

Assignment-2

Name: Sushma Purella

H. No:2303A52188

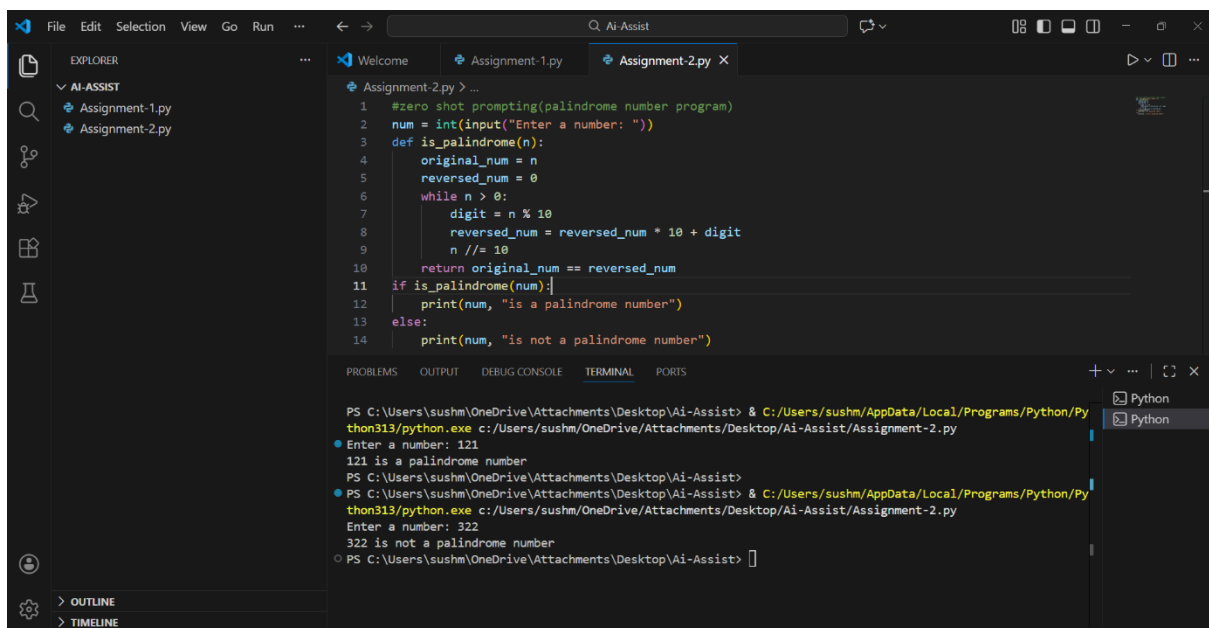
Batch:44

Question 1: Zero-Shot Prompting (Palindrome Number Program)

Prompt:

Write a Python program to check whether a given number is a palindrome or not and display the result.

Code:



The screenshot shows a Python IDE with a file explorer on the left containing 'Assignment-1.py' and 'Assignment-2.py'. The main editor displays the code for 'Assignment-2.py', which is a zero-shot prompting program to check if a number is a palindrome. The code defines a function 'is_palindrome(n)' that reverses the digits of 'n' and compares it with the original. The program prompts the user to enter a number and prints the result. The terminal at the bottom shows the execution of the program twice: first with input '121' resulting in '121 is a palindrome number', and second with input '322' resulting in '322 is not a palindrome number'.

```
1 #zero shot prompting(palindrome number program)
2 num = int(input("Enter a number: "))
3 def is_palindrome(n):
4     original_num = n
5     reversed_num = 0
6     while n > 0:
7         digit = n % 10
8         reversed_num = reversed_num * 10 + digit
9         n //= 10
10    return original_num == reversed_num
11 if is_palindrome(num):
12     print(num, "is a palindrome number")
13 else:
14     print(num, "is not a palindrome number")
```

PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
Enter a number: 121
121 is a palindrome number
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist>
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
Enter a number: 322
322 is not a palindrome number
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist>

Observation:

The model is able to generate a working palindrome program without any prior examples.

