ASSIGNMENT-1

R.Nikhil Kumar

2303A52260

Batch-44

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

Scenario : You are developing a basic validation script for a numerical learning

application

Prompt : generate prime numbers without using any function definitions

Code :

```
# Prime Number Check Without Functions
number = int(input("Enter a number: "))
if number > 1:
    for i in range(2, int(number**0.5) + 1):
        if (number % i) == 0:
            print(f"{number} is not a prime number")
            break
    else:
        print(f"{number} is a prime number")
else:
    print(f"{number} is not a prime number")
# End of code
```

```
Enter a number: 4
4 is not a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe
/OneDrive/Desktop/AI Assisted coding/lab.py"
Enter a number: 5
5 is a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding>
```

Report: This code checks whether a given number is a prime number or not without using functions.

Task-2 :

Efficiency & Logic Optimization (Cleanup)

Scenario: The script must handle larger input values efficiently.

```python
# Task 2
# Efficiency & Logic Optimization (Cleanup)
# Scenario : The script must handle larger input values efficiently
number = int(input("Enter a number: "))
if number > 1:
    is_prime = True
    for i in range(2, int(number**0.5) + 1):
        if (number % i) == 0:
            is_prime = False
            break
    if is_prime:
        print(f"{number} is a prime number")
    else:
        print(f"{number} is not a prime number")
else:
    print(f"{number} is not a prime number")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                Python

PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DEL
/OneDrive/Desktop/AI Assisted coding/lab.py"
Enter a number: 40
40 is not a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DEL
/OneDrive/Desktop/AI Assisted coding/lab.py"
Enter a number: 43
43 is a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding>
```

Report: In this task, I optimized the prime number checking algorithm to improve its efficiency for larger input values.

Task-3:

Modular Design Using AI Assistance (Prime Number Check Using Functions)

Scenario: The prime-checking logic will be reused across multiple modules

Code :

```python
lab.py > ...
#  Task 3
# Modular Design Using AI Assistance (Prime Number Check Using Functions
# Scenario : The prime-checking logic will be reused across multiple modules
def is_prime(number):
    if number <= 1:
        return False
    for i in range(2, int(number**0.5) + 1):
        if (number % i) == 0:
            return False
    return True
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")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                Python

PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desktop/AI Assisted co
ding/lab.py"
Enter a number: 88
88 is not a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desktop/AI Assisted co
ding/lab.py"
Enter a number: 83
83 is a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding>
```

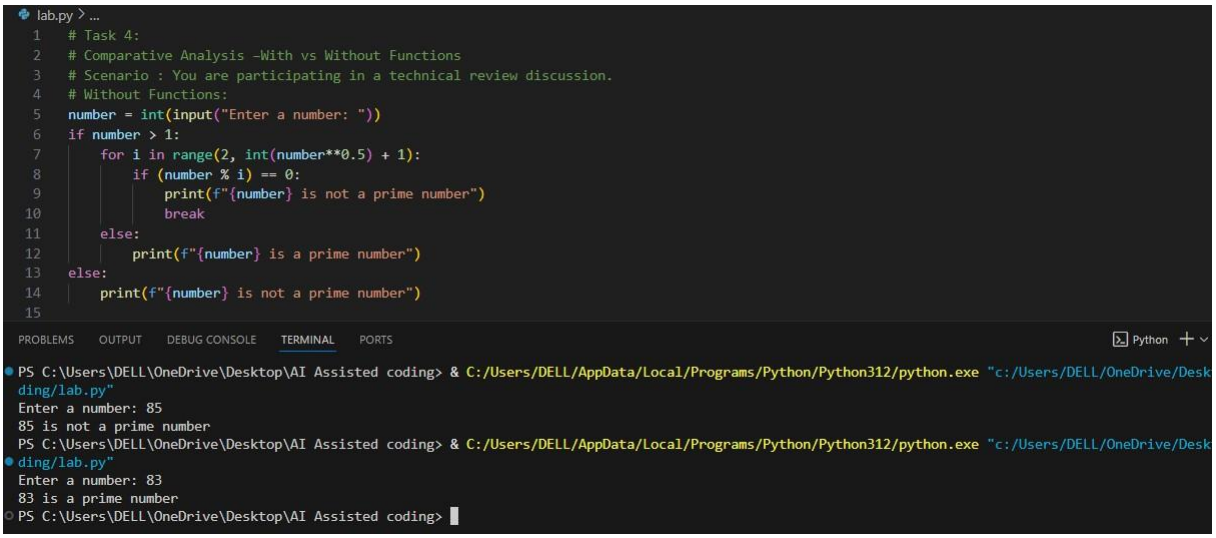
Report : The code defines a function `is_prime` that checks if a given number is prime. It first checks if the number is less than or equal to 1, in which case it returns False.

