AI ASSISTED CODING

Hall Ticket No: 2303A51850

Batch:13

Assignment-1.4

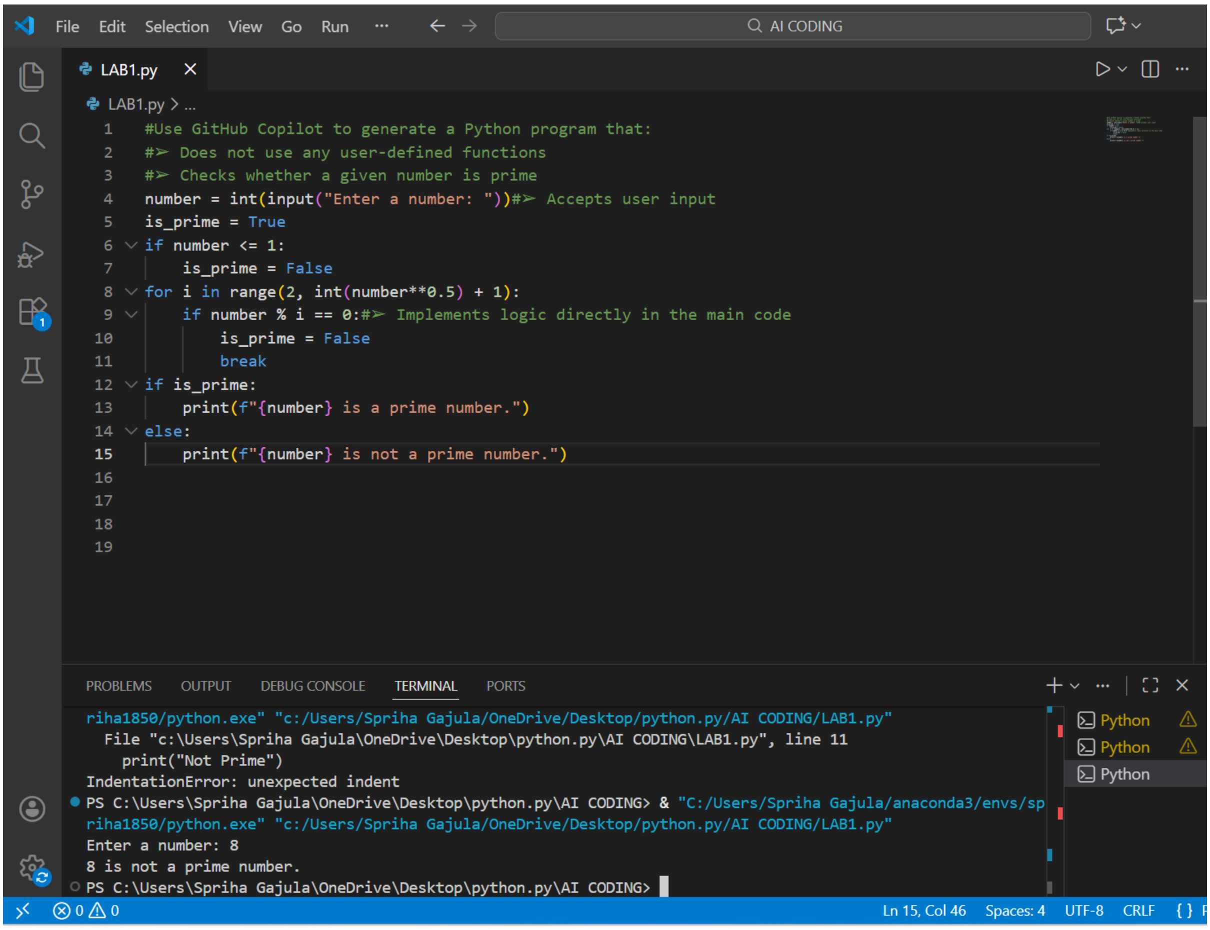
Task-1. AI-Generated Logic Without Modularization (Prime Number Check

