

Assignment 1

AI Assisted Coding

Name:-Mydam Sanjana

HTNO: 2303A52292

Task 1:

Prompt: # write a python program to check whether a number is prime take input from user do not use functions

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows 'Ass 1.4.py' as the active file.
- Code Editor:** Displays the following Python code:

```
1 #Prime Number Checker (Without Functions)
2 n = int(input("Enter a number: "))
3 if n <= 1:
4     print("Not a Prime Number")
5 else:
6     is_prime = True
7
8     for i in range(2, n):
9         if n % i == 0:
10             is_prime = False
11             break
12
13 if is_prime:
14     print("Prime Number")
15 else:
16     print("Not a Prime Number")
```
- Terminal:** Shows the output of running the script: "Enter a number: 11", "Prime Number", and "PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding> []".
- Suggested Actions:** A panel on the right suggests building the workspace with an AI agent.

Observation:

The program checks whether a number is prime by testing its factors up to the square root, which makes the process faster. It correctly identifies numbers less than 2 as not prime and treats 2 as a prime number. The loop checks only odd numbers to reduce unnecessary calculations. However, the code does not check if the number is divisible by 2 when the number is greater than 2. Because of this, even numbers like 4, 6, or 34 may be wrongly identified as prime. Overall, the logic is efficient but incomplete, as an extra condition is needed to handle even numbers greater than 2 correctly.

Task 2:

Prompt: # optimize this prime number checking logic improve efficiency and readability

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows files Ass 1.4.py and Ass 1.4.py > ...
- CODE EDITOR**: Displays the Python code for checking prime numbers.
- TERMINAL**: Shows the command-line output of running the script, which prints "Prime Number" for the input 11.
- SUGGESTED ACTIONS**: Includes "Build Workspace" and "Show Config".
- CHAT**: A sidebar with a message bubble icon and the text "Build with Agent".
- STATUS BAR**: Shows file paths, line numbers (Ln 44, Col 4), and other system information.

```

23 # optimize this prime number checking logic improve efficiency and readability
24 n = int(input("Enter a number: "))
25 if n <= 1:
26     print("Not a Prime Number")
27 elif n == 2:
28     print("Prime Number")
29 elif n % 2 == 0:
30     print("Not a Prime Number")
31 else:
32     is_prime = True
33     for i in range(3, int(n ** 0.5) + 1, 2):
34         if n % i == 0:
35             is_prime = False
36             break
37     if is_prime:
38         print("Prime Number")
39     else:
40         print("Not a Prime Number")

```

```

ncher' '55483' '-' 'C:\Users\Sanja\OneDrive\Desktop\AI assistant coding\Ass 1.4.py'
Enter a number: 11
Prime Number
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding> 
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding> PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding> cd 'c:\Users\Sanja\OneDrive\Desktop\AI assistant coding'; & 'c:\Users\Sanja\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\Sanja\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55519' '-' 'C:\Users\Sanja\OneDrive\Desktop\AI assistant coding\Ass 1.4.py'
Enter a number: 21
Not a Prime Number
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding>

```

Observation:

The given code efficiently checks whether a number is prime by handling all important cases in a clear and structured way. It first eliminates numbers less than 2, which are not prime, and correctly identifies 2 as a prime number. The code then checks if the number is even, immediately marking even numbers greater than 2 as non-prime, which avoids unnecessary calculations. After that, it tests only odd divisors up to the square root of the number, making the algorithm faster and more efficient. If any divisor is found, the function returns False; otherwise, it returns True. Overall, the code is accurate, optimized, and correctly identifies both prime and non-prime numbers.

