

ASSIGNMENT -3.1

BATCH-29

ROLL_NO:2303A51600

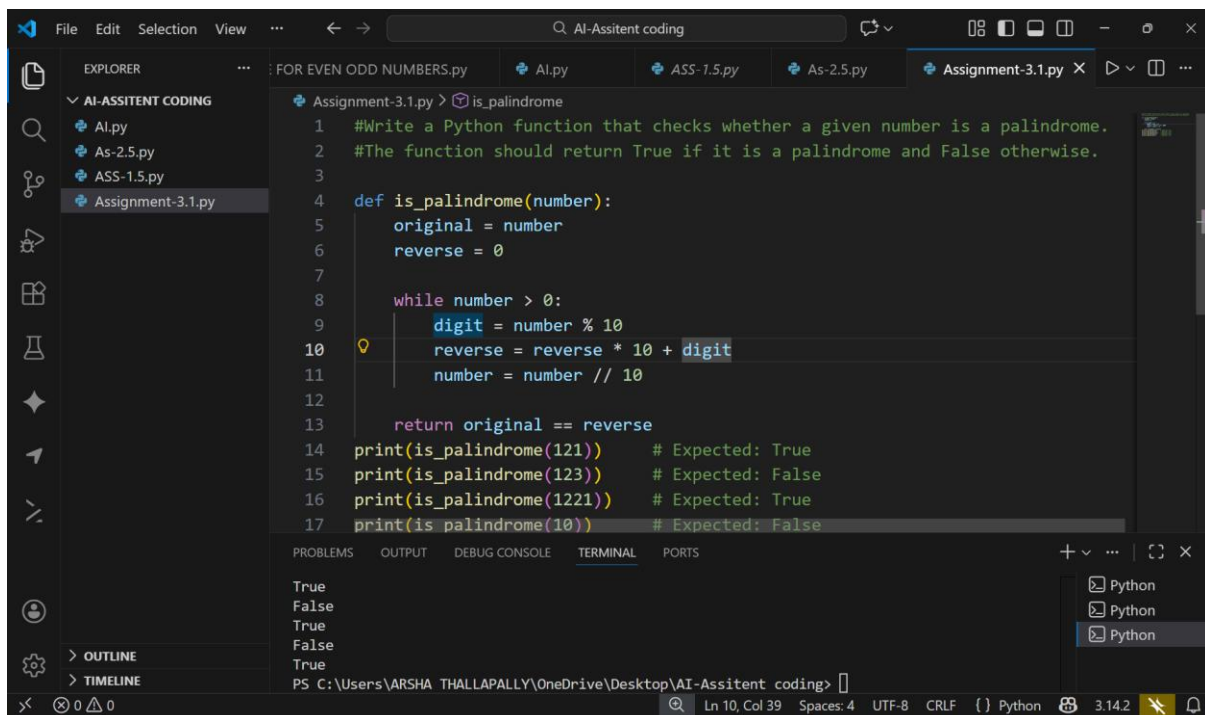
NAME-ARSHA VARDHINI

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:



The screenshot shows a Visual Studio Code editor window with a Python file named 'Assignment-3.1.py'. The code defines a function 'is_palindrome' that checks if a number is a palindrome by reversing its digits. The function returns True if the original number equals the reversed number, and False otherwise. The code includes test cases for 121, 123, 1221, and 10. The terminal output shows the results of these tests: True, False, True, and False respectively. The status bar at the bottom indicates the file is saved, the cursor is at line 10, column 39, and the file is encoded in UTF-8 with CRLF line endings.

```
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
15 print(is_palindrome(121))    # Expected: True
16 print(is_palindrome(123))    # Expected: False
17 print(is_palindrome(1221))   # Expected: True
18 print(is_palindrome(10))     # Expected: False
```

True
False
True
False

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent 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)

