Assignment 1

2303A510A1 - Bhanu

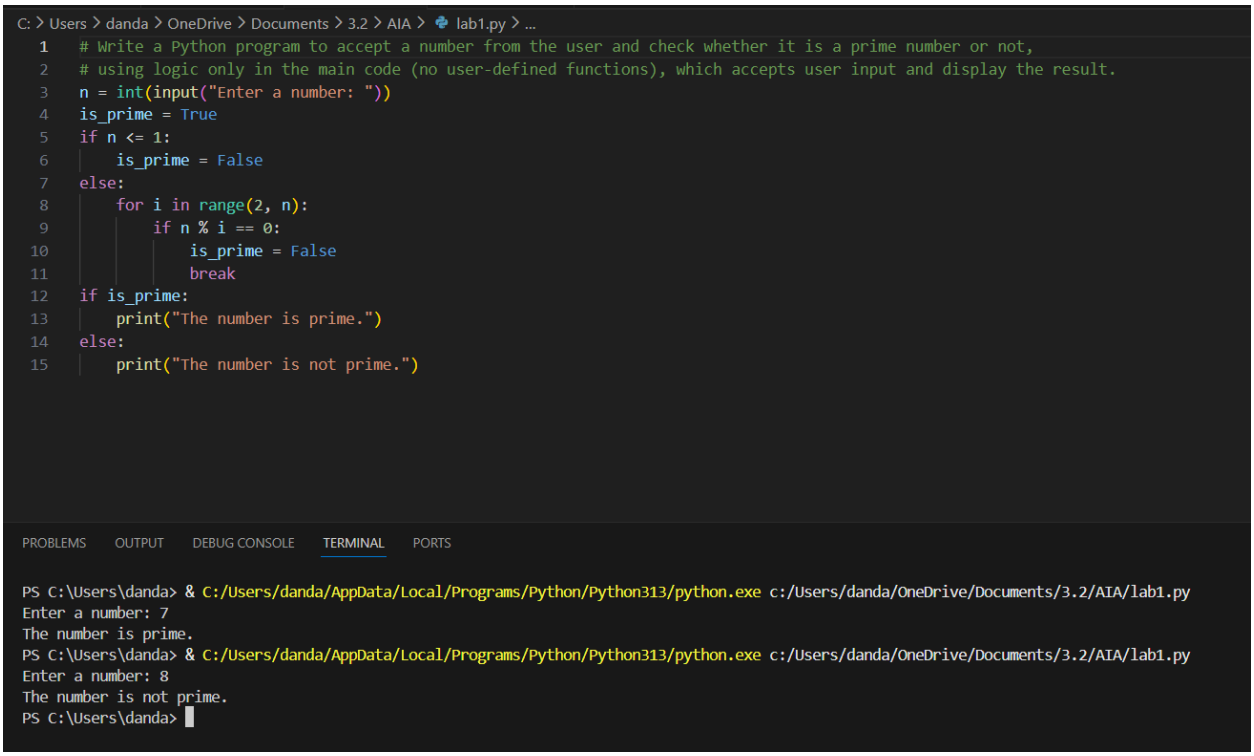
Batch - 14

Task 1: AI-Generated Logic Without Modularization (Prime Number Check Without Functions)

Prompt:

Write a Python program to accept a number from the user and check whether it is a prime number or not, using logic only in the main code (no user-defined functions), which accepts user input and display the result.

Code & Output:



C: > Users > danda > OneDrive > Documents > 3.2 > AIA > lab1.py > ...

```python
# Write a Python program to accept a number from the user and check whether it is a prime number or not,
# using logic only in the main code (no user-defined functions), which accepts user input and display the result.
n = int(input("Enter a number: "))
is_prime = True
if n <= 1:
    is_prime = False
else:
    for i in range(2, n):
        if n % i == 0:
            is_prime = False
            break
if is_prime:
    print("The number is prime.")
else:
    print("The number is not prime.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\danda> & C:/Users/danda/AppData/Local/Programs/Python/Python313/python.exe c:/Users/danda/OneDrive/Documents/3.2/AIA/lab1.py
Enter a number: 7
The number is prime.
PS C:\Users\danda> & C:/Users/danda/AppData/Local/Programs/Python/Python313/python.exe c:/Users/danda/OneDrive/Documents/3.2/AIA/lab1.py
Enter a number: 8
The number is not prime.
PS C:\Users\danda>
```

Explanation:

- The program takes a number as input from the user.

