

ASSIGNMENT -3.1

BATCH-29

ROLL_NO:2303A51600

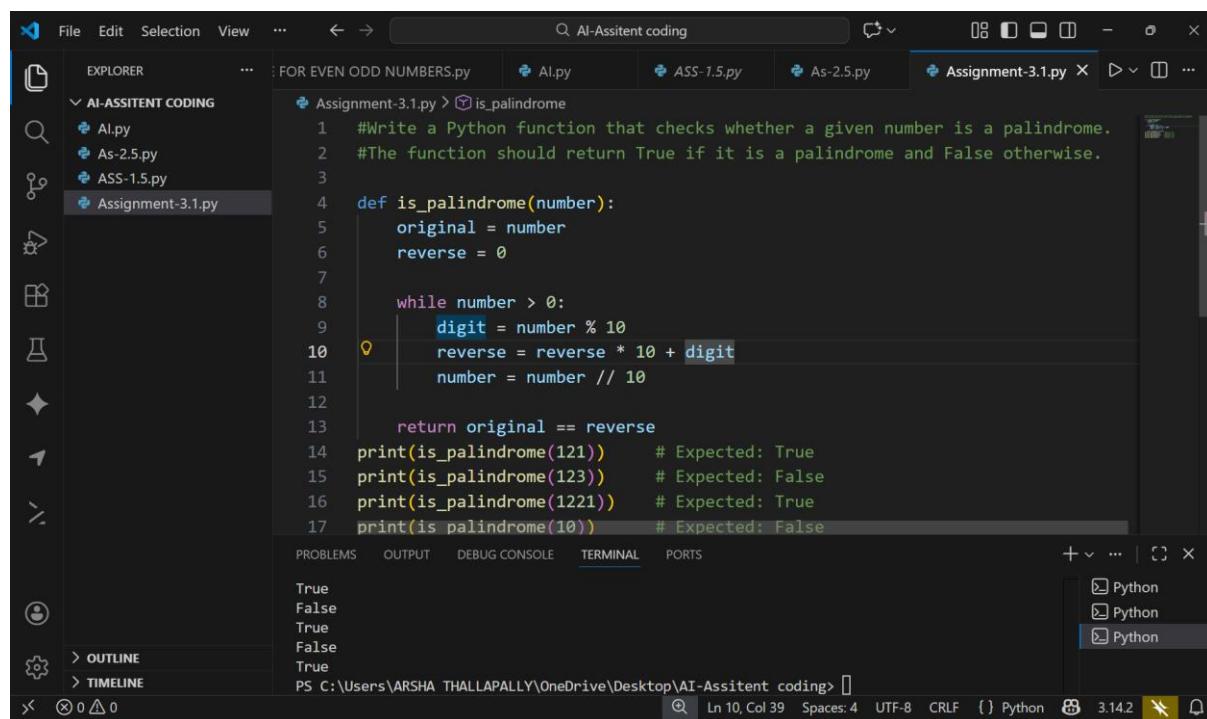
NAME-ARSHA VARDHINI

TASK-1: ZERO-SHOT PROMPTING (PALINDROME NUMBER PROGRAM)

PROMPT: Write a Python function that checks whether a given number is a palindrome.

The function should return True if it is a palindrome and False otherwise.

CODE:



A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows the 'EXPLORER' view with files: AI-assitant coding, AI-ASSITENT CODING, AI.py, As-2.5.py, ASS-1.5.py, and Assignment-3.1.py (which is currently selected). The main editor area displays a Python script named 'Assignment-3.1.py'. The code defines a function 'is_palindrome' that takes a 'number' as input and returns True if it is a palindrome. It uses a while loop to reverse the number by extracting digits and building a 'reverse' number. The code then prints the result of calling this function with several test cases. The bottom status bar shows the file path 'C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>' and other details like line and column numbers. A terminal window is also visible at the bottom.

```
FOR EVEN ODD NUMBERS.py AI.py ASS-1.5.py As-2.5.py Assignment-3.1.py X

Assignment-3.1.py > is_palindrome
1  #Write a Python function that checks whether a given number is a palindrome.
2  #The function should return True if it is a palindrome and False otherwise.
3
4  def is_palindrome(number):
5      original = number
6      reverse = 0
7
8      while number > 0:
9          digit = number % 10
10         reverse = reverse * 10 + digit
11         number = number // 10
12
13     return original == reverse
14 print(is_palindrome(121))    # Expected: True
15 print(is_palindrome(123))    # Expected: False
16 print(is_palindrome(1221))   # Expected: True
17 print(is_palindrome(10))     # Expected: False

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

True
False
True
False
True
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>
```