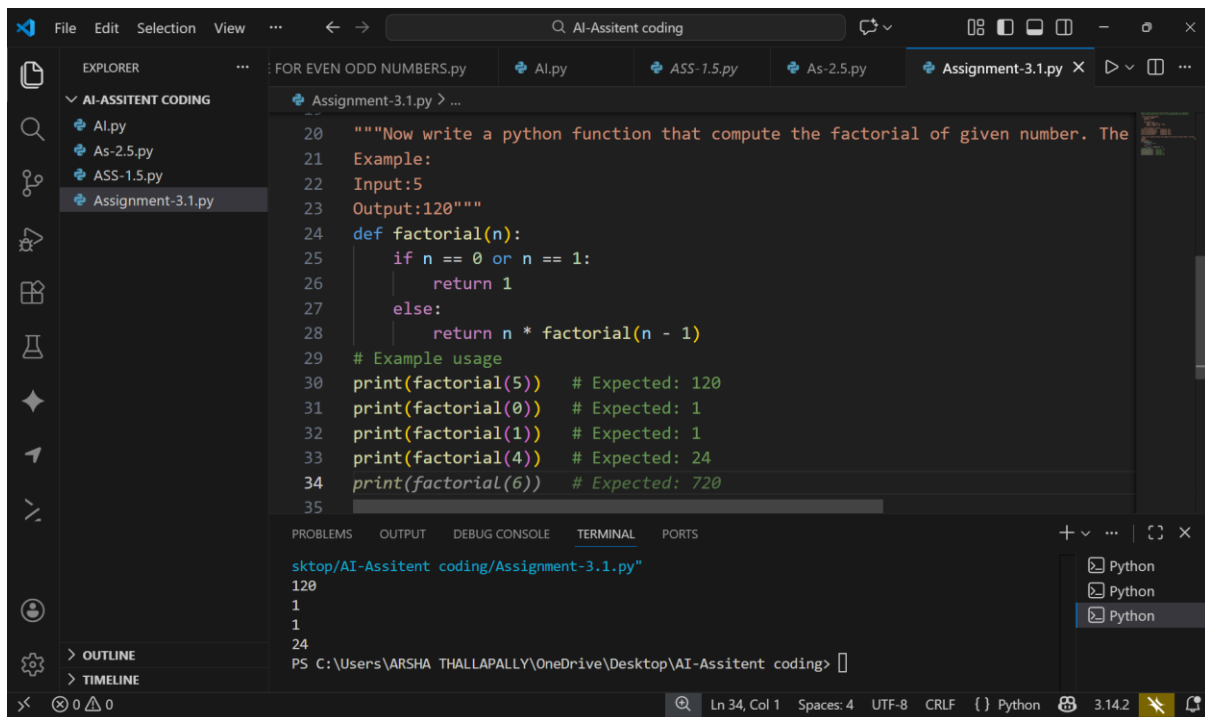
PROMPT: Now write a python function that compute the factorial of given number. The function should return the result.

Example:

Input:5

Output:120

CODE:



```
File Edit Selection View ... AI-Assitent coding Assignment-3.1.py X
EXPLORER AI-ASSITENT CODING Assignment-3.1.py > ...
  AI.py
  As-2.5.py
  ASS-1.5.py
  Assignment-3.1.py
FOR EVEN ODD NUMBERS.py AI.py ASS-1.5.py As-2.5.py Assignment-3.1.py X
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
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
skttop/AI-Assitent coding/Assignment-3.1.py"
120
1
1
24
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent 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:

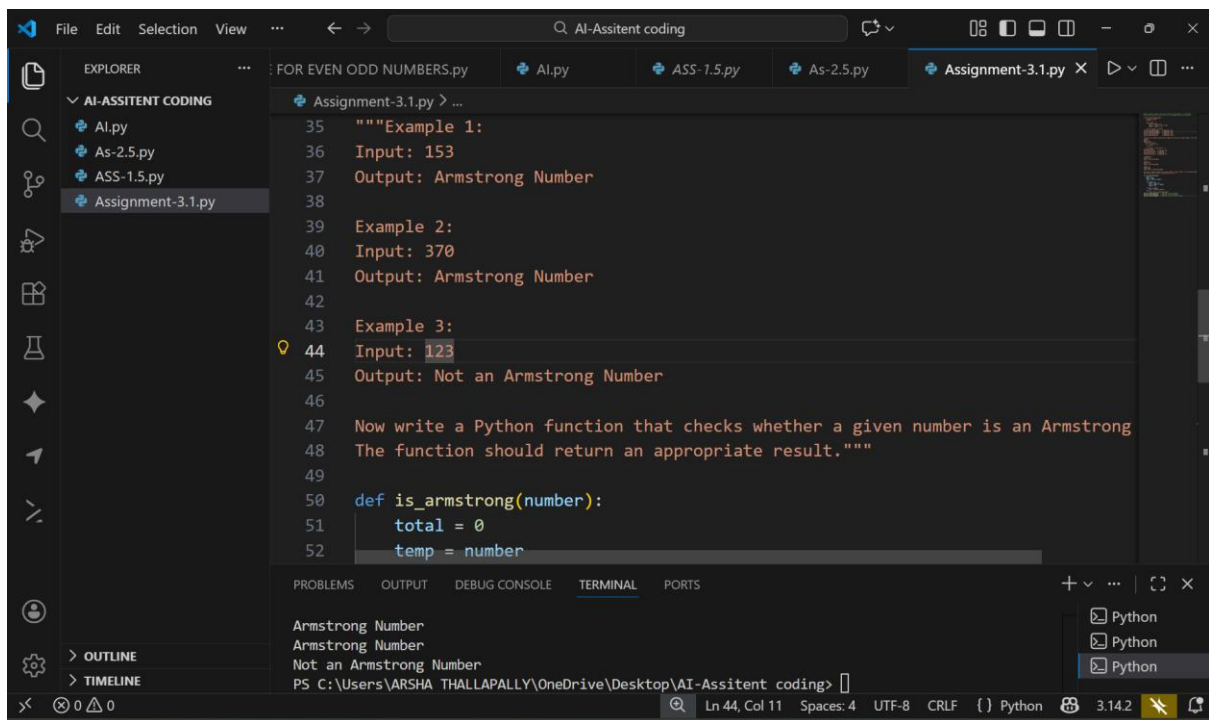
Input: 123

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:



```
35 """Example 1:
36 Input: 153
37 Output: Armstrong Number
38
39 Example 2:
40 Input: 370
41 Output: Armstrong Number
42
43 Example 3:
44 Input: 123
45 Output: Not an Armstrong Number
46
47 Now write a Python function that checks whether a given number is an Armstrong
48 The function should return an appropriate result."""
49
50 def is_armstrong(number):
51     total = 0
52     temp = number
```

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:

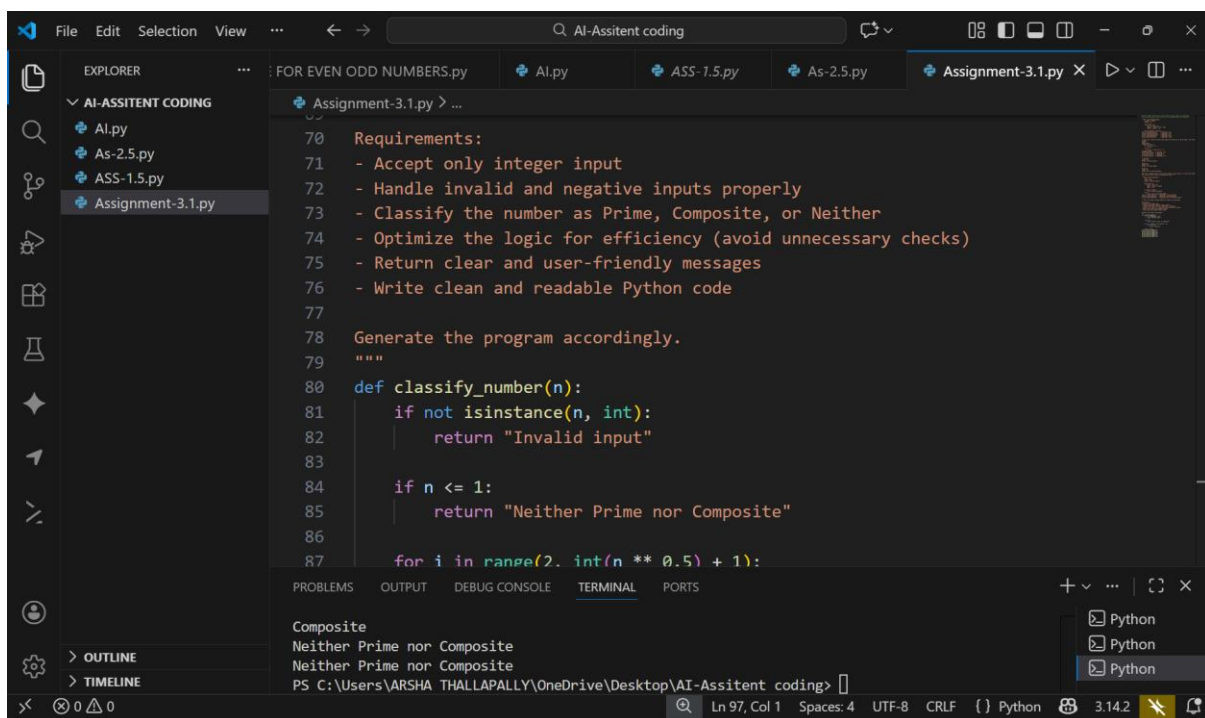
You are writing a Python program for number classification.

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:



```
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):
```

The screenshot shows a VS Code editor with a file named 'Assignment-3.1.py'. The code defines a function 'classify_number' that checks for integer input, handles negative numbers, and uses an optimized loop to check for primality. The bottom panel shows the 'TERMINAL' output with the command 'PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>' and the output 'Composite'.

OBSERVATION:

The role is defined

Constraints are clearly stated

Efficiency and validation of the code

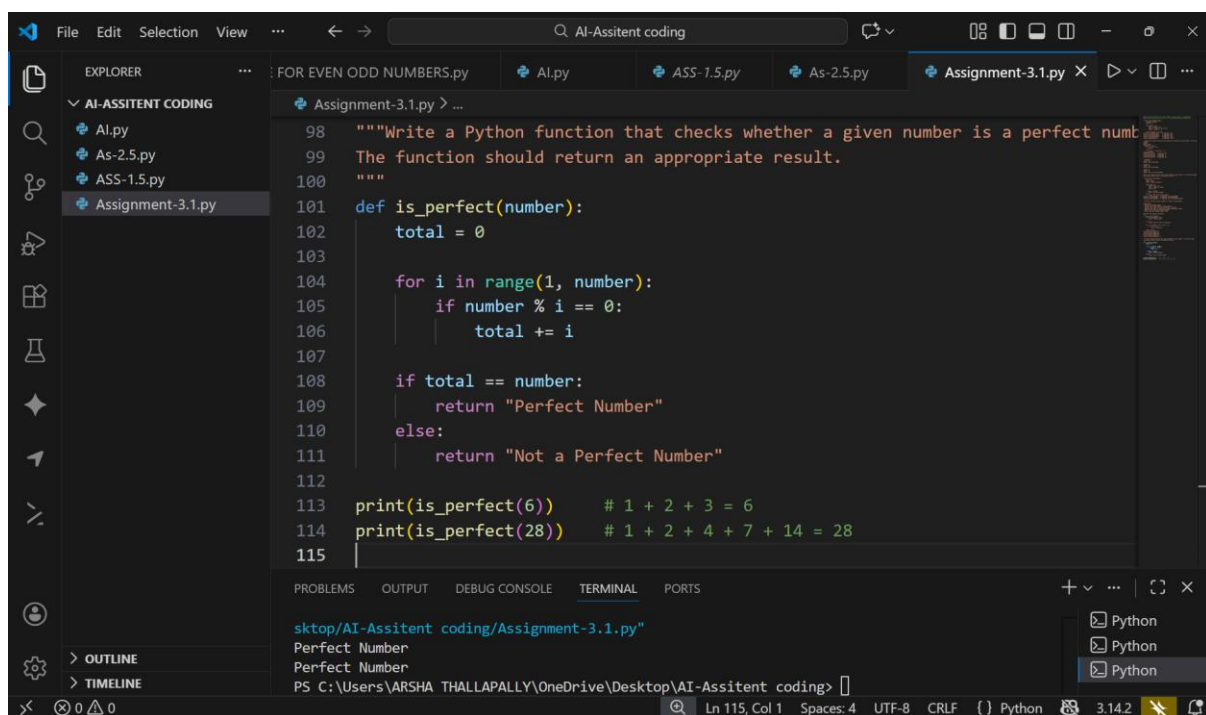
But the inputs should be specified more clearly mentioned

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 Visual Studio Code editor window. The Explorer sidebar on the left shows a project named 'AI-ASSITENT CODING' with files 'AI.py', 'As-2.5.py', 'ASS-1.5.py', and 'Assignment-3.1.py'. The main editor area displays the code for 'Assignment-3.1.py'. The code includes a docstring, a function definition 'def is_perfect(number):', and test calls. The docstring says: 'Write a Python function that checks whether a given number is a perfect number. The function should return an appropriate result.' The function 'is_perfect' initializes 'total = 0', loops from 1 to 'number' (exclusive), and adds 'i' to 'total' if 'number % i == 0'. It then returns 'Perfect Number' if 'total == number', otherwise 'Not a Perfect Number'. Test calls are 'print(is_perfect(6))' (commented as '# 