- It checks whether the number can be divided by any number other than 1 and itself.
- If it is divisible, the number is not a prime number.
- If it is not divisible by any number, it is a prime number.

Justification:

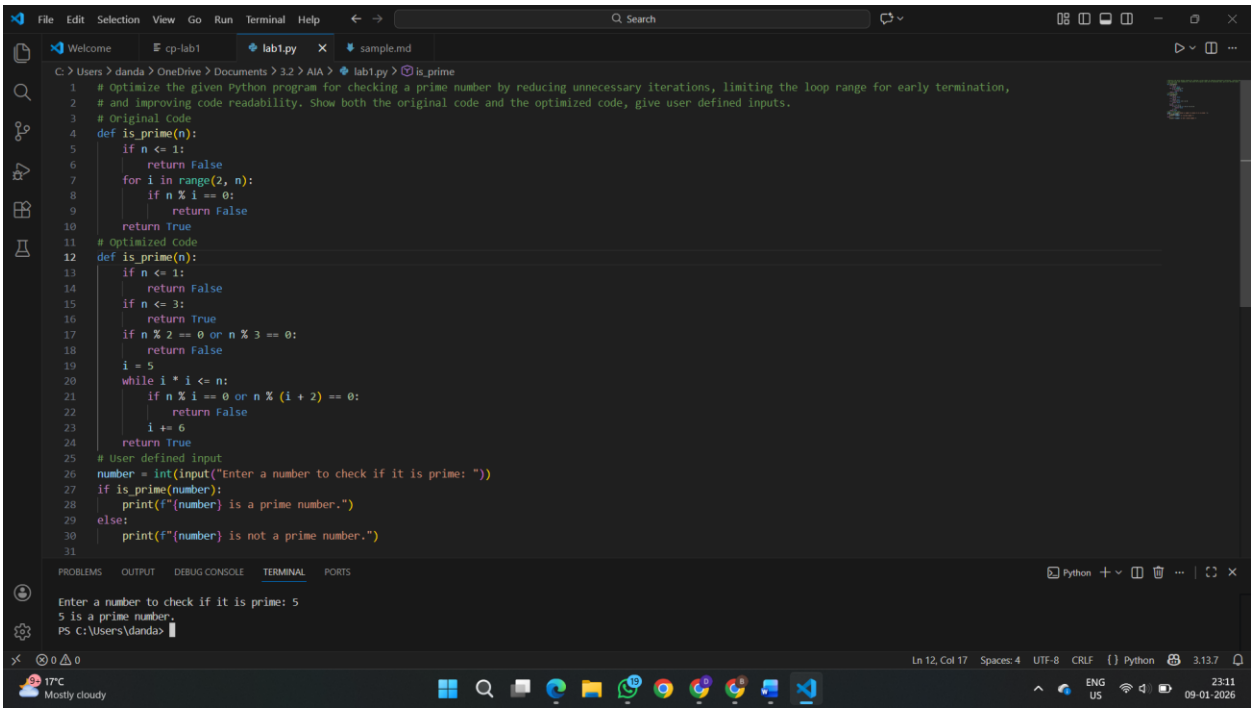
This program checks whether a given number is prime using direct conditional logic without defining any functions. All computations are performed sequentially in a single block, making the logic easy to follow and suitable for beginners.

Task 2: Efficiency & Logic Optimization (Cleanup)

Prompt:

Optimize the given Python program for checking a prime number by reducing unnecessary iterations, limiting the loop range for early termination, and improving code readability. Show both the original code and the optimized code, and explain how the changes improve efficiency and reduce time complexity.