The generated code correctly reverses the number and compares it with the original.

However, the prompt does not specify input validation or optimization, so the solution may not handle edge cases (like negative numbers) efficiently.

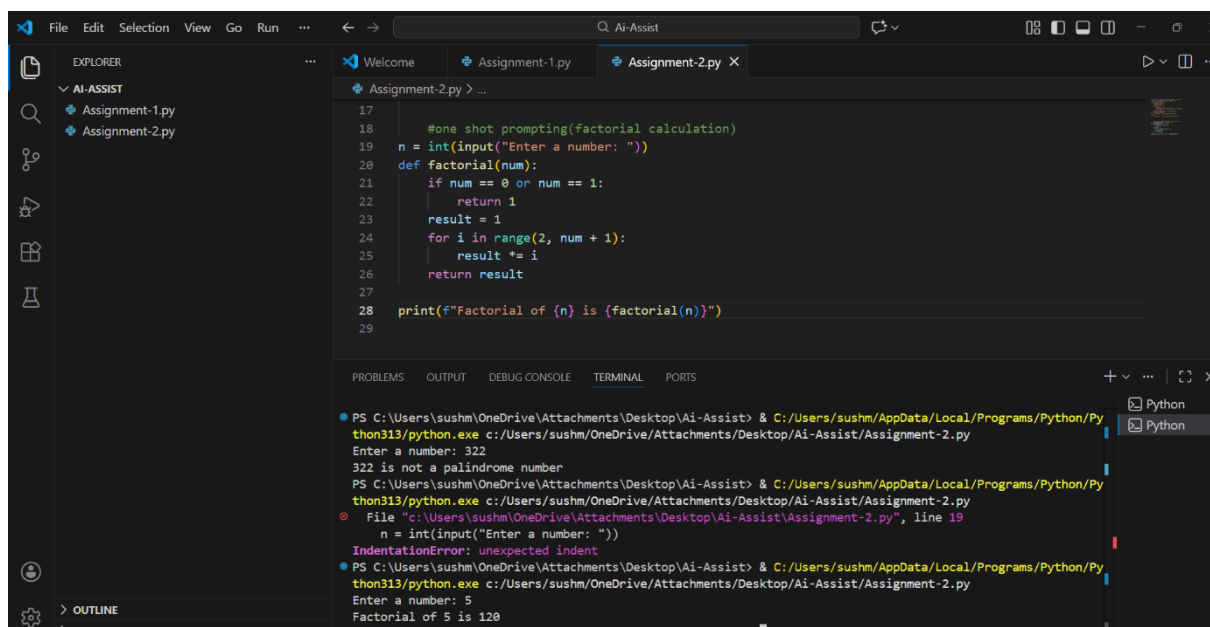
This shows that zero-shot prompting works for simple problems, but for better control and optimized output, structured or context-managed prompts are preferred.

Question 2: One-Shot Prompting (Factorial Calculation)

Prompt:

Write a Python program to calculate the factorial of a given number

Code:



The screenshot shows a Visual Studio Code editor with a Python file named 'Assignment-2.py'. The code defines a factorial function using a loop and handles base cases for 0 and 1. The terminal window shows the execution of the program, where the user enters '322' and '5'. The output for '322' is '322 is not a palindrome number' (which is not part of the provided code but appears in the terminal output), and the output for '5' is 'Factorial of 5 is 120'.

```
17 #one shot prompting(factorial calculation)
18 n = int(input("Enter a number: "))
19 def factorial(num):
20     if num == 0 or num == 1:
21         return 1
22     result = 1
23     for i in range(2, num + 1):
24         result *= i
25     return result
26
27
28 print(f"Factorial of {n} is {factorial(n)}")
29
```

```
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
Enter a number: 322
322 is not a palindrome number
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
File "C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist\Assignment-2.py", line 19
    n = int(input("Enter a number: "))
        ^
IndentationError: unexpected indent
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist>
```

Observation:

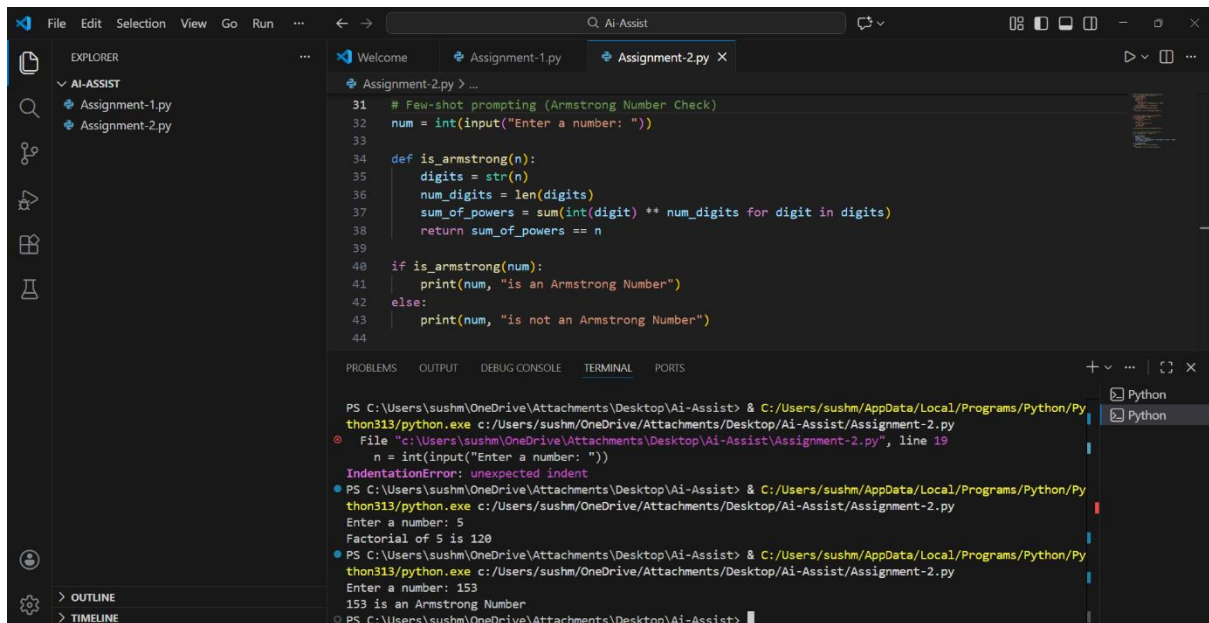
- The provided example helps the model understand the expected logic and output format.
- The generated code correctly handles base cases such as 0 and 1.
- The program uses an iterative approach, which is efficient and easy to understand.
- Compared to zero-shot prompting, one-shot prompting produces more accurate and structured output.

Question 3: Few-Shot Prompting (Armstrong Number Check)

Prompt:

Write a Python program to check whether a given number is an Armstrong Number

Code:



```
31 # Few-shot prompting (Armstrong Number Check)
32 num = int(input("Enter a number: "))
33
34 def is_armstrong(n):
35     digits = str(n)
36     num_digits = len(digits)
37     sum_of_powers = sum(int(digit) ** num_digits for digit in digits)
38     return sum_of_powers == n
39
40 if is_armstrong(num):
41     print(num, "is an Armstrong Number")
42 else:
43     print(num, "is not an Armstrong Number")
44
```

Terminal Output:

```
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
Enter a number: 153
153 is an Armstrong Number
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist>
```

Observation:

- Providing multiple examples helps the model clearly understand the definition of an Armstrong number.
- The generated program correctly computes the sum of each digit raised to the power of the number of digits.
- The output format closely matches the examples.
- Few-shot prompting produces more accurate and consistent results than zero-shot and one-shot prompting.

