

# ASSIGNMENT-1.3

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

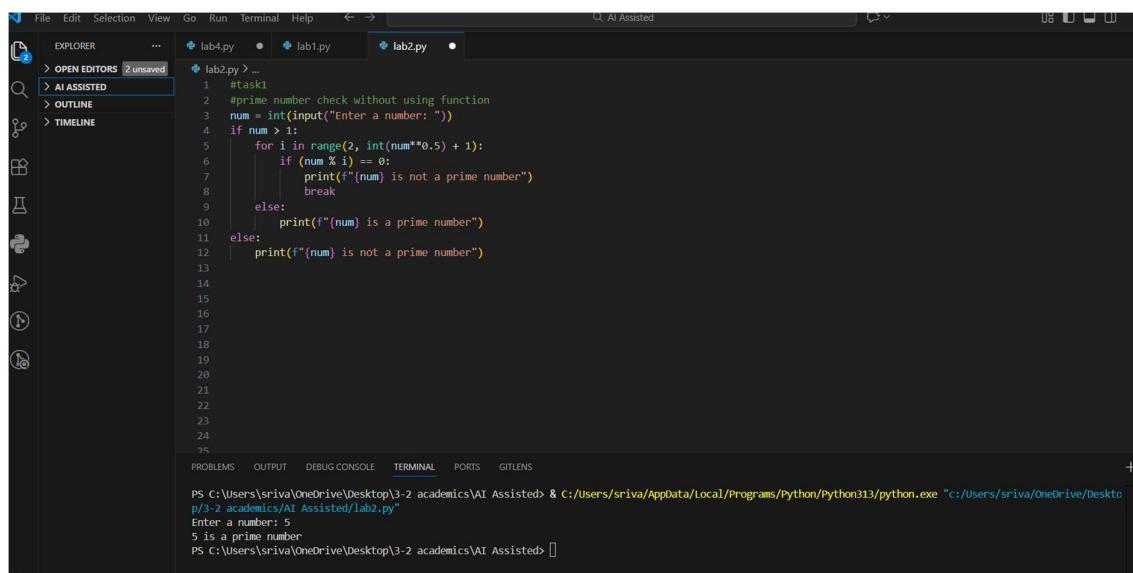
    else:

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

else:

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

### OUTPUT:



The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for Open Editors, AI Assisted, Outline, Timeline, and others.
- Explorer:** Shows files lab2.py, lab1.py, and lab1.py (unpublished).
- Editor Area:** Displays the Python code for prime number checking.
- Terminal:** Shows the command PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted> & C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriva/OneDrive/Desktop\p/3-2 academics\AI Assisted\lab2.py". The terminal also displays the user's input "Enter a number: 5" and the output "5 is a prime number".

**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:

```
File Edit Selection View Go Run Terminal Help ⏎ → Q AI Assisted
EXPLORER ... lab4.py ● lab1.py lab2.py ●
OPEN EDITORS 2 unsaved
AI ASSISTED
OUTLINE
TIMELINE
GitLens
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted> & C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriva/OneDrive/Desktop\3-2 academics\AI Assisted\lab2.py"
Enter a number: 5
5 is a prime number
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted> & C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriva/OneDrive/Desktop\3-2 academics\AI Assisted\lab2.py"
Enter a number: 4
4 is not a prime number
PS C:\Users\sriva\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 the Visual Studio Code interface with the following details:

- File Explorer:** Shows files lab4.py, lab1.py, and lab2.py.
- Terminal:** Displays the Python code for generating a Fibonacci sequence up to 5 terms. The code includes error handling for non-positive integers and prints the sequence starting from 0.
- Output:** Shows the terminal output where the user inputs "5" and the program prints the Fibonacci sequence: 0, 1, 1, 2, 3.
- Python Interpreter:** A sidebar shows the Python interpreter path: Python > Python > Python.

### 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 code editor interface with the following details:

- File Explorer:** Shows files lab4.py, lab1.py, and lab2.py.
- Terminal:** Displays the command `p/3-2 academics/AI Assisted/lab2.py*` and the output of the program execution.
- Output:** Shows the Fibonacci sequence starting from 0 up to 8.
- Python Interpreter:** A sidebar on the right shows three active Python sessions.

```
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
```

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:

```

File Edit Selection View Go Run Terminal Help ⏎ ↻ 🔍 AI Assisted
EXPLORER ... lab4.py lab1.py lab2_13.py
AI ASSISTED
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
The longest common prefix is: 'fl'
ps C:\Users\asus\Desktop\lab2_13_2_academic\AI Assisted
Python

```

## 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.