

Assignment -3.1

Hall ticket no:2303A510A6

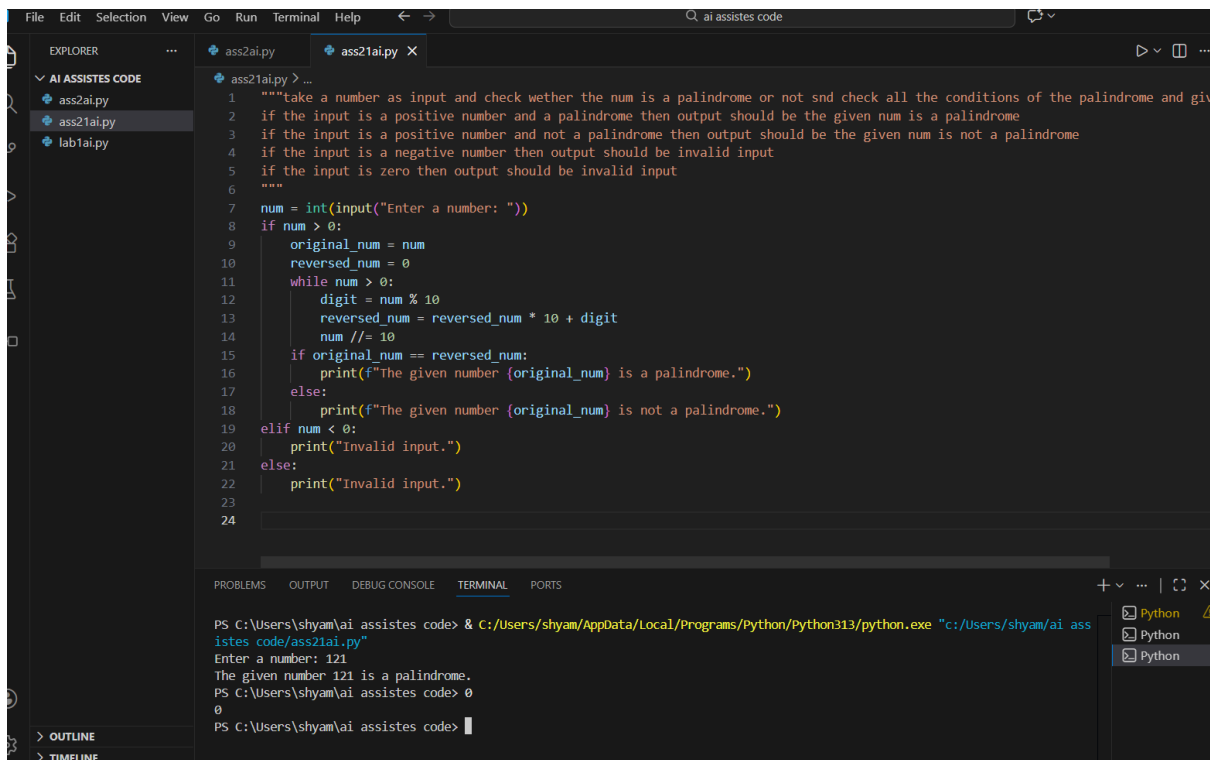
Batch no:02

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

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a palindrome.

Task:

- Record the AI-generated code.
- Test the code with multiple inputs.
- Identify any logical errors or missing edge-case handling.



```
File Edit Selection View Go Run Terminal Help
ai assistes code

EXPLORER
AI ASSISTES CODE
  ass2ai.py
  ass21ai.py
  lab1ai.py

ass21ai.py > ...
1 """take a number as input and check wether the num is a palindrome or not snd check all the conditions of the palindrome and give the output accordingly
2 if the input is a positive number and a palindrome then output should be the given num is a palindrome
3 if the input is a positive number and not a palindrome then output should be the given num is not a palindrome
4 if the input is a negative number then output should be invalid input
5 if the input is zero then output should be invalid input
6 """
7 num = int(input("Enter a number: "))
8 if num > 0:
9     original_num = num
10    reversed_num = 0
11    while num > 0:
12        digit = num % 10
13        reversed_num = reversed_num * 10 + digit
14        num //= 10
15    if original_num == reversed_num:
16        print(f"The given number {original_num} is a palindrome.")
17    else:
18        print(f"The given number {original_num} is not a palindrome.")
19 elif num < 0:
20     print("Invalid input.")
21 else:
22     print("Invalid input.")
23
24

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\shyam\ai assistes code> & C:/Users/shyam/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/shyam/ai assistes code/ass21ai.py"
Enter a number: 121
The given number 121 is a palindrome.
PS C:\Users\shyam\ai assistes code> 0
0
PS C:\Users\shyam\ai assistes code> |
```

write a one-shot prompt by providing one input-output example and ask the AI to generate a Python function to compute the factorial of a given number.

