

# ASSIGNMENT -3.1

## BATCH-30

ROLL-NO:2306A91001

NAME- V.Architha

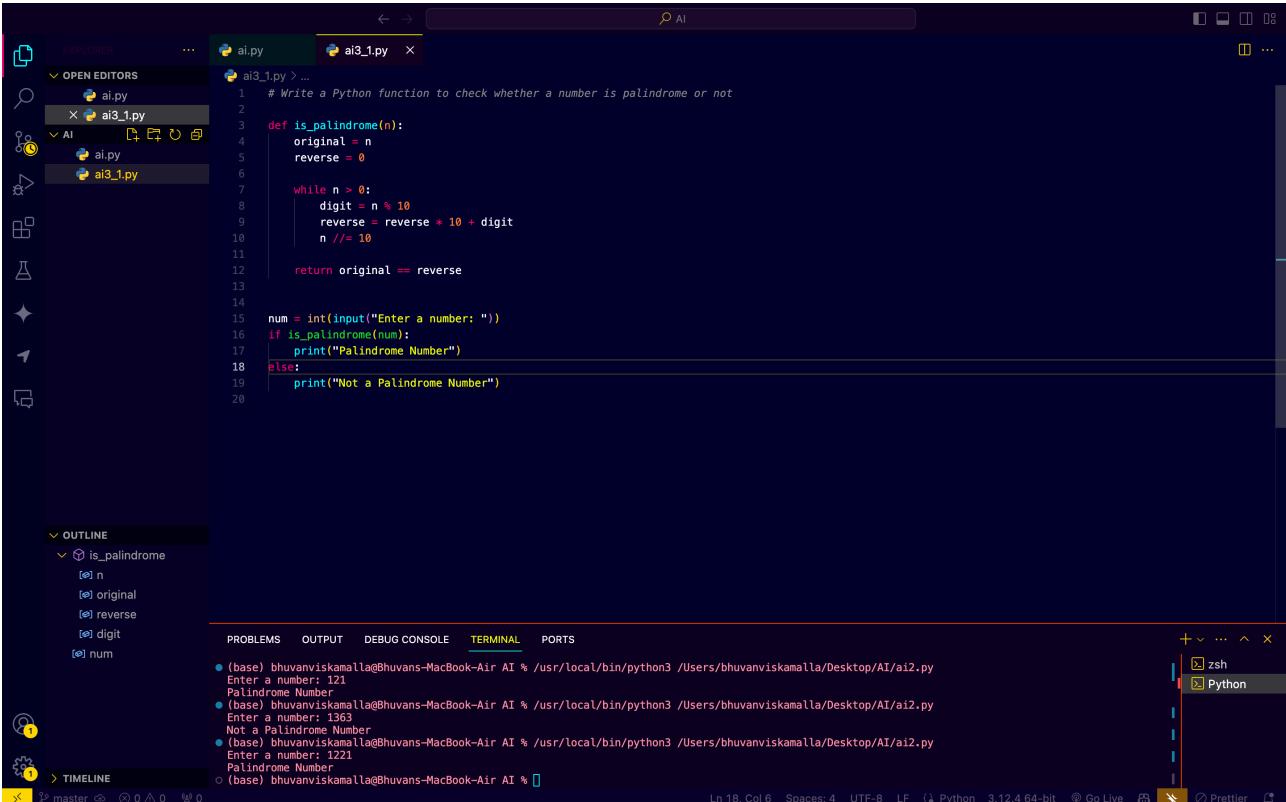
### TASK-1:

#### 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 the VS Code interface with the following details:

- EXPLORER:** Shows files: ai.py, ai3\_1.py, and ai3\_1.py (selected).
- EDITOR:** Displays the following Python code:

```
1  # Write a Python function to check whether a number is palindrome or not
2
3  def is_palindrome(n):
4      original = n
5      reverse = 0
6
7      while n > 0:
8          digit = n % 10
9          reverse = reverse * 10 + digit
10         n //= 10
11
12     return original == reverse
13
14
15 num = int(input("Enter a number: "))
16 if is_palindrome(num):
17     print("Palindrome Number")
18 else:
19     print("Not a Palindrome Number")
20
```

- OUTLINE:** Shows the structure of the code, including the function definition and its parameters.
- TERMINAL:** Shows the command line interface with the following session:

```
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai2.py
Enter a number: 121
Palindrome Number
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai2.py
Enter a number: 1363
Not a Palindrome Number
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai2.py
Enter a number: 1221
Palindrome Number
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI %
```

- STATUS BAR:** Shows the current file is ai3\_1.py, line 18, column 6, and other system information.