OBSERVATION:

- The model is given only the explanation of the question
- Any example or detailed explanation is not given
- Answer is accurate but not specific with negative and non-integers values

TASK-2: ONE-SHOT PROMPTING (FACTORIAL CALCULATION)

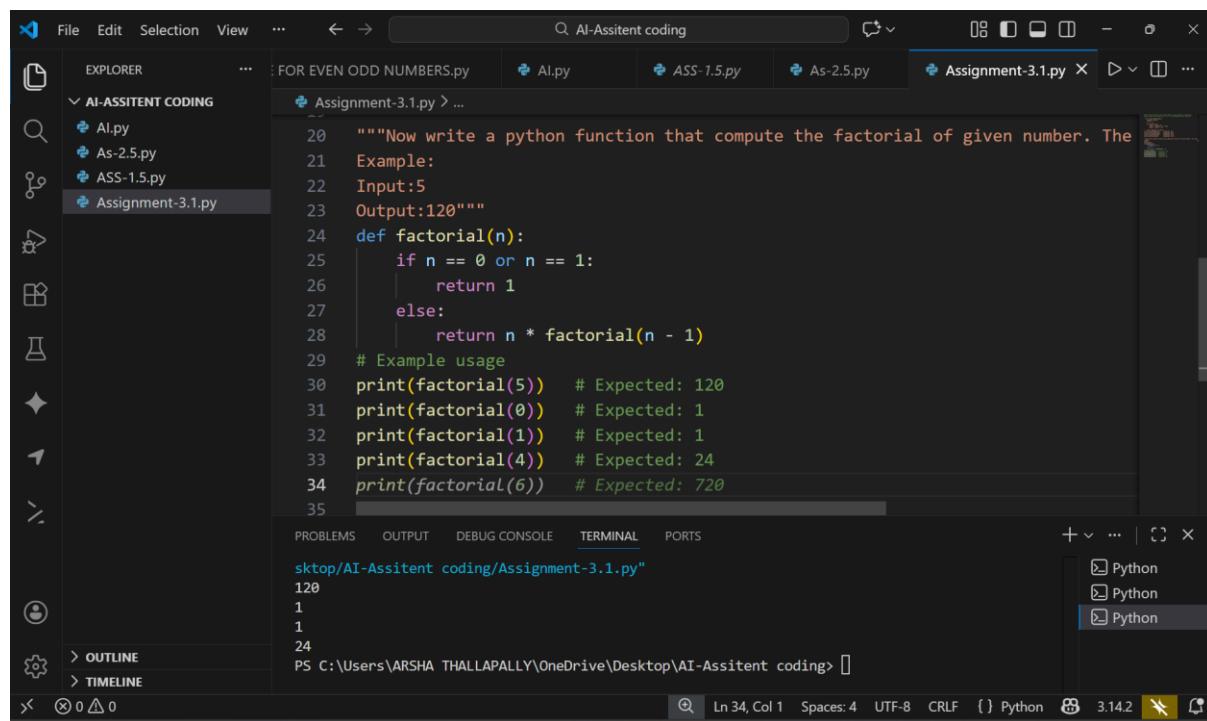
PROMPT: Now write a python function that compute the factorial of given number. The function should return the result.

Example:

Input:5

Output:120

CODE:



A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "AI-Assitant coding". The left sidebar shows the "EXPLORER" view with files: FOR EVEN ODD NUMBERS.py, AI.py, ASS-1.5.py, As-2.5.py, and Assignment-3.1.py. Assignment-3.1.py is selected. The main editor area contains the following Python code:

```
20  """Now write a python function that compute the factorial of given number. The
21 Example:
22 Input:5
23 Output:120"""
24 def factorial(n):
25     if n == 0 or n == 1:
26         return 1
27     else:
28         return n * factorial(n - 1)
29 # Example usage
30 print(factorial(5)) # Expected: 120
31 print(factorial(0)) # Expected: 1
32 print(factorial(1)) # Expected: 1
33 print(factorial(4)) # Expected: 24
34 print(factorial(6)) # Expected: 720
35
```

The bottom terminal window shows the output of running the code:

```
sktop/AI-Assitant coding/Assignment-3.1.py"
120
1
1
24
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>
```

OBSERVATION:

Clear understanding of the output

Better choice of logic-stack overflow, recursion complexity

Correct handling of base case

Improve code simplicity

TASK-3: FEW-SHOT PROMPTING (ARMSTRONG NUMBER CHECK)

Prompt: Example 1:

Input: 153

Output: Armstrong Number

Example 2:

Input: 370

Output: Armstrong Number

Example 3:

Input: 123

Output: Not an Armstrong Number

Now write a Python function that checks whether a given number is an Armstrong number.