Question 4: Context-Managed Prompting (Optimized Number Classification)

Prompt:

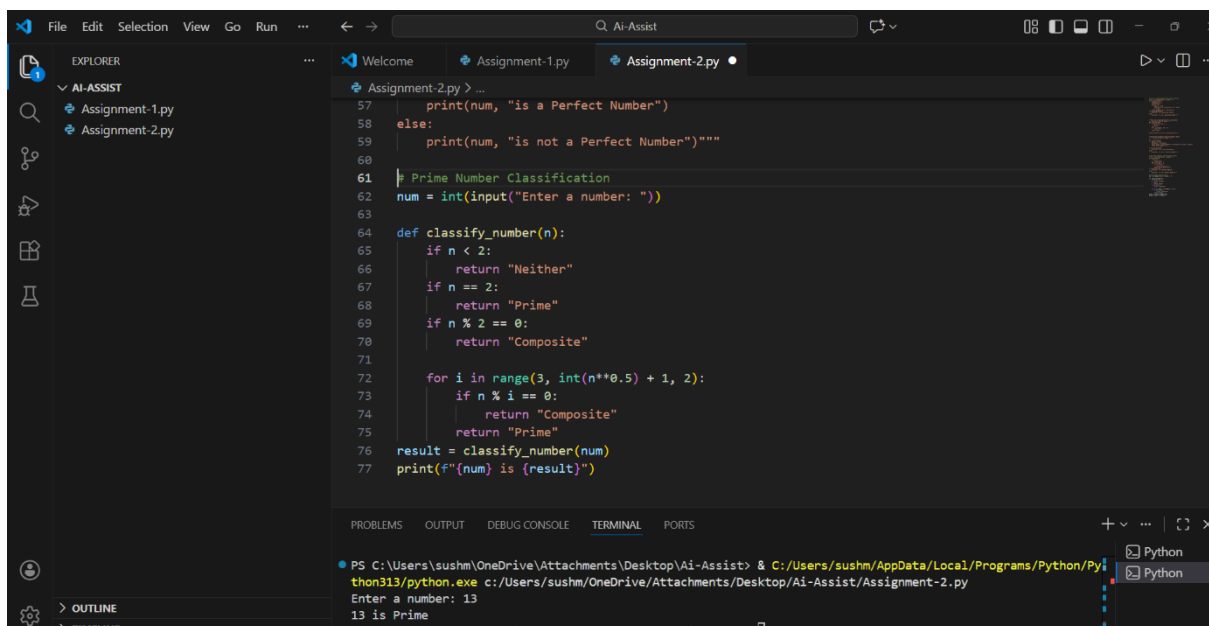
You are an experienced Python developer.

Generate an optimized Python program that classifies a given integer as Prime, Composite, or Neither.

Instructions & Constraints:

- Accept only valid integer input.
- Handle special cases such as negative numbers, 0, and 1 correctly.
- Use an efficient algorithm (check divisibility only up to \sqrt{n}).
- Avoid unnecessary loops or redundant checks.
- The program must clearly print whether the number is *Prime*, *Composite*, or *Neither*.
- Do not include explanations or comments—only clean, executable Python code.

Code:

A screenshot of the Visual Studio Code (VS Code) editor interface. The main window displays a Python file named 'Assignment-2.py'. The code defines a function 'classify_number(n)' that checks if a number is prime, composite, or neither. It handles special cases for n < 2, n == 2, and n % 2 == 0. For other numbers, it checks divisibility up to the square root of n. The program prompts the user to 'Enter a number:' and prints the result. The terminal at the bottom shows the command to run the script and the output: 'Enter a number: 13' followed by '13 is Prime'. The Explorer sidebar on the left shows the project structure with 'Assignment-1.py' and 'Assignment-2.py' under the 'AI-ASSIST' folder. The Run and Debug sidebar on the right shows the Python interpreter path and the execution of the script.

```
57 print(num, "is a Perfect Number")
58 else:
59     print(num, "is not a Perfect Number")
60
61 # Prime Number Classification
62 num = int(input("Enter a number: "))
63
64 def classify_number(n):
65     if n < 2:
66         return "Neither"
67     if n == 2:
68         return "Prime"
69     if n % 2 == 0:
70         return "Composite"
71
72     for i in range(3, int(n**0.5) + 1, 2):
73         if n % i == 0:
74             return "Composite"
75         return "Prime"
76 result = classify_number(num)
77 print(f"{num} is {result}")
```

PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py

Enter a number: 13

13 is Prime

Observation:

- The program efficiently classifies a number as Prime, Composite, or Neither.
- Special cases such as numbers less than 2 are correctly handled as Neither.