## 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)

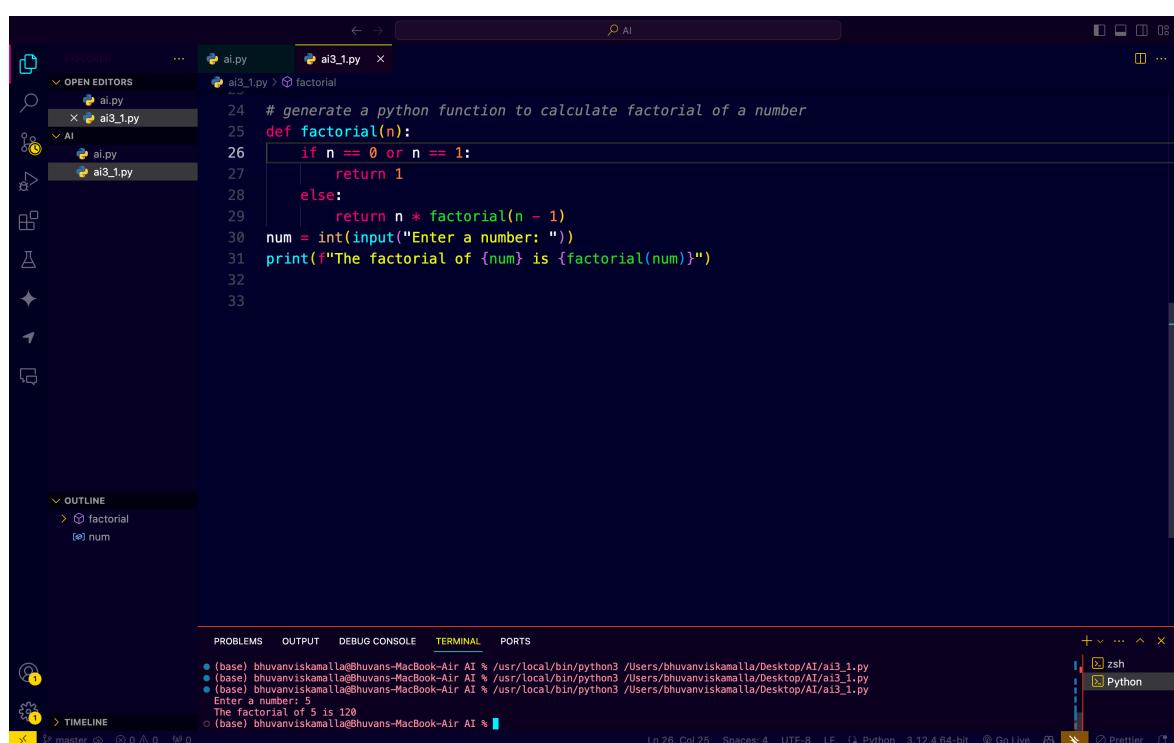
PROMPT: write a python function that compute the factorial of given number. The function should return the result.

Example:

Input:5

Output:120

## CODE:



The screenshot shows a code editor interface with a dark theme. The left sidebar includes an 'EXPLORERS' section with 'OPEN EDITORS' containing 'ai.py' and 'ai3\_1.py', and an 'AI' section with 'ai3\_1.py > factorial'. The main editor area displays the following Python code:

```
24 # generate a python function to calculate factorial of a number
25 def factorial(n):
26     if n == 0 or n == 1:
27         return 1
28     else:
29         return n * factorial(n - 1)
30 num = int(input("Enter a number: "))
31 print(f"The factorial of {num} is {factorial(num)}")
```

The terminal at the bottom shows the output of running the script:

```
(base) bhuvanviskamalla@huvans-MacBook-Air: ~ /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai3_1.py
Enter a number:
The factorial of 5 is 120
```

