

AI ASSISTED CODING

Hall Ticket No: 2303A51850

Batch:13

Assignment-1.4

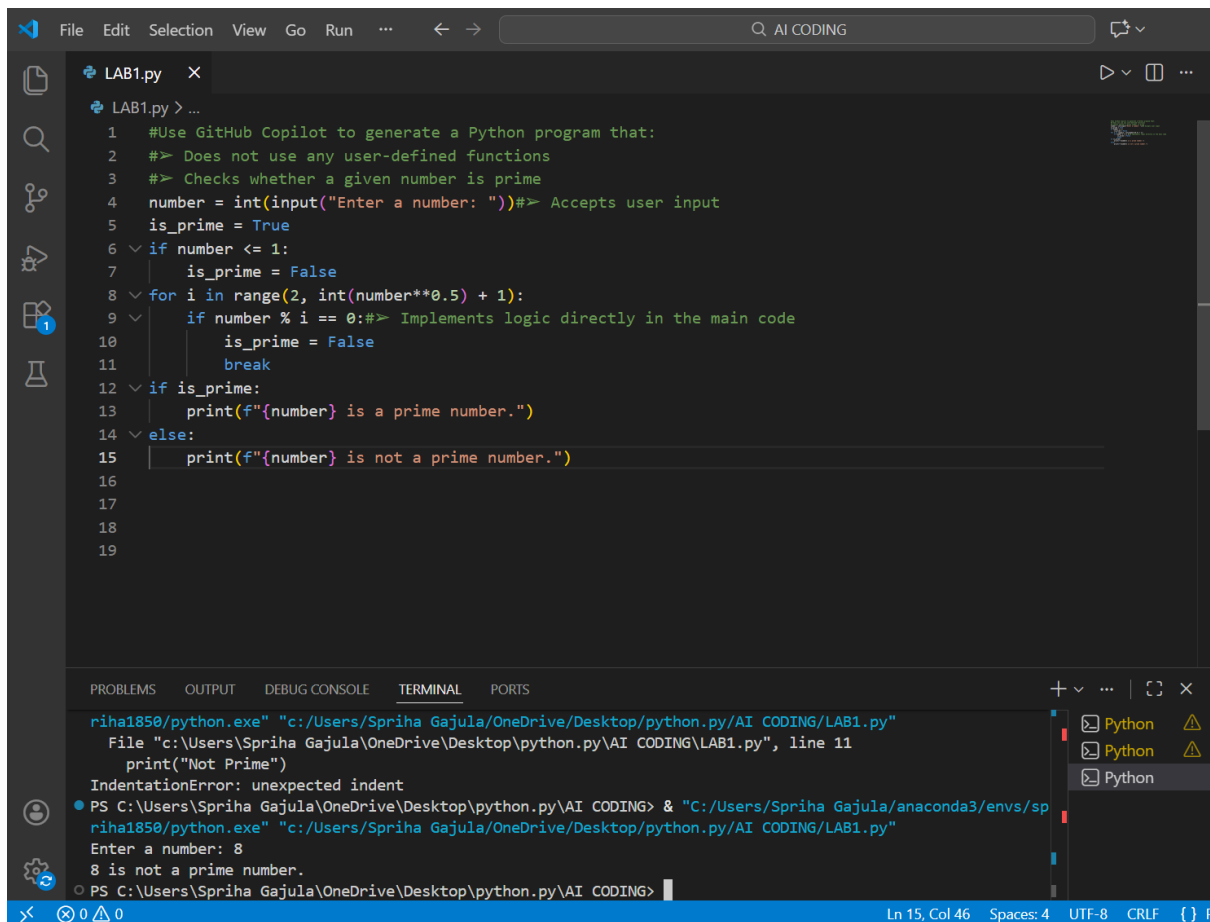
Task-1. AI-Generated Logic Without Modularization (Prime Number Check

Without Functions)