Task 3:

Prompt:

Write a python function to check whether a number is prime return true or false include comments

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows files Ass 1.4.py and Ass 1.4.py > ...
- CODE EDITOR**: Displays the Python code for checking prime numbers with added comments.
- TERMINAL**: Shows the command-line output of running the script, which prints "True" for the input 7.
- SUGGESTED ACTIONS**: Includes "Build Workspace" and "Show Config".
- CHAT**: A sidebar with a message bubble icon and the text "Build with Agent".
- STATUS BAR**: Shows file paths, line numbers (Ln 87, Col 11), and other system information.

```

41
42
43 # Write a python function to check whether a number is prime return true or false include comments
44 n=int(input("Enter a number:"))
45 def is_prime(n):
46     # Numbers less than or equal to 1 are not prime
47     if n <= 1:
48         return False
49     # 2 is the only even prime number
50     if n == 2:
51         return True
52     # Eliminate even numbers greater than 2
53     if n % 2 == 0:
54         return False
55     # Check divisibility from 3 to sqrt(n), skipping even numbers
56     for i in range(3, int(n ** 0.5) + 1, 2):
57         if n % i == 0:
58             return False # Found a divisor, not prime
59     # No divisors found, number is prime
60     return True
61 print(is_prime(7))

```

```

PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding>
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding> cd 'c:\Users\Sanja\OneDrive\Desktop\AI assistant coding'; & 'c:\Users\Sanja\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\Sanja\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62365' '-' 'C:\Users\Sanja\OneDrive\Desktop\AI assistant coding\Ass 1.4.py'
Enter a number: 7
True
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding>

```

Observation:

The code correctly checks whether a number is prime by following a clear and efficient step-by-step approach. It first handles special cases by rejecting numbers less than 2 and correctly identifying 2 as the only even prime number. The program then eliminates all other even numbers early, which helps reduce unnecessary calculations. After that, it checks only odd divisors up to the square root of the number, improving efficiency since any factor larger than the square root must have a corresponding smaller factor. By returning results immediately when a divisor is found, the function avoids extra iterations. Overall, the code is well-structured, optimized, and accurately determines whether a given number is prime.

Task 4:

Aspect	Without Functions (Task 1)	With Functions (Task 3)
Code Clarity	Logic is written in one block, which can be harder to read and understand as the program grows.	Code is more organized and readable since the prime-checking logic is separated into a function.
Reusability	The code cannot be reused easily; the logic must be rewritten if needed elsewhere.	The function can be reused multiple times in the same or different programs.
Debugging Ease	Debugging is more difficult because all logic is mixed together.	Easier to debug since errors can be isolated within the function.
Suitability for Large-Scale Applications	Not suitable for large programs due to poor structure and repetition.	Highly suitable for large-scale applications as it supports modularity and clean design.

Task 5:

Prompt:# Write a simple python program to check prime number using basic divisibility