Without Functions)

Prompt

#Generate a proper python code to check weather the given number is prime or not without using any functions

Code & Output



File Edit Selection View Go Run ···
AI CODING
LAB1.py
LAB1.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Python
Python
Ln 15, Col 46 Spaces: 4 UTF-8 CRLF { }

```python
#Use GitHub Copilot to generate a Python program that:
#➢ Does not use any user-defined functions
#➢ Checks whether a given number is prime
number = int(input("Enter a number: "))#➢ Accepts user input
is_prime = True
if number <= 1:
    is_prime = False
for i in range(2, int(number**0.5) + 1):
    if number % i == 0:#➢ Implements logic directly in the main code
        is_prime = False
        break
if is_prime:
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

```
riha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/AI CODING/LAB1.py"
  File "c:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING\LAB1.py", line 11
    print("Not Prime")
IndentationError: unexpected indent
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/sp
riha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/AI CODING/LAB1.py"
Enter a number: 8
8 is not a prime number.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING>
```

Justification:

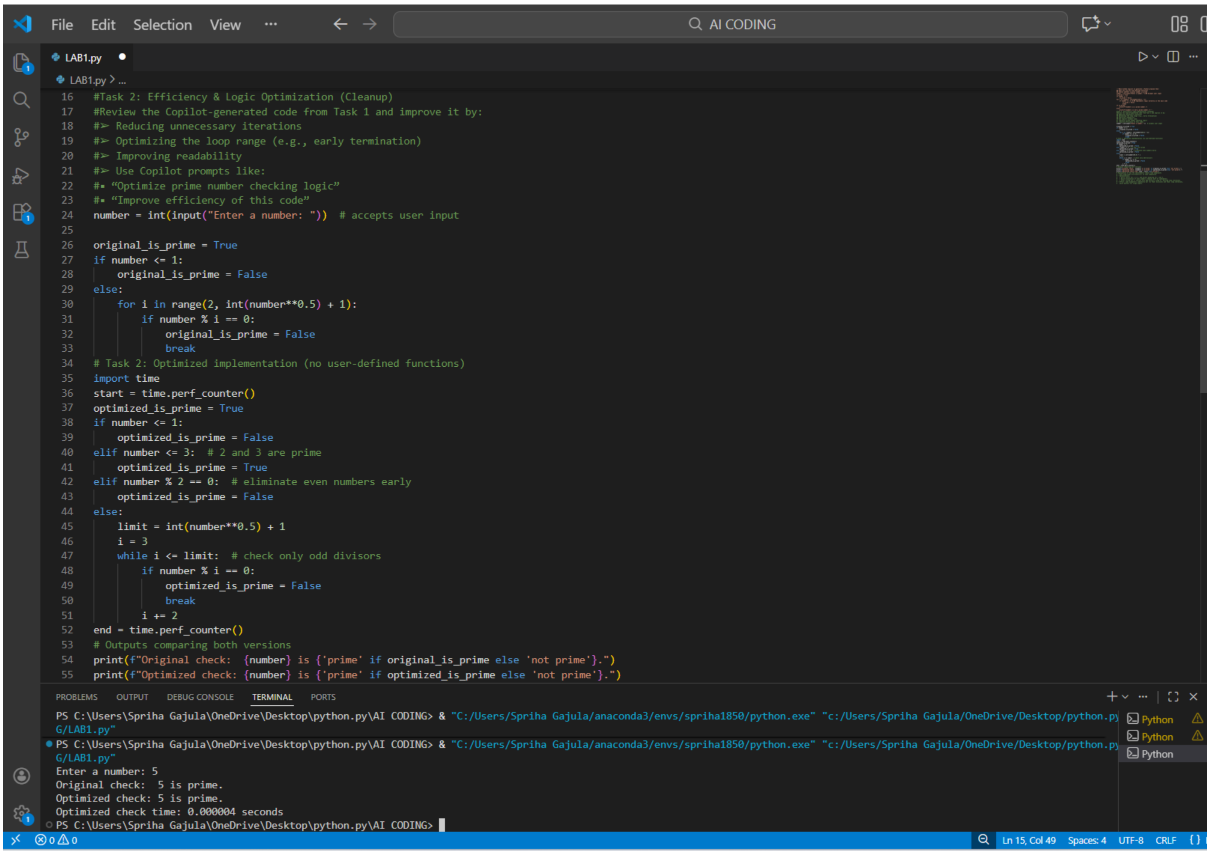
This program checks whether a given number is prime using direct conditional logic without defining any functions.
All computations are performed sequentially in a single block, making the logic easy to follow and suitable for beginners.

Task-2. Efficiency & Logic Optimization (Cleanup)

Prompt

#Improve readability while keeping the logic simple and improve efficiency of the code by reducing iterations also minimize the code length

Code & Ouput :