Example:

Input: 5 → Output: 120

Task:

- Compare the generated code with a zero-shot solution.

- Examine improvements in clarity and correctness.

The screenshot shows a VS Code editor with a file explorer on the left containing files like `ass2ai.py`, `ass21ai.py`, `ass22ai.py`, and `lab1ai.py`. The main editor window displays a Python script for calculating the factorial of a number. The script includes a docstring, a function definition `def factorial(n):`, and a main block for user input and output. The terminal at the bottom shows the command to run the script and the resulting output for the input 5.

```

1  """
2  compute the factorial of a given number.
3  """
4
5  def factorial(n):
6      if n < 0:
7          return "Invalid input."
8      elif n == 0:
9          return 1
10     else:
11         result = 1
12         for i in range(1, n + 1):
13             result *= i
14         return result
15
16 # Example usage:
17 number = int(input("Enter a number: "))
18 print(f"The factorial of {number} is {factorial(number)}.")
19

```

```

PS C:\Users\shyam\ai assistes code> & C:/Users/shyam/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"
Enter a number: 5
The factorial of 5 is 120.
PS C:\Users\shyam\ai assistes code>

```

Aspect	One-Shot Prompting	Zero-Shot Prompting
Guidance	Includes example (5 → 120)	No example provided
Clarity	More explicit task understanding	Relies on model's assumption
Approach	Iterative solution	Recursive solution
Readability	Easier for beginners	Slightly more complex
Correctness	Correct for non-negative integers	Correct but risk of recursion depth

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

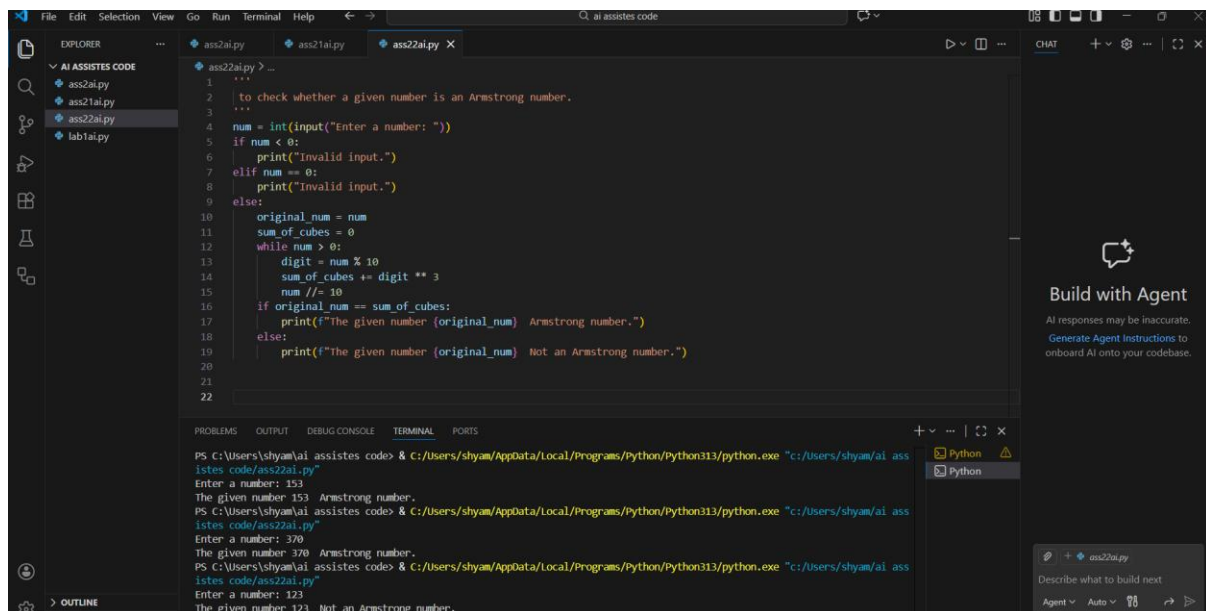
Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python function to check whether a given number is an Armstrong number.

Examples:

- Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

Task:

- Analyze how multiple examples influence code structure and accuracy.
- Test the function with boundary values and invalid inputs.