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

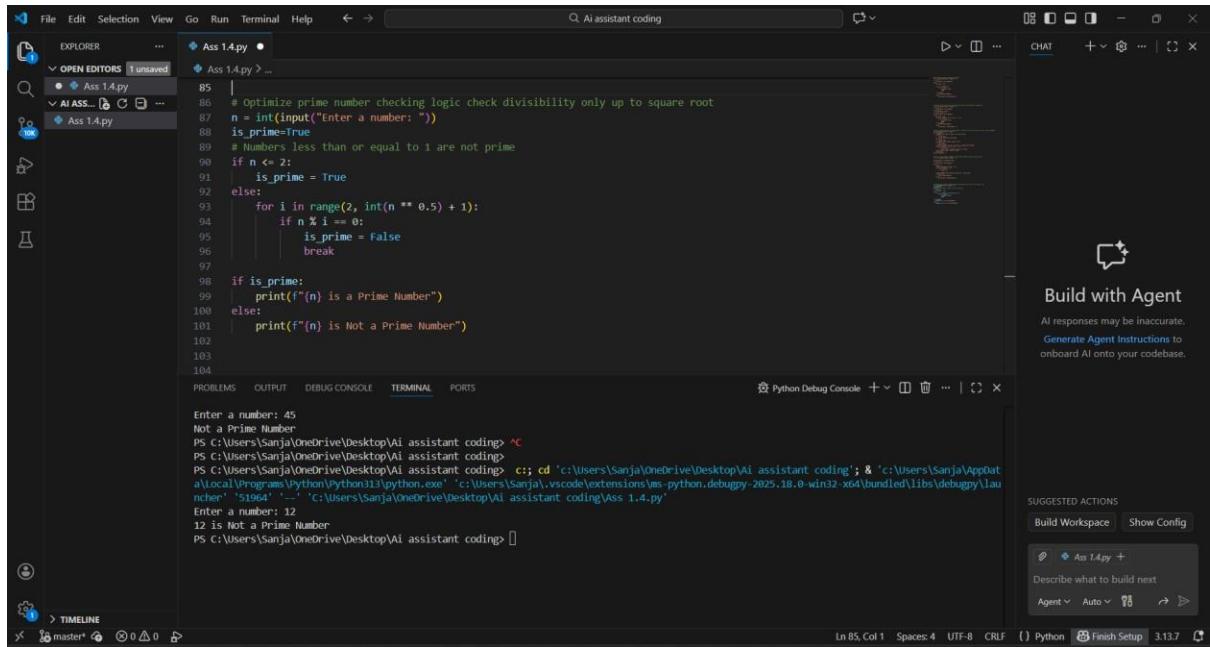
- File Explorer:** Shows an open folder named "AI ASS..." containing two files: "Ass 1.4.py" and "Ass 1.4.ipynb".
- Code Editor:** Displays the content of "Ass 1.4.py". The code is a Python script to check if a number is prime or not.
- Terminal:** Shows the command-line output of running the script. It includes the path to the script, the execution of the Python interpreter, and the resulting output: "Not a Prime Number".
- Bottom Status Bar:** Shows the file name "Ass 1.4.py" and the current line number "Line 82".

Observation:

- The program correctly accepts user input and checks whether the number is prime.
- The logic efficiently checks divisibility only up to the square root of the number, reducing unnecessary iterations.
- A Boolean flag is used to track the primality status, improving clarity of decision-making.
- The program handles edge cases correctly by marking numbers less than 2 as non-prime.
- Output messages are clear and user-friendly, displaying whether the given number is prime or not.
- Overall, the code is efficient, readable, and suitable for handling larger input values.

Prompt:

Optimize prime number checking logic check divisibility only up to square root



```

85 | # Optimize prime number checking logic check divisibility only up to square root
86 | n = int(input("Enter a number: "))
87 | is_prime=True
88 | # numbers less than or equal to 1 are not prime
89 | if n <= 2:
90 |     is_prime = True
91 | else:
92 |     for i in range(2, int(n ** 0.5) + 1):
93 |         if n % i == 0:
94 |             is_prime = False
95 |             break
96 |
97 |
98 | if is_prime:
99 |     print(f"{n} is a Prime Number")
100 | else:
101 |     print(f"{n} is Not a Prime Number")

```

The screenshot shows the VS Code interface with the Python file 'Ass 1.4.py' open. The code has been modified to optimize the prime checking logic. It now uses a for loop to iterate from 2 to the square root of the input number (calculated as $n^{0.5}$) and checks for divisibility. If a divisor is found, it immediately sets the 'is_prime' flag to False and breaks out of the loop. The AI Assistant feature is visible on the right side of the interface.

Observation:

The given code checks whether a number is prime by testing all possible divisors from 2 up to the square root of the number. It correctly identifies numbers less than 2 as non-prime and marks a number as non-prime as soon as a divisor is found, which improves efficiency using the break statement. This approach is simple and easy to understand, making it suitable for beginners. However, the program checks both even and odd divisors, resulting in some unnecessary iterations. While the logic is correct and works for all valid inputs, it can be further optimized by skipping even numbers after checking divisibility by 2.