```python
#Task 2: Efficiency & Logic Optimization (Cleanup)
#Review the Copilot-generated code from Task 1 and improve it by:
#➢ Reducing unnecessary iterations
#➢ Optimizing the loop range (e.g., early termination)
#➢ Improving readability
#➢ Use Copilot prompts like:
#▪ “Optimize prime number checking logic”
#▪ “Improve efficiency of this code”
number = int(input("Enter a number: "))  # accepts user input

original_is_prime = True
if number <= 1:
    original_is_prime = False
else:
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            original_is_prime = False
            break
# Task 2: Optimized implementation (no user-defined functions)
import time
start = time.perf_counter()
optimized_is_prime = True
if number <= 1:
    optimized_is_prime = False
elif number <= 3:  # 2 and 3 are prime
    optimized_is_prime = True
elif number % 2 == 0:  # eliminate even numbers early
    optimized_is_prime = False
else:
    limit = int(number**0.5) + 1
    i = 3
    while i <= limit:  # check only odd divisors
        if number % i == 0:
            optimized_is_prime = False
            break
        i += 2
end = time.perf_counter()
# Outputs comparing both versions
print(f"Original check:  {number} is {'prime' if original_is_prime else 'not prime'}.")
print(f"Optimized check: {number} is {'prime' if optimized_is_prime else 'not prime'}.")
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py
G/LAB1.py"
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py
G/LAB1.py"
Enter a number: 5
Original check:  5 is prime.
Optimized check: 5 is prime.
Optimized check time: 0.000004 seconds
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING>
```

Justification:

The optimized script improves performance by reducing unnecessary iterations and limiting the loop range, enabling faster execution for large

input values.
Early termination and simplified conditions lower the overall time complexity while maintaining correct prime number validation.

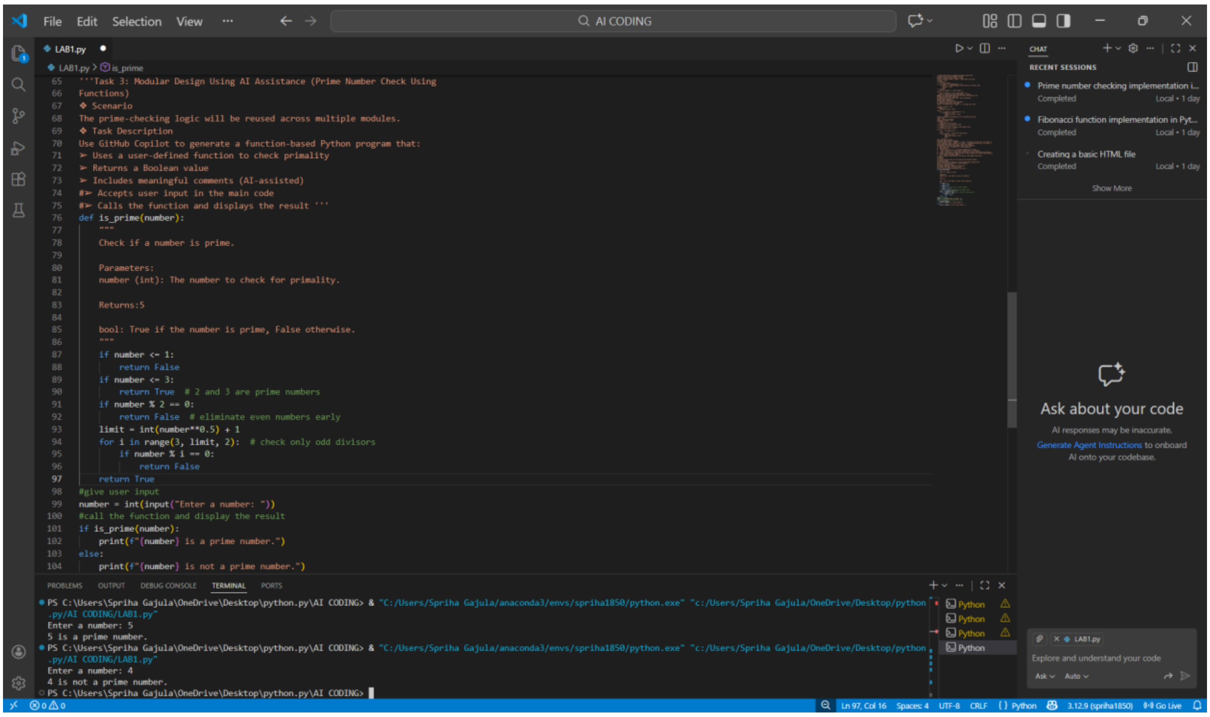
Task-3. Modular Design Using AI Assistance (Prime Number Check Using

Functions)

Prompt:

#The function must return a Boolean value (True if prime, False otherwise)

Code & output :