The screenshot shows a VS Code editor with a Python file named `ass22ai.py`. The code is a function to check if a number is an Armstrong number. It takes an input number, checks for invalid inputs (non-integer or zero), and then calculates the sum of the cubes of its digits. If the sum equals the original number, it's an Armstrong number; otherwise, it's not.

```
1 """  
2 to check whether a given number is an Armstrong number.  
3 """  
4 num = int(input("Enter a number: "))  
5 if num < 0:  
6     print("Invalid input.")  
7 elif num == 0:  
8     print("Invalid input.")  
9 else:  
10    original_num = num  
11    sum_of_cubes = 0  
12    while num > 0:  
13        digit = num % 10  
14        sum_of_cubes += digit ** 3  
15        num //= 10  
16    if original_num == sum_of_cubes:  
17        print(f"The given number {original_num} Armstrong number.")  
18    else:  
19        print(f"The given number {original_num} Not an Armstrong number.")  
20  
21  
22
```

The terminal output shows the program being run with three different inputs:

```
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\shyam\ai assistes code\ass22ai.py"  
Enter a number: 153  
The given number 153 Armstrong number.  
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\shyam\ai assistes code\ass22ai.py"  
Enter a number: 370  
The given number 370 Armstrong number.  
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\shyam\ai assistes code\ass22ai.py"  
Enter a number: 123  
The given number 123 Not an Armstrong number.
```

Influence of Multiple Examples on Code Structure and Accuracy

- Multiple examples clearly demonstrate **both positive and negative cases**, reducing ambiguity.
- The AI correctly identifies that:
 - Each digit must be raised to the **number of digits**.
 - The sum must be compared with the original number.

Few-shot prompting helps the AI infer:

- The definition of an Armstrong number
- The expected output wording
- The correct handling of digit extraction and exponentiation

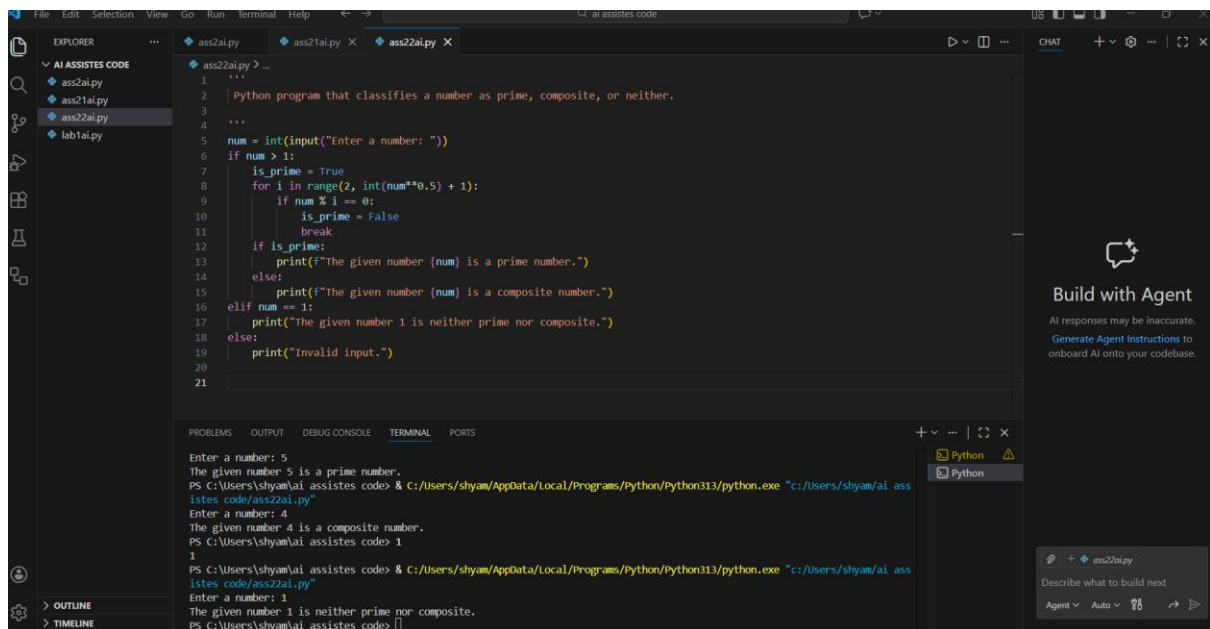
(Optional Extension)

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

Design a context-managed prompt with clear instructions and constraints to generate an optimized Python program that classifies a number as prime, composite, or neither.

ask:

- Ensure proper input validation.
- Optimize the logic for efficiency.
- Compare the output with earlier prompting strategies.