Prompt

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

Code & Output



```
LAB1.py
LAB1.py > ...
1 #Use GitHub Copilot to generate a Python program that:
2 #> Does not use any user-defined functions
3 #> Checks whether a given number is prime
4 number = int(input("Enter a number: "))#> Accepts user input
5 is_prime = True
6 if number <= 1:
7     is_prime = False
8 for i in range(2, int(number**0.5) + 1):
9     if number % i == 0:#> Implements logic directly in the main code
10         is_prime = False
11         break
12 if is_prime:
13     print(f"{number} is a prime number.")
14 else:
15     print(f"{number} is not a prime number.")
16
17
18
19
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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>
```

Ln 15, Col 46 Spaces: 4 UTF-8 CRLF {}

Justification:

This program checks whether a given number is prime using direct conditional logic without defining any functions.

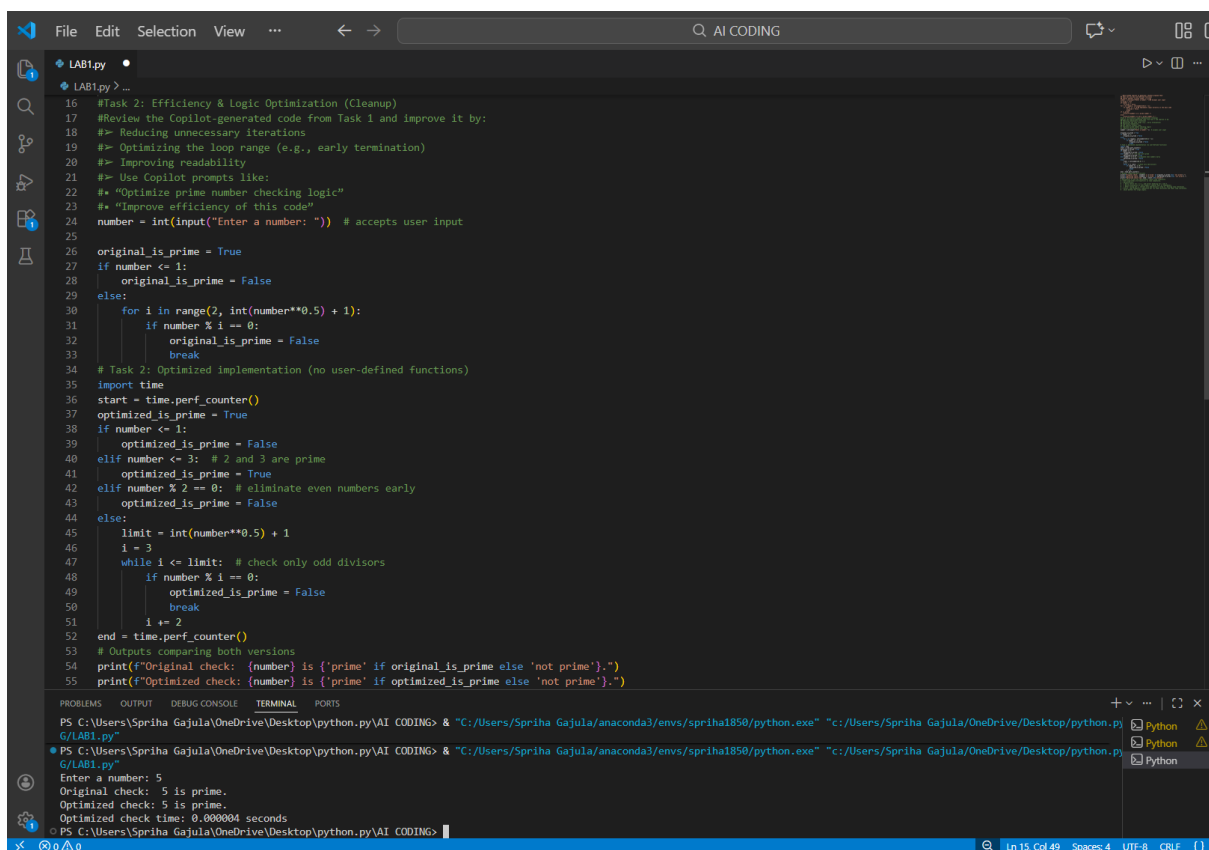
All computations are performed sequentially in a single block, making the logic easy to follow and suitable for beginners.