```python
'''Task 3: Modular Design Using AI Assistance (Prime Number Check Using
Functions)
❖ Scenario
The prime-checking logic will be reused across multiple modules.
❖ Task Description
Use GitHub Copilot to generate a function-based Python program that:
➢ Uses a user-defined function to check primality
➢ Returns a Boolean value
➢ Includes meaningful comments (AI-assisted)
#➢ Accepts user input in the main code
#➢ Calls the function and displays the result '''
def is_prime(number):
    """
    Check if a number is prime.

    Parameters:
    number (int): The number to check for primality.

    Returns:5

    bool: True if the number is prime, False otherwise.
    """
    if number <= 1:
        return False
    if number <= 3:
        return True  # 2 and 3 are prime numbers
    if number % 2 == 0:
        return False  # eliminate even numbers early
    limit = int(number**0.5) + 1
    for i in range(3, limit, 2):  # check only odd divisors
        if number % i == 0:
            return False
    return True
#give user input
number = int(input("Enter a number: "))
#call the function and display the result
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python
.py/AI CODING/LAB1.py"
Enter a number: 5
5 is a prime number.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python
.py/AI CODING/LAB1.py"
Enter a number: 4
4 is not a prime number.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING>
```

Justification:

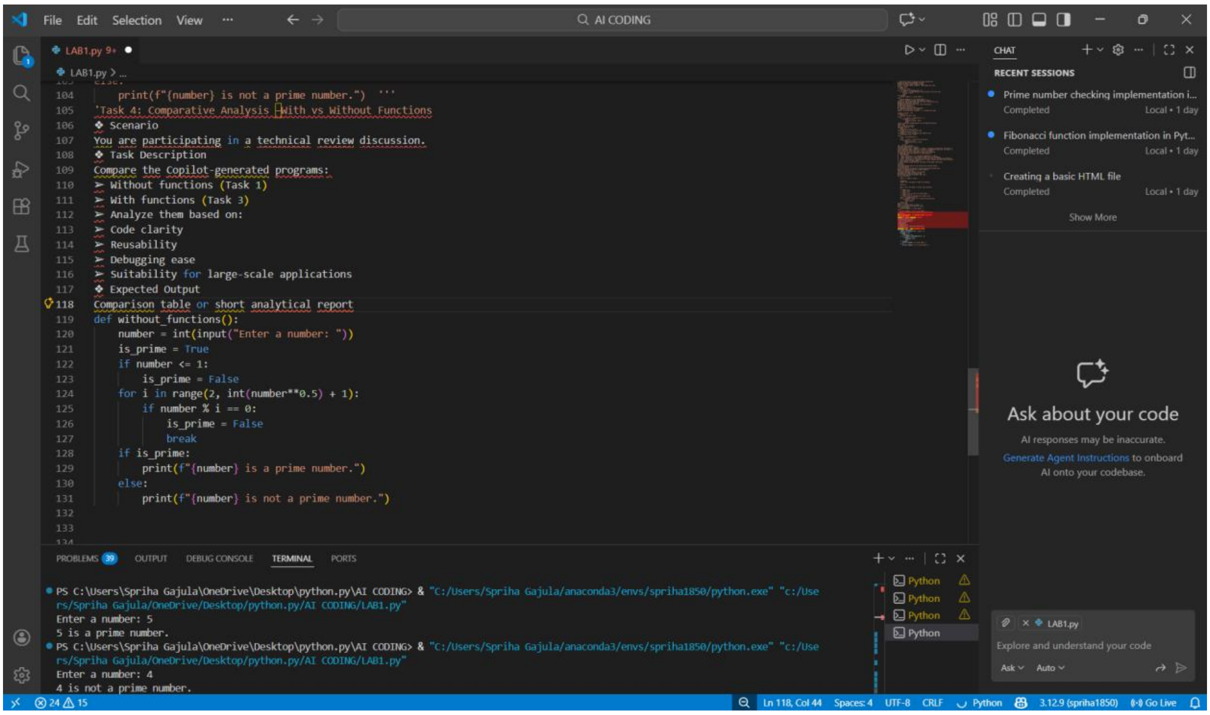
Using a user-defined function makes the prime-checking logic reusable across multiple modules, improving code modularity and maintainability. Returning a Boolean value enables easy integration with conditional statements and other program components.

Task-4: Comparative Analysis –With vs Without Functions

Prompt:

# Compare both code with function without function Analyze and compare two Python programs for checking whether a number is prime

Code & Output :