## 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:

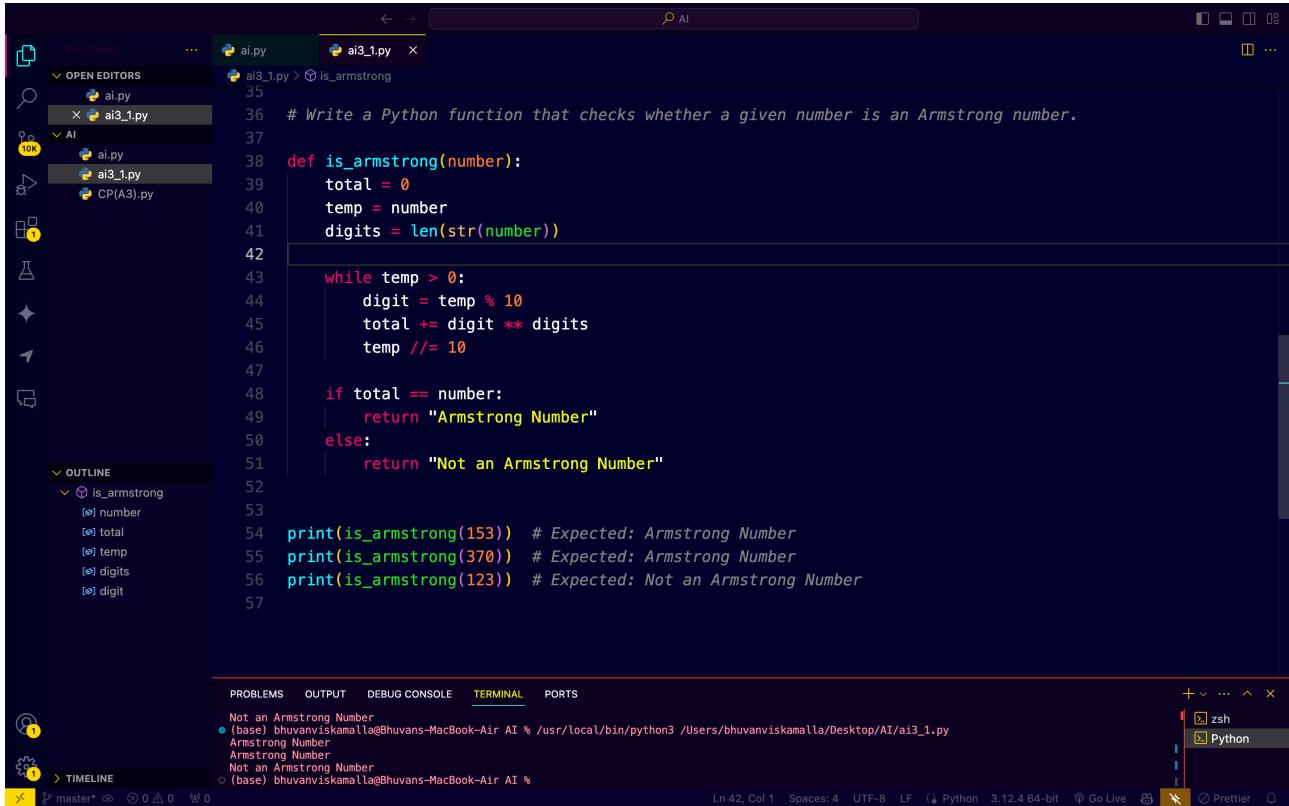
Input: 370

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 the VS Code interface with the following details:

- EXPLORER:** Shows files: ai.py, ai3\_1.py, and AI.
- EDITOR:** The active file is ai3\_1.py, containing the following code:

```
# Write a Python function that checks whether a given number is an Armstrong number.
36 def is_armstrong(number):
37     total = 0
38     temp = number
39     digits = len(str(number))
40
41     while temp > 0:
42         digit = temp % 10
43         total += digit ** digits
44         temp //= 10
45
46     if total == number:
47         return "Armstrong Number"
48     else:
49         return "Not an Armstrong Number"
50
51
52
53
54 print(is_armstrong(153)) # Expected: Armstrong Number
55 print(is_armstrong(370)) # Expected: Armstrong Number
56 print(is_armstrong(123)) # Expected: Not an Armstrong Number
```