The screenshot shows a Visual Studio Code editor with a Python file named `ass22ai.py`. The code is a program that classifies a number as prime, composite, or neither. It prompts the user to enter a number and then checks its primality using a loop. The terminal output shows the program being run with inputs 5, 4, and 1, resulting in the correct classifications: prime, composite, and neither prime nor composite, respectively. The chat panel on the right shows a message from the AI assistant.

```
1 ...  
2 Python program that classifies a number as prime, composite, or neither.  
3 ...  
4  
5 num = int(input("Enter a number: "))  
6 if num > 1:  
7     is_prime = True  
8     for i in range(2, int(num*0.5) + 1):  
9         if num % i == 0:  
10            is_prime = False  
11            break  
12 if is_prime:  
13     print(f"The given number {num} is a prime number.")  
14 else:  
15     print(f"The given number {num} is a composite number.")  
16 elif num == 1:  
17     print("The given number 1 is neither prime nor composite.")  
18 else:  
19     print("Invalid input.")  
20  
21
```

Enter a number: 5
The given number 5 is a prime number.
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\shyam\ai assistes code\ass22ai.py"
Enter a number: 4
The given number 4 is a composite number.
PS C:\Users\shyam\ai assistes code> 1
1
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\shyam\ai assistes code\ass22ai.py"
Enter a number: 1
The given number 1 is neither prime nor composite.
PS C:\Users\shyam\ai assistes code> []

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

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

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a perfect number.

Task:

- Record the AI-generated code.
- Test the program with multiple inputs.
- Identify any missing conditions or inefficiencies in the logic.

```
1 def is_perfect_number(n):
2     """
3     Checks if a given number is a perfect number.
4     A perfect number is a positive integer that is equal to the sum of its proper positive divisors. print true or false
5     """
6     if n <= 0:
7         return False
8
9     sum_of_divisors = 0
10    for i in range(1, n):
11        if n % i == 0:
12            sum_of_divisors += i
13
14    return sum_of_divisors == n
15
16 num = int(input("Enter a number: "))
17 if is_perfect_number(num):
18     print("True")
19 else:
20     print("False")
```

Terminal Output:

```
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"
Enter a number: 12
False
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"
Enter a number: 6
True
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"
Enter a number: 0
False
PS C:\Users\shyam\ai assistes code> & C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"
Enter a number: 1
False
```

Missing Conditions and Inefficiencies

- No validation for negative numbers or zero.
- Loop runs up to n-1, which is inefficient for large values.
- Function does not handle non-integer inputs.

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

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python program that determines whether a given number is even or odd, including proper input validation.

Examples:

- Input: 8 → Output: Even
- Input: 15 → Output: Odd
- Input: 0 → Output: Even

Task:

- Analyze how examples improve input handling and output clarity.
- Test the program with negative numbers and non-integer inputs.

The screenshot displays the Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search icon. The Explorer sidebar on the left shows a project named 'AI ASSISTES CODE' with files 'ass2ai.py', 'ass21ai.py', 'ass22ai.py' (selected), and 'lab1ai.py'. The main editor window shows the content of 'ass22ai.py', which is a Python program for checking even/odd numbers with input validation. The code includes comments, input prompts, and conditional logic. The bottom panel contains the TERMINAL tab, showing the execution of the script using the command: `PS C:\Users\shyam\ai assistes code> & C:/Users/shyam/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"`. The terminal output shows three test cases: input 15 results in 'Odd', input 8 results in 'Even', and input 0 results in 'Even'.

```
File Edit Selection View Go Run Terminal Help
🔍 ai assistes code
```

EXPLORER

- AI ASSISTES CODE
 - ass2ai.py
 - ass21ai.py
 - ass22ai.py
 - lab1ai.py

```
ass22ai.py > ...
1  """
2      Python program that determines whether a given number is even or odd, including proper input validation.
3      Input: 8 → Output: Even
4      • Input: 15 → Output: Odd
5      • Input: 0 → Output: Even
6
7  """
8  num = int(input("Enter a number: "))
9  if num > 0:
10     if num % 2 == 0:
11         print(f"The given number {num} is Even.")
12     else:
13         print(f"The given number {num} is Odd.")
14 elif num < 0:
15     print("Invalid input.")
16 else:
17     print(f"The given number {num} is Even.")
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Python + - 🗑️ ... 🔄 ✕
PS C:\Users\shyam\ai assistes code> & C:/Users/shyam/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"
Enter a number: 15
The given number 15 is Odd.
PS C:\Users\shyam\ai assistes code> & C:/Users/shyam/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"
Enter a number: 8
The given number 8 is Even.
PS C:\Users\shyam\ai assistes code> & C:/Users/shyam/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/shyam/ai assistes code/ass22ai.py"
Enter a number: 0
The given number 0 is Even.
PS C:\Users\shyam\ai assistes code>
```

> OUTLINE
> TIMELINE

Analysis: Effect of Examples on Input Handling and Output Clarity

- Examples guide the program to return only “Even” or “Odd”.
- Including 0 → Even ensures correct handling of zero.
- Input validation is added to prevent non-integer inputs.