```
    print(f"{number} is not a prime number.")  '''
'Task 4: Comparative Analysis -With vs Without Functions
❖ Scenario
You are participating in a technical review discussion.
❖ Task Description
Compare the Copilot-generated programs:
➢ Without functions (Task 1)
➢ With functions (Task 3)
➢ Analyze them based on:
➢ Code clarity
➢ Reusability
➢ Debugging ease
➢ Suitability for large-scale applications
❖ Expected Output
Comparison table or short analytical report
def without_functions():
    number = int(input("Enter a number: "))
    is_prime = True
    if number <= 1:
        is_prime = False
    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.")
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Use
rs/Spriha Gajula/OneDrive/Desktop/python.py/AI CODING/LAB1.py"
Enter a number: 5
5 is a prime number.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Use
rs/Spriha Gajula/OneDrive/Desktop/python.py/AI CODING/LAB1.py"
Enter a number: 4
4 is not a prime number.
```

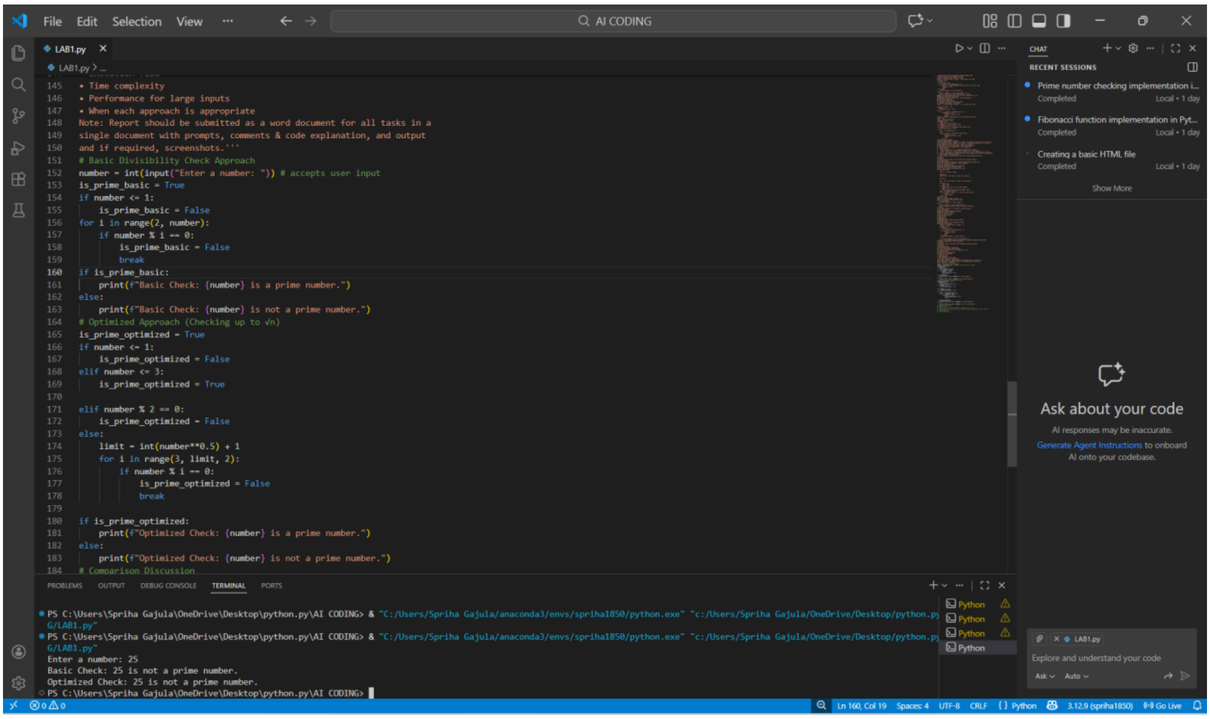
Justification:

Programs written with functions offer better code clarity by separating logic into well-defined blocks, making them easier to read and understand. Function-based designs improve reusability and debugging ease, as changes or fixes can be applied in one place without affecting the entire code.

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

Prompt: Prime Number Check – Basic vs Optimized Approach

Code & output:



```python
• Time complexity
• Performance for large inputs
• When each approach is appropriate
Note: Report should be submitted as a word document for all tasks in a