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 :



```
16 #Task 2: Efficiency & Logic Optimization (Cleanup)
17 #Review the Copilot-generated code from Task 1 and improve it by:
18 #> Reducing unnecessary iterations
19 #> Optimizing the loop range (e.g., early termination)
20 #> Improving readability
21 #> Use Copilot prompts like:
22 #* "Optimize prime number checking logic"
23 #* "Improve efficiency of this code"
24 number = int(input("Enter a number: ")) # accepts user input
25
26 original_is_prime = True
27 if number <= 1:
28     original_is_prime = False
29 else:
30     for i in range(2, int(number**0.5) + 1):
31         if number % i == 0:
32             original_is_prime = False
33             break
34 # Task 2: Optimized implementation (no user-defined functions)
35 import time
36 start = time.perf_counter()
37 optimized_is_prime = True
38 if number <= 1:
39     optimized_is_prime = False
40 elif number <= 3: # 2 and 3 are prime
41     optimized_is_prime = True
42 elif number % 2 == 0: # eliminate even numbers early
43     optimized_is_prime = False
44 else:
45     limit = int(number**0.5) + 1
46     i = 3
47     while i <= limit: # check only odd divisors
48         if number % i == 0:
49             optimized_is_prime = False
50             break
51         i += 2
52 end = time.perf_counter()
53 # Outputs comparing both versions
54 print(f"Original check: {number} is {'prime' if original_is_prime else 'not prime'}.")
55 print(f"Optimized check: {number} is {'prime' if optimized_is_prime else 'not prime'}.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spriha Gajula/anaconda3/envs/sprihal850/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/sprihal850/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.

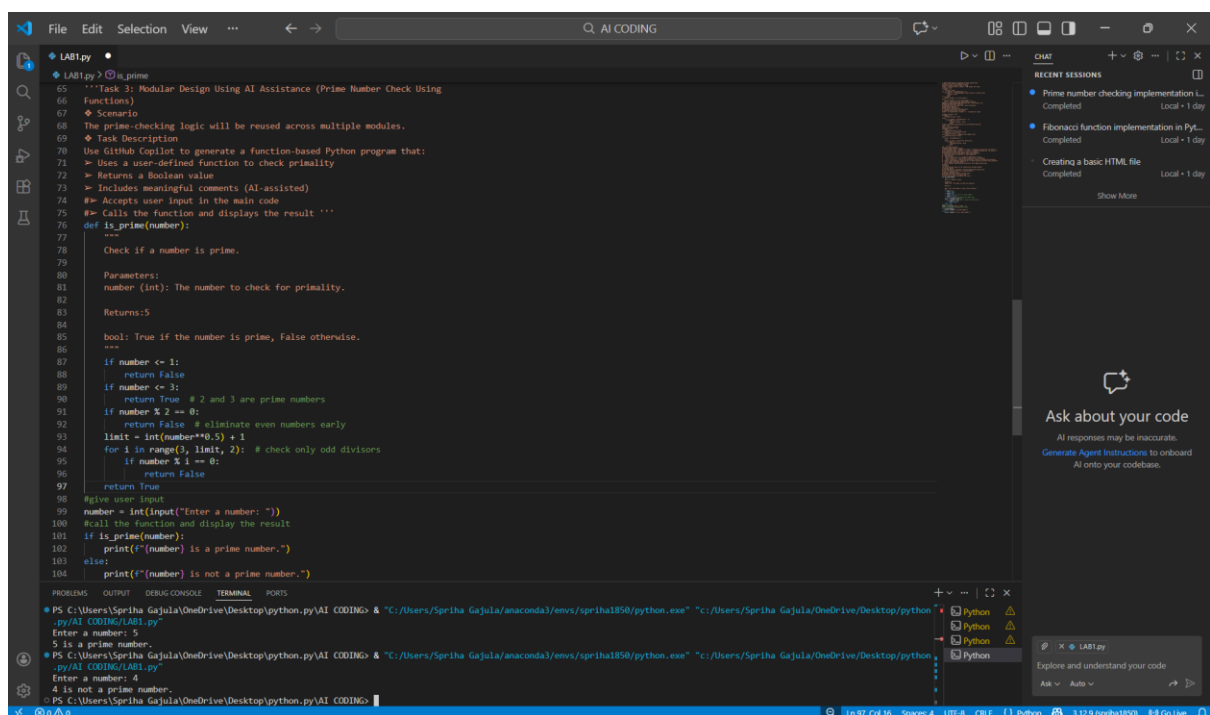
Early termination and simplified conditions lower the overall time complexity while maintaining correct prime number validation.

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 :