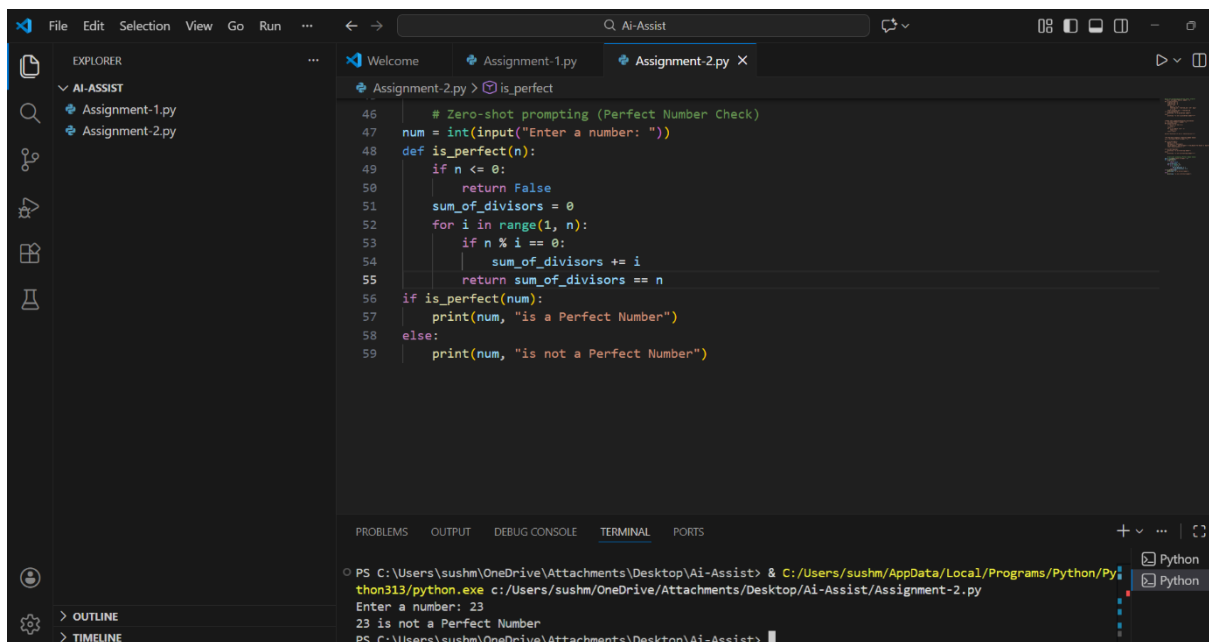
- The logic optimizes performance by checking divisibility only up to the square root of the number.
- Even numbers greater than 2 are immediately classified as Composite, reducing unnecessary iterations.
- This approach is more efficient and accurate compared to basic prime-checking methods that test all numbers up to n.

Question 5: Zero-Shot Prompting (Perfect Number Check)

Prompt:

Write a Python program to check whether a given number is a Perfect Number or not and display the result.

Code:



```

46 # Zero-shot prompting (Perfect Number Check)
47 num = int(input("Enter a number: "))
48 def is_perfect(n):
49     if n <= 0:
50         return False
51     sum_of_divisors = 0
52     for i in range(1, n):
53         if n % i == 0:
54             sum_of_divisors += i
55     return sum_of_divisors == n
56 if is_perfect(num):
57     print(num, "is a Perfect Number")
58 else:
59     print(num, "is not a Perfect Number")

```

```

PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
Enter a number: 23
23 is not a Perfect Number
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist>

```

Observation:

- The program checks whether the sum of all proper divisors of a number is equal to the number itself.
- It correctly handles invalid inputs such as zero or negative numbers.
- The logic is simple and easy to understand, making it suitable for basic problem understanding.