- OUTLINE:** Shows the structure of the code, including the function and its variables: number, total, temp, digits, and digit.
- TERMINAL:** Shows the output of the code execution:

```
Not an Armstrong Number
● (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai3_1.py
Armstrong Number
Armstrong Number
Not an Armstrong Number
○ (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

- STATUS BAR:** Shows the current line (Ln 42, Col 1), spaces (Spaces: 4), encoding (UTF-8), line separator (LF), Python version (3.12.4 64-bit), Go Live, and Prettier.

## OBSERVATION:

Clear output formatting. structured way Correct logic selection Easy understanding of code Exact Appropriate answer Optimised and customised solution

## TASK-4:

### CONTEXT-MANAGED PROMPTING (OPTIMISED NUMBER CLASSIFICATION)

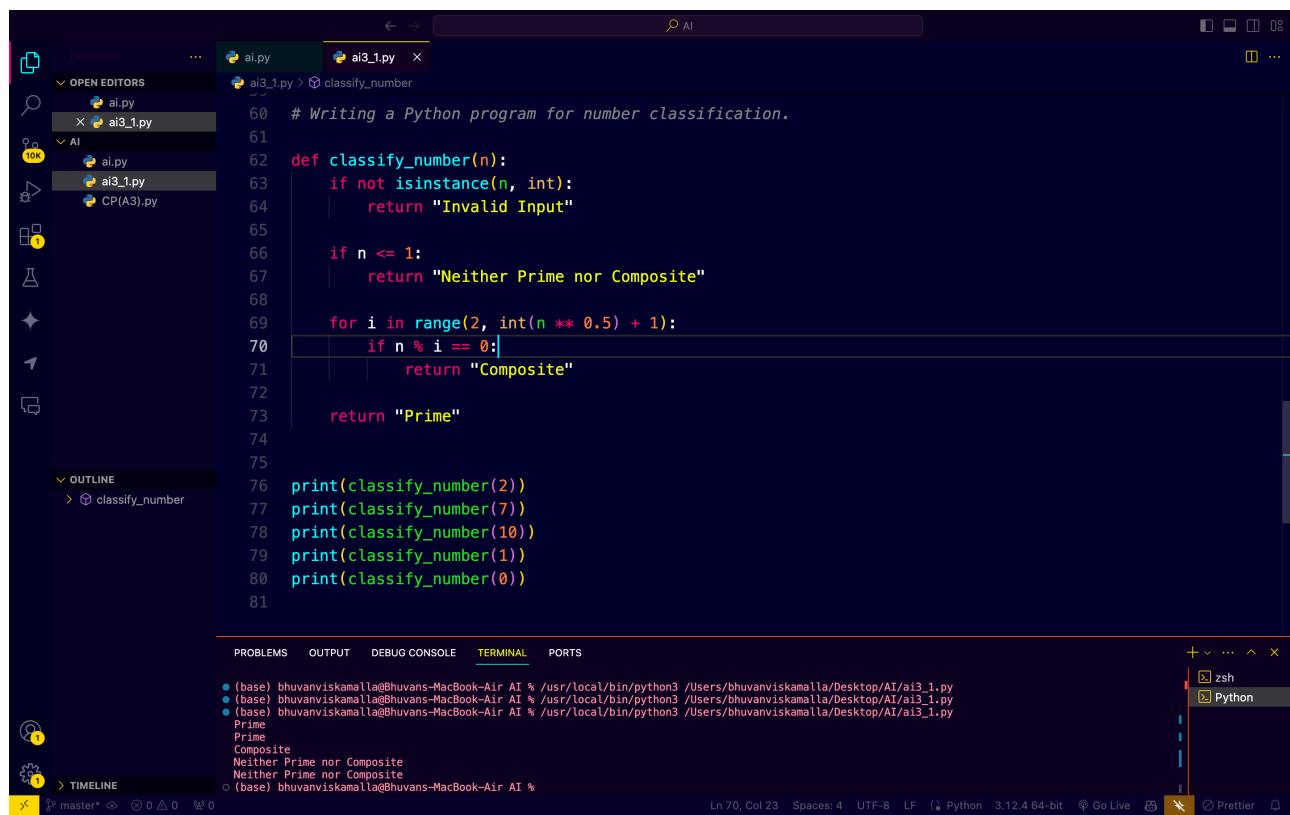
#### PROMPT:

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:



A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows the 'EXPLORER' view with files 'ai.py' and 'ai3\_1.py' listed. The main 'CODE' editor window displays the following Python code:

```
60  # Writing a Python program for number classification.
61
62  def classify_number(n):
63      if not isinstance(n, int):
64          return "Invalid Input"
65
66      if n <= 1:
67          return "Neither Prime nor Composite"
68
69      for i in range(2, int(n ** 0.5) + 1):
70          if n % i == 0:
71              return "Composite"
72
73      return "Prime"
74
75
76  print(classify_number(2))
77  print(classify_number(7))
78  print(classify_number(10))
79  print(classify_number(1))
80  print(classify_number(0))
81
```

The 'TERMINAL' tab at the bottom shows the output of running the code in a terminal:

