

# 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:

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows files in the workspace, including AI ASSISTED, assignment-1.py, Assignment-3.3.py, lab-1, LAB-1.4.py, Lab-3.4.py, Lab-4.3.py, Lab-5.4.py, lab-6.3.py, and webapp.log.
- EDITOR:** Displays the Python code for prime number checking.
- TERMINAL:** Shows the command-line output of running the script. It prompts for a number (5), prints "5 is a prime number", and ends with the prompt PS C:\Users\srini\OneDrive\Desktop\AI Assisted>.
- STATUS BAR:** Shows the path PS C:\Users\srini\OneDrive\Desktop\AI Assisted> and icons for powershell and Python Deb...

## **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  $\leq 1$ , 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:**

The screenshot shows the Visual Studio Code interface. In the center is a code editor window displaying a Python script named `LAB-1.4.py`. The code defines a function `is_prime` to check if a number is prime. It includes a loop from 2 to  $\sqrt{\text{num}}$  to find divisors. The main program takes user input, calls the function, and prints the result. Below the code editor is a terminal window showing the execution of the script and its output.

```

13
14 #TASK-2
15 #optimized the above code using function.
16 def is_prime(num):
17     if num <= 1:
18         return False
19     for i in range(2, int(num**0.5) + 1):
20         if (num % i) == 0:
21             return False
22     return True
23
24 number = int(input("Enter a number: "))
25 if is_prime(number):
26     print(number, "is a prime number")
27 else:
28     print(number, "is not a prime number")
29
30
PS C:\Users\srini\OneDrive\Desktop\AI Assisted> & 'c:\users\srini\appdata\local\programs\python\python313\python.exe' 'c:\users\srini\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52847' '--' 'c:\users\srini\onedrive\desktop\ai assisted\lab-1.4.py'
Enter a number: 10
10 is not a prime number
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>

```

## Analysis:

A function `is_prime(num)` is defined.

If  $\text{number} \leq 1 \rightarrow$  returns False.

Loop runs from 2 to  $\sqrt{\text{num}}$ .

If any divisor is found  $\rightarrow$  returns False.

If no divisor is found  $\rightarrow$  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:

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows files in the AI ASSISTED folder: Welcome, lab-1, Assignment-3.3.py, assignment-1.py, Lab-4.3.py, Lab-3.4.py, LAB-1.4.py (selected), Lab-5.4.py, Lab-6.3.py, and webapp.log.
- CODE EDITOR:** Displays the Python code for generating a Fibonacci series. The code includes error handling for non-positive integers and prints the series up to n terms.
- TERMINAL:** Shows the command-line output of the program running in a PowerShell terminal. It prompts for the number of terms (n=10) and then prints the Fibonacci series from 0 to 34.
- STATUS BAR:** Shows the path C:\Users\srini\OneDrive\Desktop\AI Assisted and the file LAB-1.4.py.

## Analysis:

User enters number of terms n.

If  $n \leq 0$  → 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:**

The screenshot shows the VS Code interface with the 'AI ASSISTED' feature active. The left sidebar shows files like 'assignment-1.py', 'Assignment-3.3.py', 'lab-1', 'LAB-1.4.py', 'Lab-3.4.py', 'Lab-4.3.py', 'Lab-5.py', and 'webapp.log'. The main editor window displays a Python script for generating a Fibonacci series. The code is as follows:

```

46     #TASK-4
47     #optimized the above code using function.
48
49     def fibonacci_series(n):
50         a, b = 0, 1
51         count = 0
52         if n <= 0:
53             print("Please enter a positive integer.")
54         elif n == 1:
55             print("Fibonacci series up to", n, ":")
56             print(a)
57         else:
58             print("Fibonacci series up to", n, ":")
59             while count < n:
60                 print(a, end=" ")
61                 a, b = b, a + b
62                 count += 1
63
64     number_of_terms = int(input("Enter the number of terms in Fibonacci series: "))
65     fibonacci_series(number_of_terms)

```

The bottom right corner of the interface shows a status bar with 'powershell' and 'Python Deb...'.

## 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:
```

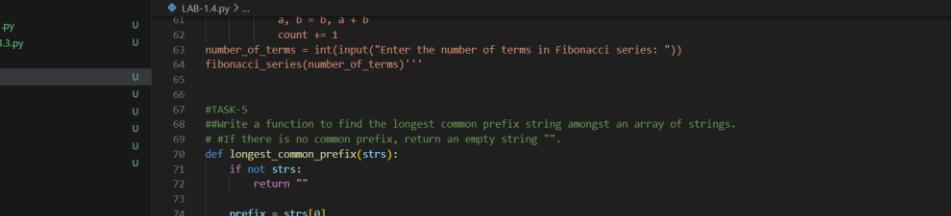
```
            prefix = prefix[:-1]
```

```
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:**



The screenshot shows the Visual Studio Code interface with the 'AI Assisted' extension active. The left sidebar displays a tree view of files under 'EXPLORER' and 'AI ASSISTED'. The main editor area shows a Python script named 'LAB-1.4.py' with code for generating a Fibonacci series. The status bar at the bottom indicates the file path as 'PS C:\Users\sriini\Desktop\AI Assisted & c:\Users\sriini\AppData\Local\Programs\Python\Python313\python.exe' and the command 'powershell'. The bottom right corner features the Python logo.

```
LAB-1.4.py ...
61     a, b = b, a + b
62     count += 1
63
64     number_of_terms = int(input("Enter the number of terms in Fibonacci series: "))
65     fibonacci_series(number_of_terms)
66
67 #TASK-5
68
69     #Write a function to find the longest common prefix string amongst an array of strings.
70     # If there is no common prefix, return an empty string "".
71     def longest_common_prefix(strs):
72         if not strs:
73             return ""
74
75         prefix = strs[0]
76         for s in strs[1:]:
77             while s[:len(prefix)] != prefix and prefix:
78                 prefix = prefix[:-1]
79
80         return prefix
81
82     strings = ["flower", "flow", "flight"]
83     result = longest_common_prefix(strings)
84     print("longest common prefix:", result)
85
86     strings = ["dog", "racecar", "car"]
87     result = longest_common_prefix(strings)
88     print("longest common prefix:", result)
```

## **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.