```
LAB1.py
65 """Task 3: Modular Design Using AI Assistance (Prime Number Check Using
66 Functions)
67 Scenario
68 The prime-checking logic will be reused across multiple modules.
69 Task Description
70 Use GitHub Copilot to generate a function-based Python program that:
71 - Uses a user-defined function to check primality
72 - Returns a Boolean value
73 - Includes meaningful comments (AI-assisted)
74 - Accepts user input in the main code
75 - Calls the function and displays the result """
76 def is_prime(number):
77     """
78     Check if a number is prime.
79     Parameters:
80     number (int): The number to check for primality.
81     Returns:
82     bool: True if the number is prime, False otherwise.
83     """
84     if number <= 1:
85         return False
86     if number <= 3:
87         return True # 2 and 3 are prime numbers
88     if number % 2 == 0:
89         return False # eliminate even numbers early
90     limit = int(number**0.5) + 1
91     for i in range(3, limit, 2): # check only odd divisors
92         if number % i == 0:
93             return False
94     return True
95
96 # Give user input
97 number = int(input("Enter a number: "))
98 # Call the function and display the result
99 if is_prime(number):
100     print(f"{number} is a prime number.")
101 else:
102     print(f"{number} is not a prime number.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Spruha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spruha Gajula/anaconda3/envs/spruha1850/python.exe" "C:/Users/Spruha Gajula/OneDrive/Desktop/python.py/
AI CODING/LAB1.py"
Enter a number: 5
5 is a prime number.

PS C:\Users\Spruha Gajula\OneDrive\Desktop\python.py\AI CODING> & "C:/Users/Spruha Gajula/anaconda3/envs/spruha1850/python.exe" "C:/Users/Spruha Gajula/OneDrive/Desktop/python.py/
AI CODING/LAB1.py"
Enter a number: 4
4 is not a prime number.