1 + 2 + 3 = 6') and 'print(is_perfect(28))' (commented as '# 1 + 2 + 4 + 7 + 14 = 28'). The bottom status bar shows 'Ln 115, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python' interpreter. The output panel at the bottom shows the execution results: 'sktop/AI-Assitent coding/Assignment-3.1.py', 'Perfect Number', and 'Perfect Number'.

OBSERVATION:

No input validation – if negative or float any..

Inefficient for large input

Did not specify input constraints

No edge case handing 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:

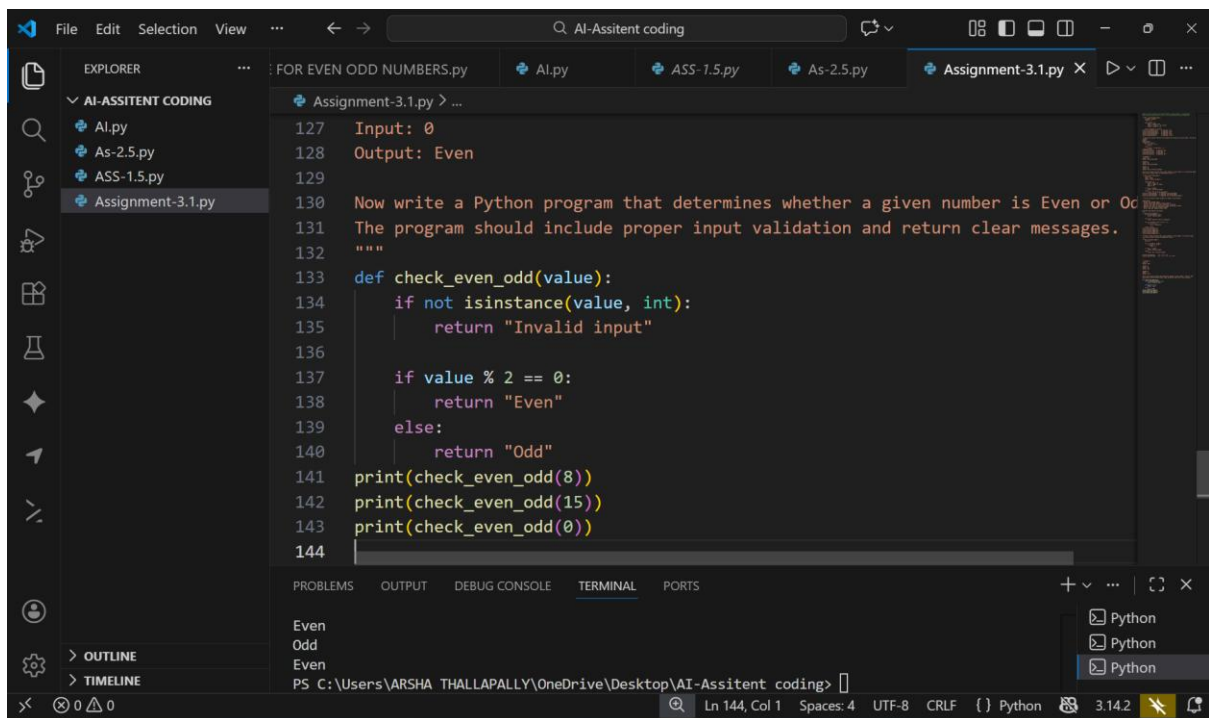
Input: 0

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:



```
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))
144
```

The screenshot shows a VS Code editor with a Python file named 'Assignment-3.1.py'. The code defines a function 'check_even_odd' that takes a value and returns 'Even', 'Odd', or 'Invalid input' based on whether it's an integer and its parity. The function is called with 8, 15, and 0. The terminal output shows 'Even', 'Odd', and 'Even' for these inputs respectively. The Explorer sidebar shows a folder 'AI-ASSITENT CODING' with files 'AI.py', 'As-2.5.py', 'ASS-1.5.py', and 'Assignment-3.1.py'. The bottom status bar shows 'Ln 144, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and '3.14.2'.

OBSERVATION:

Negative integer are handled correctly

Program safely rejected non integer inputs

Improve input handling

Clear and consistent output