The function should return an appropriate result.

CODE:

The screenshot shows a dark-themed instance of Visual Studio Code. The left sidebar has a tree view titled 'EXPLORER' under 'AI-ASSISTENT CODING' containing files: 'Assignment-3.1.py', 'AI.py', 'As-2.5.py', 'ASS-1.5.py'. The main editor area contains Python code for checking Armstrong numbers. The code includes examples for input 153 (Output: Armstrong Number), input 370 (Output: Armstrong Number), and input 123 (Output: Not an Armstrong Number). It then prompts the user to write a function to check if a given number is an Armstrong number. The code uses a function definition:

```
def is_armstrong(number):  
    total = 0  
    temp = number
```

The bottom status bar shows the file path 'C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assistent coding>' and other details like line 44, column 11, spaces: 4, and encoding: UTF-8.

OBSERVATION:

Clear output formatting. structured way

Correct logic selection

Easy understanding of code

Exact Appropriate answer

Optimized and customized solution

TASK-4: CONTEXT-MANAGED PROMPTING (OPTIMIZED NUMBER CLASSIFICATION)

PROMPT:

You are writing a Python program for number classification.

Requirements:

- Accept only integer input
- Handle invalid and negative inputs properly
- Classify the number as Prime, Composite, or Neither
- Optimize the logic for efficiency (avoid unnecessary checks)
- Return clear and user-friendly messages
- Write clean and readable Python code

Generate the program accordingly.

CODE:

The screenshot shows a dark-themed instance of Visual Studio Code. The left sidebar has a tree view titled 'EXPLORER' with a single expanded node 'AI-ASSIDENT CODING' containing files: 'AI.py', 'As-2.5.py', 'ASS-1.5.py', and 'Assignment-3.1.py'. The main editor area contains the following Python code:

```
FOR EVEN ODD NUMBERS.py AI.py ASS-1.5.py As-2.5.py Assignment-3.1.py
70 Requirements:
71 - Accept only integer input
72 - Handle invalid and negative inputs properly
73 - Classify the number as Prime, Composite, or Neither
74 - Optimize the logic for efficiency (avoid unnecessary checks)
75 - Return clear and user-friendly messages
76 - Write clean and readable Python code
77
78 Generate the program accordingly.
79 """
80     def classify_number(n):
81         if not isinstance(n, int):
82             return "Invalid input"
83
84         if n <= 1:
85             return "Neither Prime nor Composite"
86
87         for i in range(2, int(n ** 0.5) + 1):
88             if n % i == 0:
89                 return "Composite"
90
91         return "Prime"
```

The status bar at the bottom shows the file path 'PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assident coding>', line 'Ln 97, Col 1', and encoding 'UTF-8'. There are three terminal tabs labeled 'Python' in the bottom right corner.

OBSERVATION:

The role is defined

Constraints are clearly stated

Efficiency and validation of the code

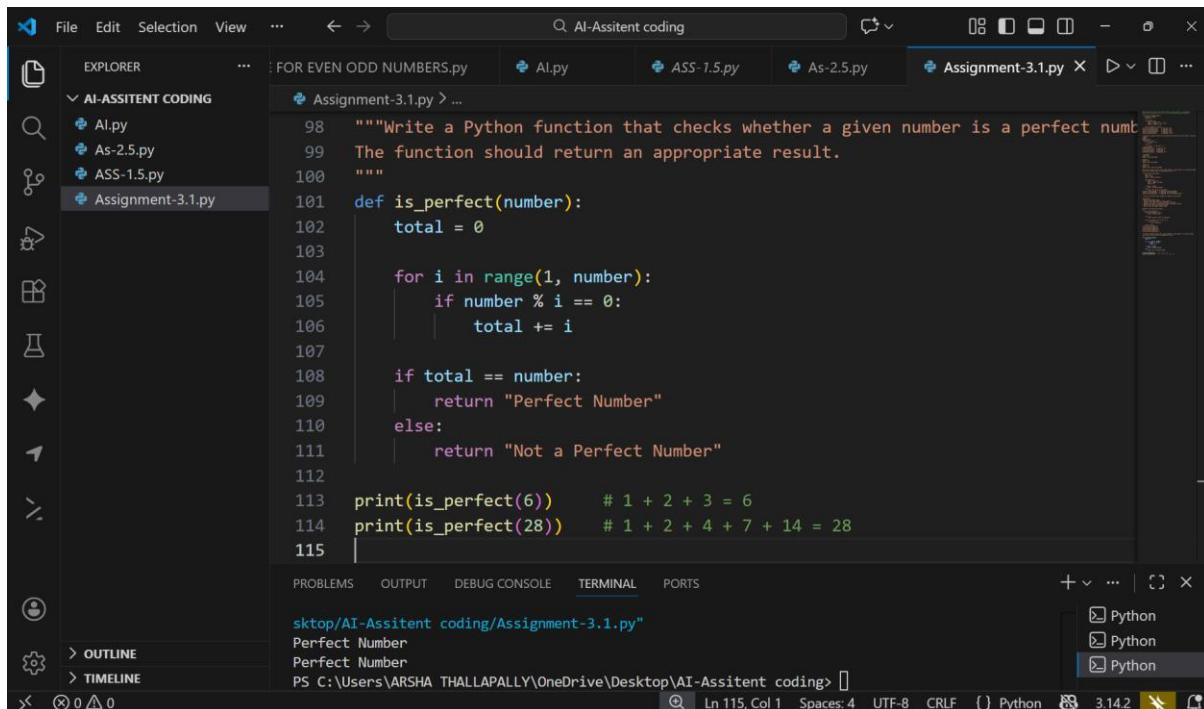
But the inputs should be specified more clearly mentioned

