break
9     else:
10        print(f"{num} is a prime number")
11 else:
12    print(f"{num} is not a prime number")
13
14
15
16
17
18
19
20
21
22
23
24
25
```

The terminal at the bottom shows the command prompt and the execution of the script:

```
PS C:\Users\sriya\OneDrive\Desktop\3-2 academics\AI Assisted> & C:/Users/sriya/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriya/OneDrive/Desktop/p/3-2 academics/AI Assisted/lab2.py"
Enter a number: 5
5 is a prime number
PS C:\Users\sriya\OneDrive\Desktop\3-2 academics\AI Assisted>
```

EXPLANATION:

- This program checks whether a number is prime by testing if it has any divisors from 2 to \sqrt{n} .
- If it is divisible by any number, it is not prime.
- If no divisor is found, it is prime.
- Numbers less than or equal to 1 are not prime..

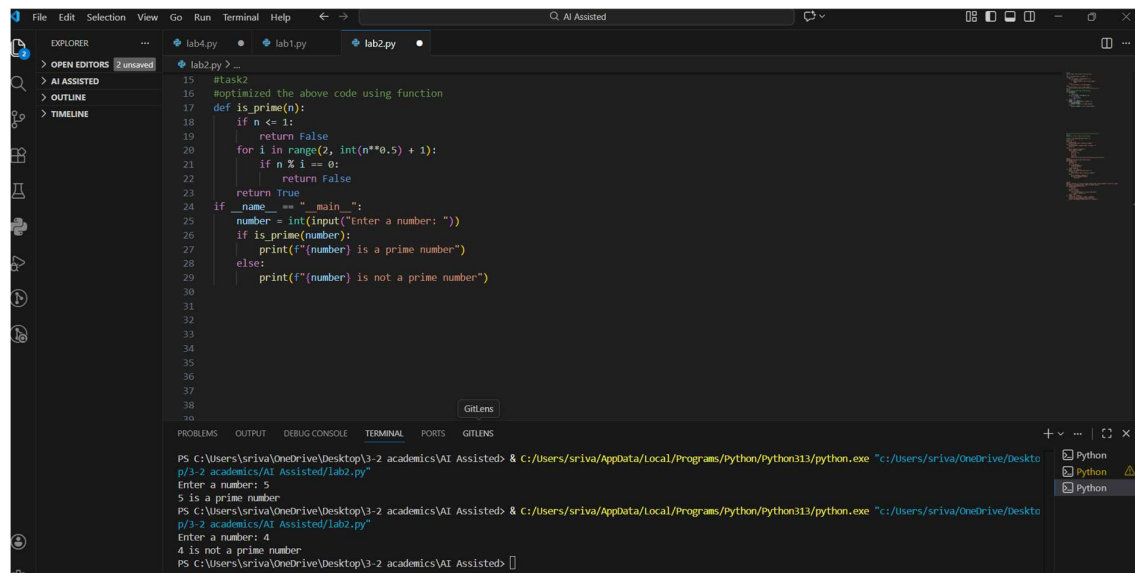
❖ TASK-2**PROMPT:**

optimized the above code using function.

CODE:

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

OUTPUT:



The screenshot shows a VS Code editor with a file named `lab2.py` open. The code defines a function `is_prime(n)` that checks if a number is prime by testing divisibility from 2 to \sqrt{n} . The main part of the program prompts the user for a number and prints whether it is prime or not. The terminal at the bottom shows the execution of the script, with the user entering 5 and 4, and the program outputting '5 is a prime number' and '4 is not a prime number' respectively.

```
15 #task2
16 #optimized the above code using function
17 def is_prime(n):
18     if n <= 1:
19         return False
20     for i in range(2, int(n**0.5) + 1):
21         if n % i == 0:
22             return False
23     return True
24 if __name__ == "__main__":
25     number = int(input("Enter a number: "))
26     if is_prime(number):
27         print(f"{number} is a prime number")
28     else:
29         print(f"{number} is not a prime number")
30
31
32
33
34
35
36
37
38
39
```

Terminal Output:

```
PS C:\Users\sriya\OneDrive\Desktop\3-2 academics\AI Assisted> & c:/Users/sriya/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriya/OneDrive/Desktop/p/3-2 academics/AI Assisted/lab2.py"
Enter a number: 5
5 is a prime number
PS C:\Users\sriya\OneDrive\Desktop\3-2 academics\AI Assisted> & c:/Users/sriya/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriya/OneDrive/Desktop/p/3-2 academics/AI Assisted/lab2.py"
Enter a number: 4
4 is not a prime number
PS C:\Users\sriya\OneDrive\Desktop\3-2 academics\AI Assisted>
```

EXPLANATION:

- This program defines a function `is_prime()` that checks whether a number is prime by trying to divide it by all numbers from 2 up to the square root of the number.
- If any division gives remainder 0, the function returns False (not prime). If no divisor is found, it returns True (prime).
- The main part of the program takes a number from the user, calls the function, and prints whether the number is prime or not.

❖ TASK-3:

PROMPT:

Fibonacci series without using function.

CODE:

```
n_terms = int(input("How many terms? "))
```

```
n1, n2 = 0, 1
```

```
count = 0
```

```
if n_terms <= 0:
```

```
    print("Please enter a positive integer")
```

```
elif n_terms == 1:
```

```
print("Fibonacci sequence upto", n_terms, ":")
```

```
print(n1)
```

```
else:
```

```
print("Fibonacci sequence:")
```

```
while count < n_terms:
```

```
    print(n1)
```

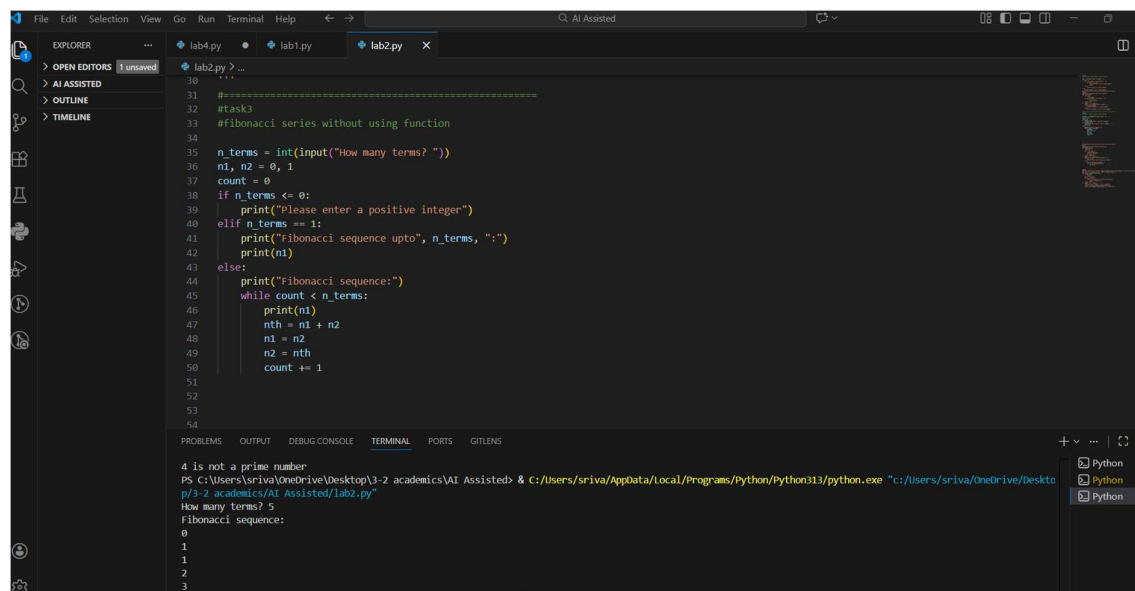
```
    nth = n1 + n2
```

```
    n1 = n2
```

```
    n2 = nth
```

```
    count += 1
```

OUTPUT:



The screenshot shows a Python IDE with a dark theme. The Explorer panel on the left shows three files: lab4.py, lab1.py, and lab2.py. The main editor window displays the code for lab2.py, which is a Python script to generate the Fibonacci sequence. The code includes comments, variable declarations, input handling, and a while loop for generating the sequence. The bottom panel shows the terminal output, which includes an error message about a missing module, followed by the program's execution results for 5 terms.