PS C:\Users\Spruha Gajula\OneDrive\Desktop\python.py\AI CODING>
```

Justification:

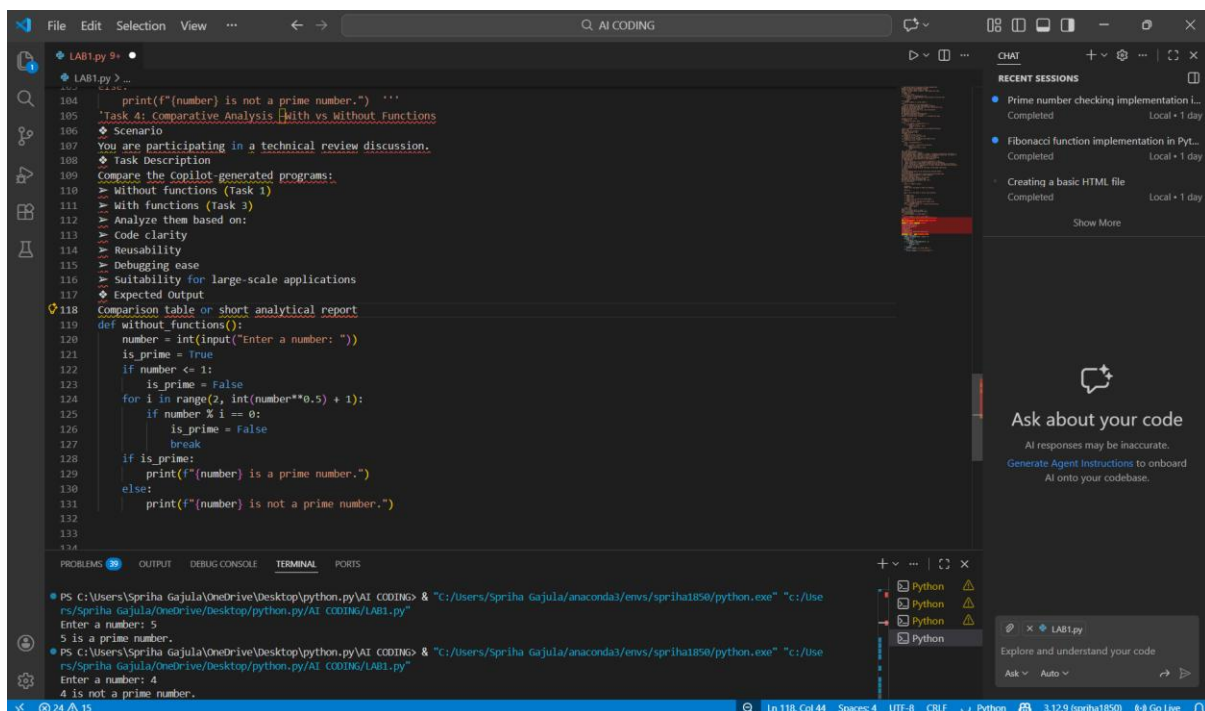
Using a user-defined function makes the prime-checking logic reusable across multiple modules, improving code modularity and maintainability. Returning a Boolean value enables easy integration with conditional statements and other program components.

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 :



The screenshot displays a VS Code editor window with a Python file named LAB1.py. The code implements a function `without_functions()` to check if a number is prime. The function takes user input, checks for divisibility from 2 to \sqrt{n} , and prints the result. The terminal shows the execution of the script, where the user enters 5 and 4, resulting in the output '5 is a prime number.' and '4 is not a prime number.' respectively. The right sidebar shows a chat interface with recent sessions and a prompt to ask about the code.

```
104 print(f"{number} is not a prime number.") '''
105 'Task 4: Comparative Analysis With vs Without Functions
106 Scenario
107 You are participating in a technical review discussion.
108 Task Description
109 Compare the Copilot-generated programs:
110 > without functions (Task 1)
111 > with functions (Task 2)
112 > Analyze them based on:
113 > Code clarity
114 > Reusability
115 > Debugging ease
116 > Suitability for large-scale applications
117 Expected Output
118 Comparison table or short analytical report
119 def without_functions():
120     number = int(input("Enter a number: "))
121     is_prime = True
122     if number <= 1:
123         is_prime = False
124     for i in range(2, int(number**0.5) + 1):
125         if number % i == 0:
126             is_prime = False
127             break
128     if is_prime:
129         print(f"{number} is a prime number.")
130     else:
131         print(f"{number} is not a prime number.")
132
133
134
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py\AI CODING> "C:/Users/Spriha Gajula/anaconda3/envs/spruha1850/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/spruha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/AI CODING/LAB1.py"

Enter a number: 4

4 is not a prime number.