Code & Output:



Justification:

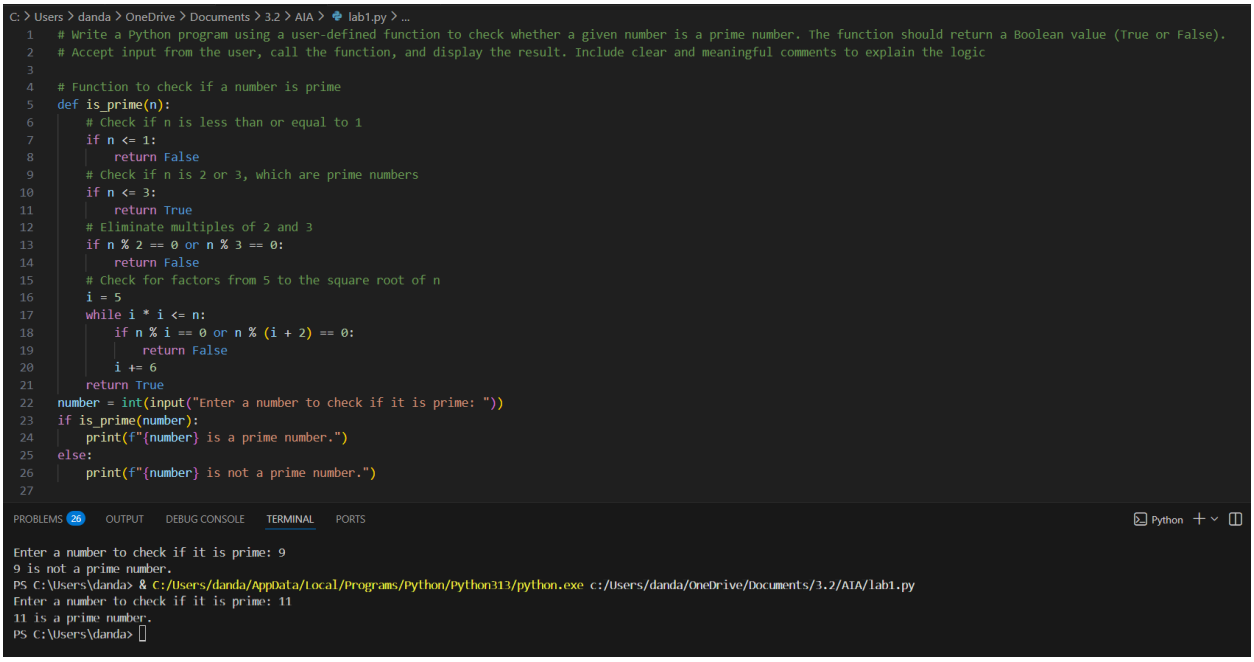
The optimized code reduces unnecessary iterations by checking divisibility only up to the square root of the number instead of all numbers up to n. Early termination using break stops the loop as soon as a factor is found, saving time. Using a boolean variable improves code readability and clarity.

Task 3: Modular Design Using AI Assistance (Prime Number Check Using Functions)

Prompt:

Write a Python program using a user-defined function to check whether a given number is a prime number. The function should return a Boolean value (True or False). Accept input from the user, call the function, and display the result. Include clear and meaningful comments to explain the logic

Code & Output:



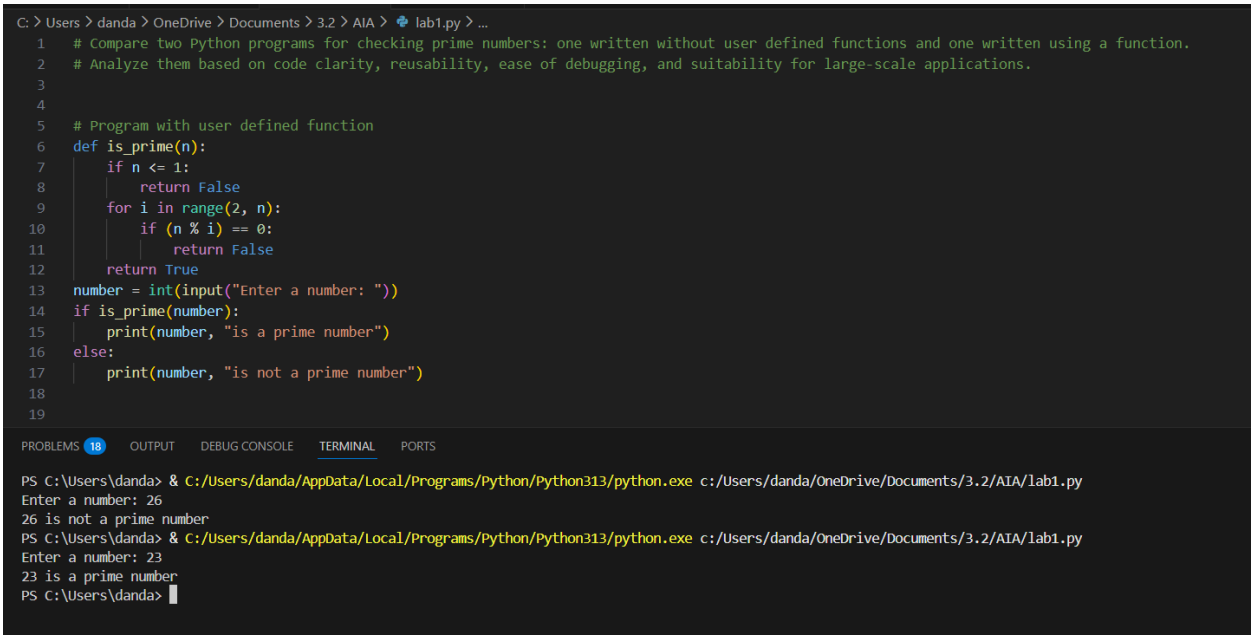
```
C: > Users > danda > OneDrive > Documents > 3.2 > AIA > lab1.py > ...
# Write a Python program using a user-defined function to check whether a given number is a prime number. The function should return a Boolean value (True or False).
# Accept input from the user, call the function, and display the result. Include clear and meaningful comments to explain the logic

# Function to check if a number is prime
def is_prime(n):
    # Check if n is less than or equal to 1
    if n <= 1:
        return False
    # Check if n is 2 or 3, which are prime numbers
    if n <= 3:
        return True
    # Eliminate multiples of 2 and 3
    if n % 2 == 0 or n % 3 == 0:
        return False
    # Check for factors from 5 to the square root of n
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True
number = int(input("Enter a number to check if it is prime: "))
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")

PROBLEMS 26  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                Python

Enter a number to check if it is prime: 9
9 is not a prime number.
PS C:\Users\danda> & C:/Users/danda/AppData/Local/Programs/Python/Python313/python.exe c:/Users/danda/OneDrive/Documents/3.2/AIA/lab1.py
Enter a number to check if it is prime: 11
11 is a prime number.
PS C:\Users\danda>
```

Task 4: Comparative Analysis –With vs Without Functions

Prompt:

Compare two Python programs for checking prime numbers: one written without user defined functions and one written using a function. Analyze them based on code clarity, reusability, ease of debugging, and suitability for large-scale applications.

Code & Output:



```
C: > Users > danda > OneDrive > Documents > 3.2 > AIA > lab1.py > ...
1   # Compare two Python programs for checking prime numbers: one written without user defined functions and one written using a function.
2   # Analyze them based on code clarity, reusability, ease of debugging, and suitability for large-scale applications.
3
4
5   # Program with user defined function
6   def is_prime(n):
7       if n <= 1:
8           return False
9       for i in range(2, n):
10          if (n % i) == 0:
11              return False
12      return True
13  number = int(input("Enter a number: "))
14  if is_prime(number):
15      print(number, "is a prime number")
16  else:
17      print(number, "is not a prime number")
18
19

PROBLEMS 18   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\danda> & C:/Users/danda/AppData/Local/Programs/Python/Python313/python.exe c:/Users/danda/OneDrive/Documents/3.2/AIA/lab1.py
Enter a number: 26
26 is not a prime number
PS C:\Users\danda> & C:/Users/danda/AppData/Local/Programs/Python/Python313/python.exe c:/Users/danda/OneDrive/Documents/3.2/AIA/lab1.py
Enter a number: 23
23 is a prime number
PS C:\Users\danda>
```