```
30 '''
31 #task3
32 #fibonacci series without using function
33
34 n_terms = int(input("How many terms? "))
35 n1, n2 = 0, 1
36 count = 0
37
38 if n_terms <= 0:
39     print("Please enter a positive integer")
40 elif n_terms == 1:
41     print("Fibonacci sequence upto", n_terms, ":")
42     print(n1)
43 else:
44     print("Fibonacci sequence:")
45     while count < n_terms:
46         print(n1)
47         nth = n1 + n2
48         n1 = n2
49         n2 = nth
50         count += 1
51
52
53
54
```

4 is not a prime number
PS C:\Users\sriya\OneDrive\Desktop\3-2 academics\AI Assisted> & C:/Users/sriya/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/sriya/OneDrive/Desktop/3-2 academics/AI Assisted/lab2.py"
How many terms? 5
Fibonacci sequence:
0
1
1
2
3

EXPLANATION:

- This program prints the Fibonacci sequence up to the number of terms entered by the user.
It starts with 0 and 1, then each next number is formed by adding the previous two numbers.
- If the user enters 0 or a negative number, it shows an error message.
If the user enters 1, it prints only the first term (0).
Otherwise, it uses a loop to generate and print the required Fibonacci numbers.

❖ TASK-4:

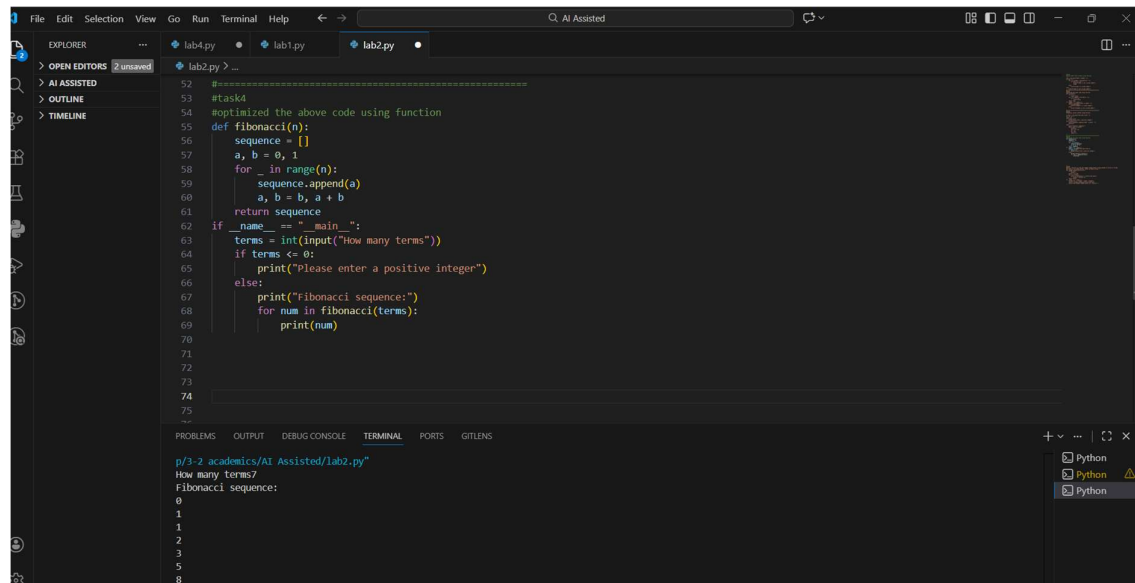
PROMPT:

optimized the above code using function.

CODE:

```
def fibonacci(n):  
    sequence = []  
    a, b = 0, 1  
    for _ in range(n):  
        sequence.append(a)  
        a, b = b, a + b  
    return sequence  
  
if __name__ == "__main__":  
    terms = int(input("How many terms"))  
    if terms <= 0:  
        print("Please enter a positive integer")  
    else:  
        print("Fibonacci sequence:")  
        for num in fibonacci(terms):  
            print(num)
```

OUTPUT:



The screenshot shows a VS Code editor with a Python file named `lab2.py`. The code defines a `fibonacci(n)` function that generates a list of Fibonacci numbers up to the `n`th term. The main part of the program prompts the user for the number of terms. If the input is 0 or negative, it asks for a positive integer. Otherwise, it calls the `fibonacci()` function and prints the sequence.

```
52 #=====
53 #task4
54 #optimized the above code using function
55 def fibonacci(n):
56     sequence = []
57     a, b = 0, 1
58     for _ in range(n):
59         sequence.append(a)
60         a, b = b, a + b
61     return sequence
62 if __name__ == "__main__":
63     terms = int(input("How many terms?"))
64     if terms <= 0:
65         print("Please enter a positive integer")
66     else:
67         print("Fibonacci sequence:")
68         for num in fibonacci(terms):
69             print(num)
70
71
72
73
74
75
```

The terminal output shows the program running and the user entering 7. The output is the Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8.