TASK-5: ZERO-SHOT PROMPTING (PERFECT NUMBER CHECK) VALIDATION)

PROMPT: Write a Python function that checks whether a given number is a perfect number.

The function should return an appropriate result.

CODE:



A screenshot of a code editor (VS Code) showing a Python file named Assignment-3.1.py. The code defines a function is_perfect that takes a number as input and returns "Perfect Number" if the sum of its divisors (excluding itself) equals the number, and "Not a Perfect Number" otherwise. The code also includes two print statements demonstrating the function with the numbers 6 and 28.

```
FOR EVEN ODD NUMBERS.py AI.py ASS-1.5.py As-2.5.py Assignment-3.1.py X ...  
EXPLORER AI-ASSISTENT CODING AI.py As-2.5.py ASS-1.5.py Assignment-3.1.py  
Assignment-3.1.py > ...  
98 """Write a Python function that checks whether a given number is a perfect number.  
99 The function should return an appropriate result.  
100 """  
101 def is_perfect(number):  
102     total = 0  
103  
104     for i in range(1, number):  
105         if number % i == 0:  
106             total += i  
107  
108     if total == number:  
109         return "Perfect Number"  
110     else:  
111         return "Not a Perfect Number"  
112  
113 print(is_perfect(6))      # 1 + 2 + 3 = 6  
114 print(is_perfect(28))    # 1 + 2 + 4 + 7 + 14 = 28  
115 |  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
sktop/AI-Assistent coding/Assignment-3.1.py  
Perfect Number  
Perfect Number  
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assistent coding>  
Ln 115, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.2 ✘ 🔍
```

OBSERVATION:

No input validation – if negative or float any..

Inefficient for large input

Did not specify input constraints

No edge case handling seen

TASK-6: FEW-SHOT PROMPTING (EVEN OR ODD CLASSIFICATION WITH VALIDATION)

PROMPT:

Example 1:

Input: 8

Output: Even

Example 2:

Input: 15

Output: Odd

Example 3:

Input: 0

Output: Even

Now write a Python program that determines whether a given number is Even or Odd.

The program should include proper input validation and return clear messages.

CODE:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar reads "AI-Assitant coding". The left sidebar has a tree view under "EXPLORER" titled "AI-ASSITENT CODING" with files "Assignment-3.1.py", "Ai.py", "As-2.5.py", and "ASS-1.5.py". The main editor area displays a Python script named "Assignment-3.1.py". The code defines a function `check_even_odd` that takes a value, checks if it's an integer, and then determines if it's even or odd using the modulo operator. It prints the results for 8, 15, and 0. The terminal at the bottom shows the output: "Even", "Odd", and "Even". The status bar at the bottom right indicates the file is "Assignment-3.1.py", line 144, column 1, with 4 spaces, encoding UTF-8, and Python 3.14.2.

```
127 Input: 0
128 Output: Even
129
130 Now write a Python program that determines whether a given number is Even or Odd.
131 The program should include proper input validation and return clear messages.
132 """
133 def check_even_odd(value):
134     if not isinstance(value, int):
135         return "Invalid input"
136
137     if value % 2 == 0:
138         return "Even"
139     else:
140         return "Odd"
141 print(check_even_odd(8))
142 print(check_even_odd(15))
143 print(check_even_odd(0))
```

OBSERVATION:

Negative integer are handled correctly

Program safely rejected non integer inputs

Improve input handling

Clear and consistent output