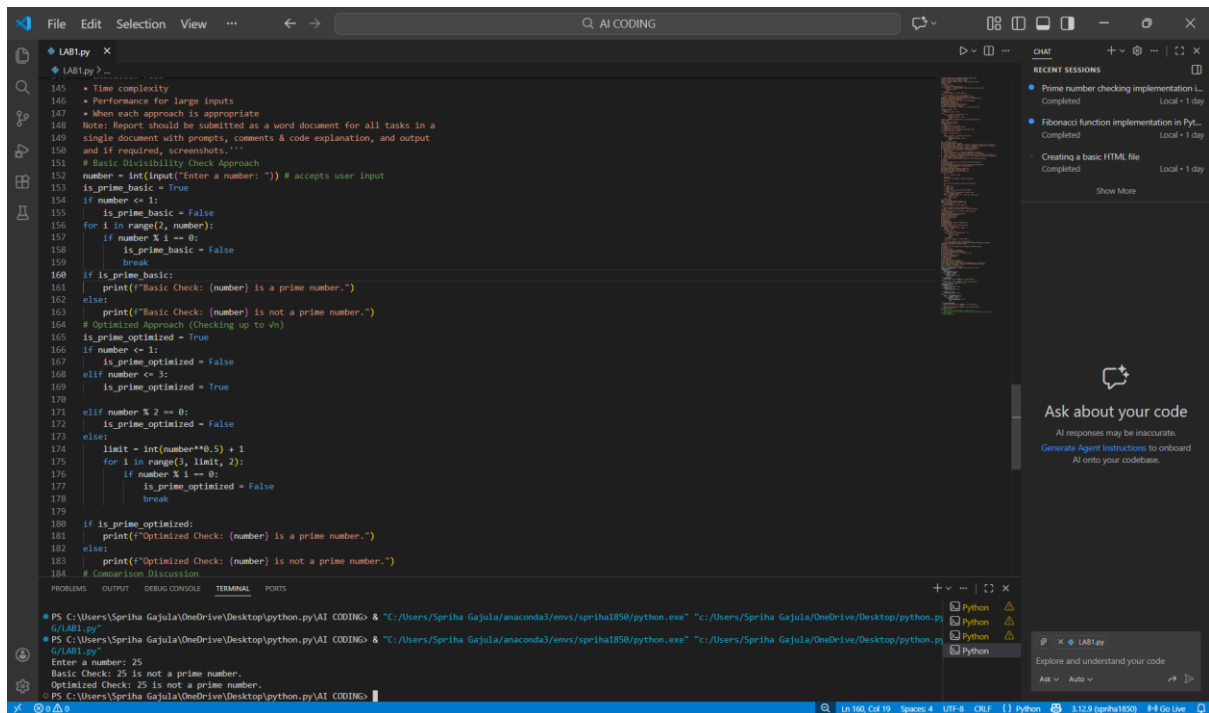
Justification:

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:



The screenshot shows a VS Code editor with a Python file named LAB1.py. The code implements two prime checking functions: a basic approach and an optimized approach. The basic approach checks divisibility from 2 to n-1, while the optimized approach checks up to the square root of n. The terminal output shows the results of running the code with the input 25.

```
LAB1.py
145 • Time complexity
146 • Performance for large inputs
147 • When each approach is appropriate
148 Note: Report should be submitted as a word document for all tasks in a
149 single document with prompts, comments & code explanation, and output
150 and if required, screenshots, ...
151 # Basic Divisibility Check Approach
152 number = int(input("Enter a number: ")) # accepts user input
153 is_prime_basic = True
154 if number <= 1:
155     is_prime_basic = False
156 for i in range(2, number):
157     if number % i == 0:
158         is_prime_basic = False
159         break
160 if is_prime_basic:
161     print(f"Basic Check: {number} is a prime number.")
162 else:
163     print(f"Basic Check: {number} is not a prime number.")
164 # Optimized Approach (Checking up to sqrt)
165 is_prime_optimized = True
166 if number <= 1:
167     is_prime_optimized = False
168 elif number <= 3:
169     is_prime_optimized = True
170 elif number % 2 == 0:
171     is_prime_optimized = False
172 else:
173     limit = int(number**0.5) + 1
174     for i in range(3, limit, 2):
175         if number % i == 0:
176             is_prime_optimized = False
177             break
178 if is_prime_optimized:
179     print(f"Optimized Check: {number} is a prime number.")
180 else:
181     print(f"Optimized Check: {number} is not a prime number.")
182 # Comparison Discussion
```

Terminal Output:

```
PS C:\Users\Spruha Gajula\OneDrive\Desktop\python.py> AI CODING & "C:/Users/Spruha Gajula/anaconda3/envs/spruha1850/python.exe" "C:/Users/Spruha Gajula/OneDrive/Desktop/python.py"
G:\LAB1.py
Enter a number: 25
Basic Check: 25 is not a prime number.
Optimized Check: 25 is not a prime number.
```

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 \sqrt{N} because any factor larger than \sqrt{N} must have a corresponding smaller factor.