Justification:

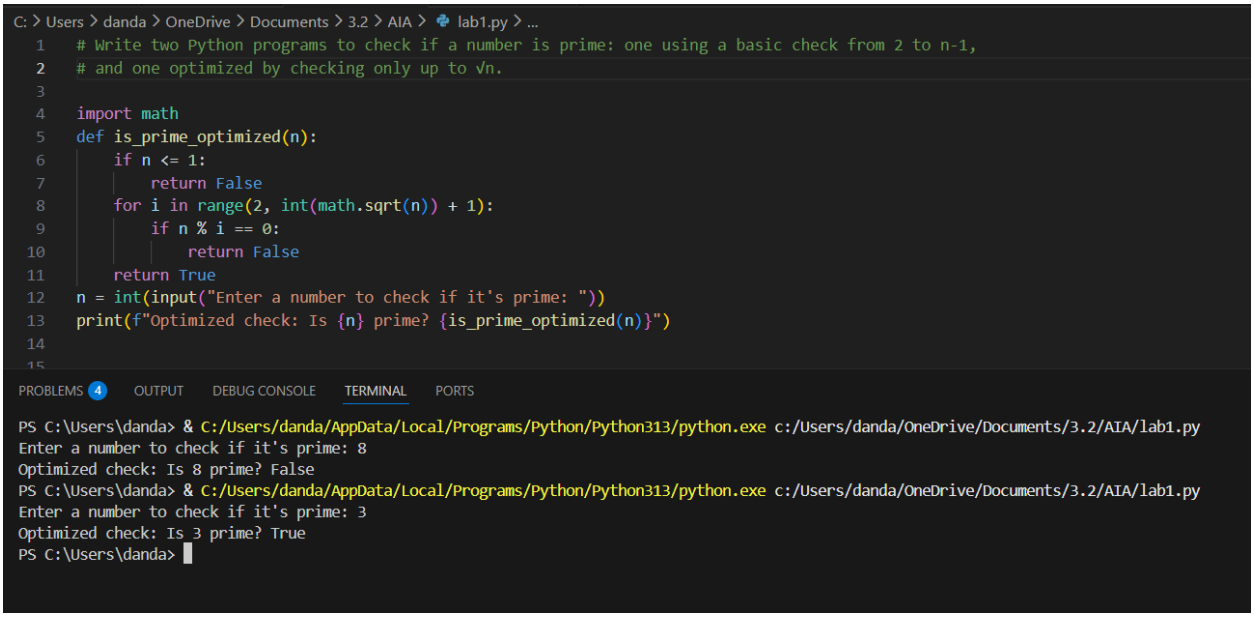
The first program without functions works correctly but has all the logic inline, making it less clear and harder to reuse or debug. The second program uses a user-defined function, which separates the prime-checking logic from the main program, improving readability and maintainability. Using a function allows reusability, so the same logic can be called multiple times without rewriting code.

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to Prime Checking)

Prompt:

Write two Python programs to check if a number is prime: one using a basic check from 2 to n-1, and one optimized by checking only up to √n.

Code & Output:



Justification:

The first program uses a basic divisibility check from 2 to n-1. While it is simple and easy to understand, it performs many unnecessary iterations, making it inefficient for large numbers. The second program uses an optimized approach, checking divisibility only up to √n. This reduces the number of iterations significantly, improving efficiency while giving the same correct result. The optimized version is faster, easier to scale, and more suitable for larger input values compared to the basic approach.