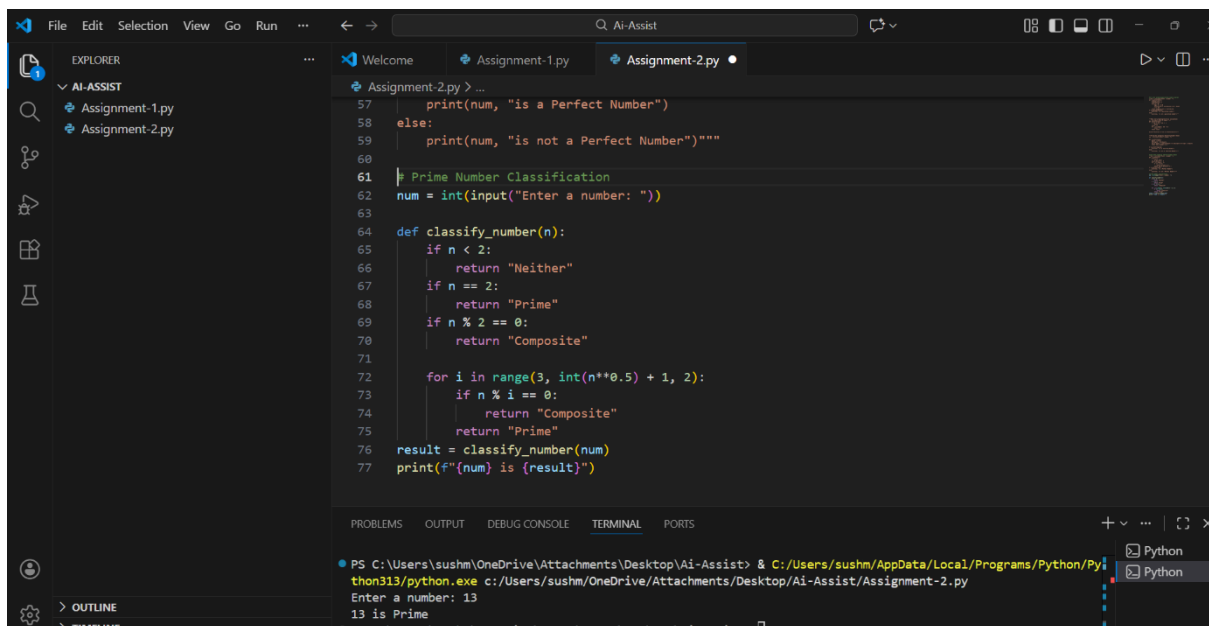
- However, since no optimization constraints were specified in the prompt, the program uses a brute-force approach, which may be inefficient for large numbers.
- This demonstrates that zero-shot prompting works well for simple tasks but may lack efficiency and control.

Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

Prompt:

Write a Python program to classify a given number as Prime, Composite, or Neither and display the result.

Code:



```

57     print(num, "is a Perfect Number")
58 else:
59     print(num, "is not a Perfect Number")"""
60
61 # Prime Number Classification
62 num = int(input("Enter a number: "))
63
64 def classify_number(n):
65     if n < 2:
66         return "Neither"
67     if n == 2:
68         return "Prime"
69     if n % 2 == 0:
70         return "Composite"
71
72     for i in range(3, int(n**0.5) + 1, 2):
73         if n % i == 0:
74             return "Composite"
75     return "Prime"
76 result = classify_number(num)
77 print(f"{num} is {result}")
  
```

Terminal Output:

```

PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-2.py
Enter a number: 13
13 is Prime
  
```

Observation:

- The program correctly classifies numbers less than 2 as Neither.
- It identifies 2 as a prime number and eliminates even numbers early to improve efficiency.
- The divisibility check is optimized by iterating only up to the square root of the number and skipping even values.
- This approach reduces time complexity compared to naive methods.

- The logic ensures accurate classification with improved performance.