```
p/3-2 academics/AI Assisted/lab2.py
How many terms?
Fibonacci sequence:
0
1
1
2
3
5
8
```

EXPLANATION:

- The function `fibonacci(n)` creates a list and fills it with Fibonacci numbers starting from 0 and 1. Each new number is formed by adding the previous two numbers. It returns the list of generated numbers.
- In the main part of the program, the user enters how many terms they want. If the number is 0 or negative, the program asks for a positive integer.
- Otherwise, it calls the `fibonacci()` function and prints the Fibonacci sequence.

❖ 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

if __name__ == "__main__":

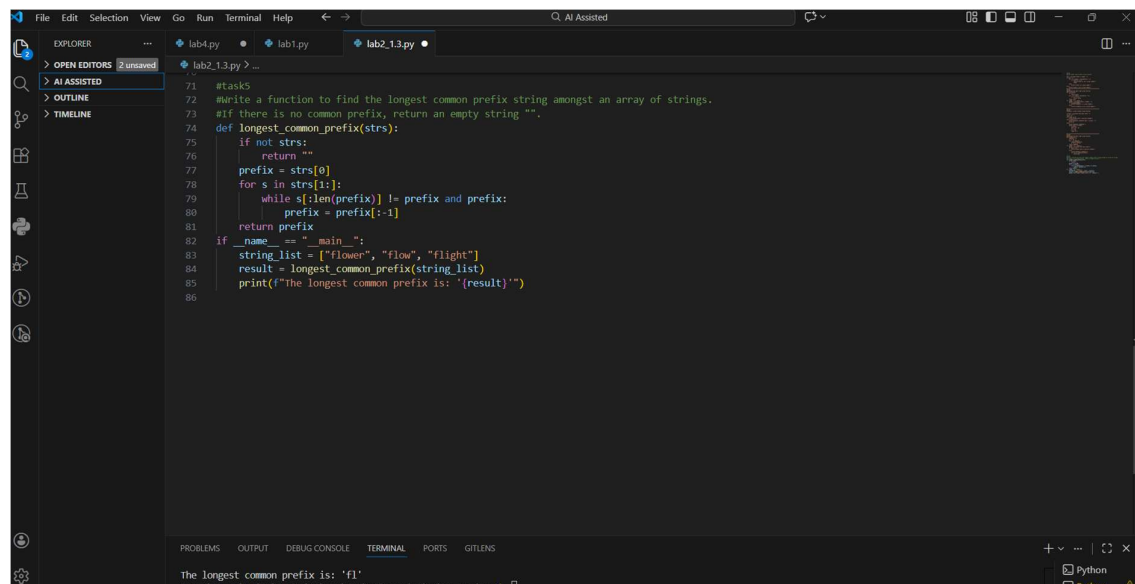
    string_list = ["flower", "flow", "flight"]

    result = longest_common_prefix(string_list)

    print(f"The longest common prefix is: '{result}'")

```

OUTPUT:



The screenshot shows a Visual Studio Code editor with a Python file named `lab2_13.py`. The code defines a function `longest_common_prefix` and a main block. The terminal at the bottom shows the output of the program.

```

71 #task5
72 write a function to find the longest common prefix amongst an array of strings.
73 #if there is no common prefix, return an empty string "".
74 def longest_common_prefix(strs):
75     if not strs:
76         return ""
77     prefix = strs[0]
78     for s in strs[1:]:
79         while s[:len(prefix)] != prefix and prefix:
80             prefix = prefix[:-1]
81     return prefix
82 if __name__ == "__main__":
83     string_list = ["flower", "flow", "flight"]
84     result = longest_common_prefix(string_list)
85     print(f"The longest common prefix is: '{result}'")
86

```

The terminal output shows:

```

The longest common prefix is: 'fl'

```

EXPLANATION:

- This program finds the longest common prefix (starting part) shared by all strings in a list.
- It begins by assuming the first string is the prefix. Then it compares this prefix with each remaining string. If the current prefix does not match the beginning of a string, it shortens the prefix by removing the last character and checks again.
- This continues until a match is found or the prefix becomes empty.
- After checking all strings, the remaining prefix is the longest common prefix, and it is printed as the result.