Task 4:

Comparative Analysis –With vs Without Functions

Scenario : You are participating in a technical review discussion

#Without Function Code

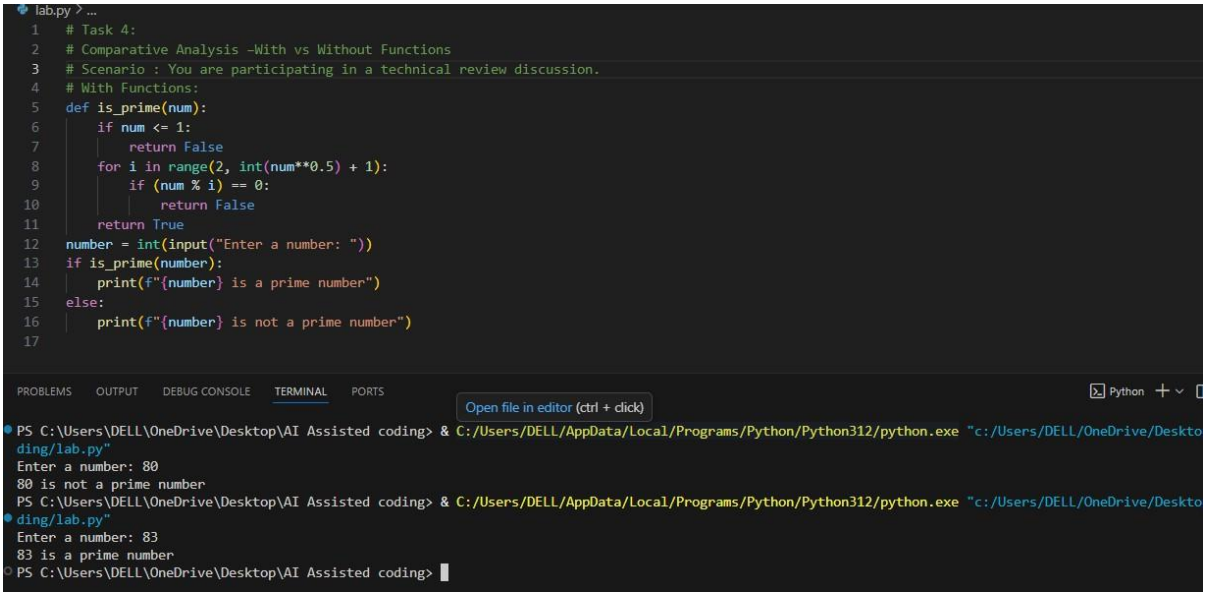


```
lab.py > ...
1  # Task 4:
2  # Comparative Analysis -With vs Without Functions
3  # Scenario : You are participating in a technical review discussion.
4  # Without Functions:
5  number = int(input("Enter a number: "))
6  if number > 1:
7      for i in range(2, int(number**0.5) + 1):
8          if (number % i) == 0:
9              print(f"{number} is not a prime number")
10             break
11     else:
12         print(f"{number} is a prime number")
13 else:
14     print(f"{number} is not a prime number")
15
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   Python

PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desk
ding/lab.py"
Enter a number: 85
85 is not a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desk
ding/lab.py"
Enter a number: 83
83 is a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding>
```

With Function Code



```
lab.py > ...
1  # Task 4:
2  # Comparative Analysis -With vs Without Functions
3  # Scenario : You are participating in a technical review discussion.
4  # With Functions:
5  def is_prime(num):
6      if num <= 1:
7          return False
8      for i in range(2, int(num**0.5) + 1):
9          if (num % i) == 0:
10             return False
11     return True
12 number = int(input("Enter a number: "))
13 if is_prime(number):
14     print(f"{number} is a prime number")
15 else:
16     print(f"{number} is not a prime number")
17
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   Python
                                  Open file in editor (ctrl + click)

PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Deskto
ding/lab.py"
Enter a number: 80
80 is not a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Deskto
ding/lab.py"
Enter a number: 83
83 is a prime number
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding>
```

Report : In the first approach without functions, the code is straightforward but lacks modularity. It is harder to reuse the prime-checking logic in other parts of the

program or in different programs. In the second approach with functions, the prime-checking logic is encapsulated within the is_prime function. This makes the code more modular, reusable, and easier to read.

Task-5:

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

Scenario : Your mentor wants to evaluate how AI handles alternative logical

Strategies

Code :

```python
# Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to Prime Checking)
# Your mentor wants to evaluate how AI handles alternative logical strategies.
def iterative_fibonacci(n):
    """Calculate Fibonacci number using an iterative approach."""
    if n <= 0:
        return 0
    elif n == 1:
        return 1

    a, b = 0, 1
    for _ in range(2, n + 1):
        a, b = b, a + b
    return b
def recursive_fibonacci(n):
    """Calculate Fibonacci number using a recursive approach."""
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return recursive_fibonacci(n - 1) + recursive_fibonacci(n - 2)
# Example usage:
n = int(input("Enter a positive integer to compute its Fibonacci number: "))
print("Iterative Fibonacci:", iterative_fibonacci(n))
print("Recursive Fibonacci:", recursive_fibonacci(n))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                Python

PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desktop/
ding/lab.py"
Enter a positive integer to compute its Fibonacci number: 10
Iterative Fibonacci: 55
Recursive Fibonacci: 55
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted coding> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desktop/
ding/lab.py"
Enter a positive integer to compute its Fibonacci number: 8
Iterative Fibonacci: 21
Recursive Fibonacci: 21
```

Report : In this code, we implement two different approaches to calculate the Fibonacci number for a given positive integer n: an iterative approach and a recursive approach. The iterative approach uses a loop to compute the Fibonacci number, which is generally more efficient in terms of time and space complexity. It runs in O(n) time and O(1) space. The recursive approach, on the other hand, calls itself to compute the Fibonacci number. While this method is more straightforward and easier to understand, it has an exponential time complexity of O(2^n) due to the repeated calculations of the same Fibonacci numbers, making it inefficient for larger values of n.