single document with prompts, comments & code explanation, and output
and if required, screenshots.'''
# Basic Divisibility Check Approach
number = int(input("Enter a number: ")) # accepts user input
is_prime_basic = True
if number <= 1:
    is_prime_basic = False
for i in range(2, number):
    if number % i == 0:
        is_prime_basic = False
        break
if is_prime_basic:
    print(f"Basic Check: {number} is a prime number.")
else:
    print(f"Basic Check: {number} is not a prime number.")
# Optimized Approach (Checking up to √n)
is_prime_optimized = True
if number <= 1:
    is_prime_optimized = False
elif number <= 3:
    is_prime_optimized = True

elif number % 2 == 0:
    is_prime_optimized = False
else:
    limit = int(number**0.5) + 1
    for i in range(3, limit, 2):
        if number % i == 0:
            is_prime_optimized = False
            break

if is_prime_optimized:
    print(f"Optimized Check: {number} is a prime number.")
else:
    print(f"Optimized Check: {number} is not a prime number.")
# Comparison Discussion
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/AI CODING/LAB1.py"
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/AI CODING/LAB1.py"
Enter a number: 25
Basic Check: 25 is not a prime number.
Optimized Check: 25 is not a prime number.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> 
```

Justification:

The basic approach checks divisibility up to N−1, resulting in unnecessary iterations and higher time complexity.

The optimized approach checks only up to √N because any factor larger than √N must have a corresponding smaller factor.