```
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai3_1.py
Prime
Prime
Composite
Neither Prime nor Composite
Neither Prime nor Composite
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI %
```

OBSERVATION:

The role is defined Constraints are clearly stated Efficiency and validation of the code but the inputs should be specified more clearly mentioned

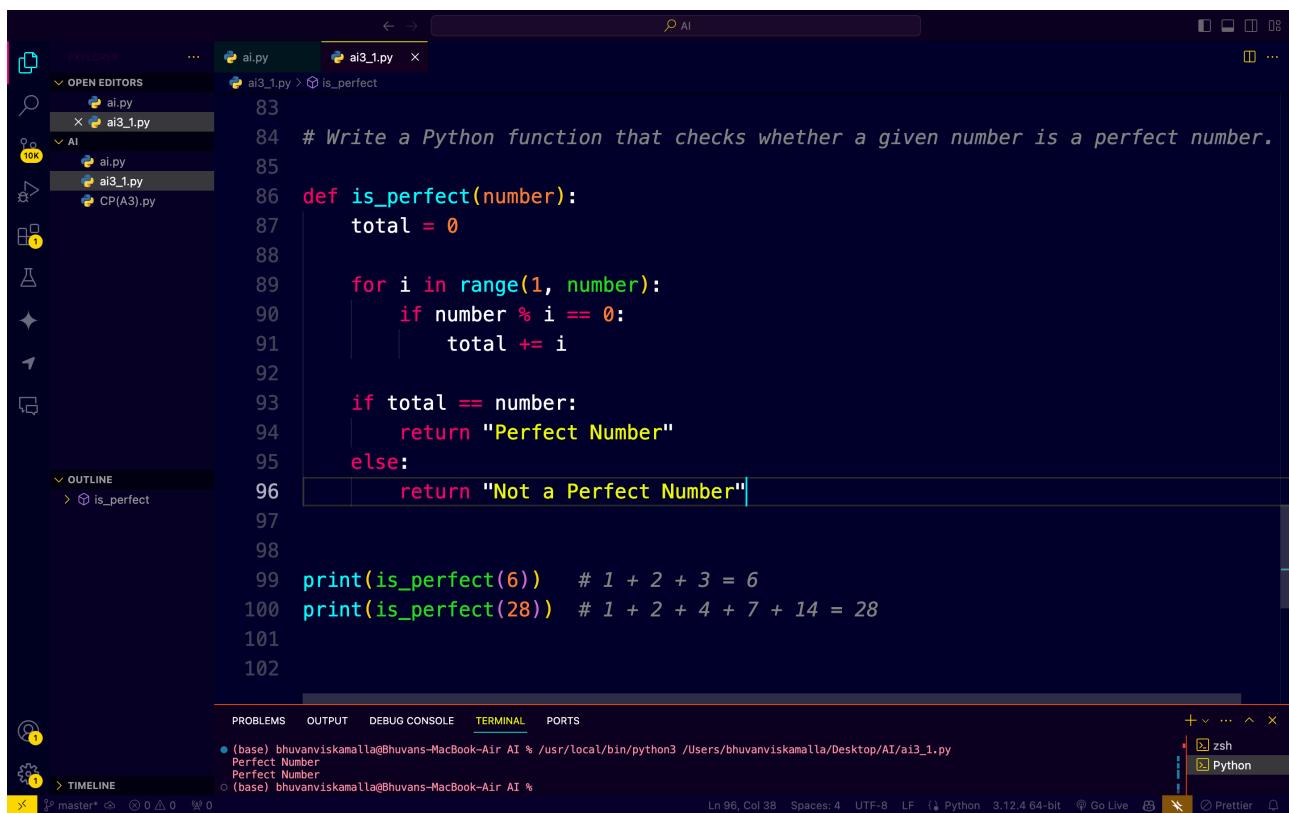
## TASK-5:

# 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:



The screenshot shows a dark-themed code editor with a sidebar on the left containing icons for file operations, a search bar, and a list of open files: 'ai.py', 'ai3\_1.py', 'ai3\_1.py > is\_perfect', 'ai.py', 'ai3\_1.py', and 'CP(A3).py'. The main area displays a Python script with the following code:

```
83
84 # Write a Python function that checks whether a given number is a perfect number.
85
86 def is_perfect(number):
87     total = 0
88
89     for i in range(1, number):
90         if number % i == 0:
91             total += i
92
93     if total == number:
94         return "Perfect Number"
95     else:
96         return "Not a Perfect Number"
97
98
99 print(is_perfect(6))    # 1 + 2 + 3 = 6
100 print(is_perfect(28))   # 1 + 2 + 4 + 7 + 14 = 28
101
102
```

The bottom status bar shows the terminal output: '(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai3\_1.py'. The status bar also includes icons for file operations, a timeline, and various developer tools.

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

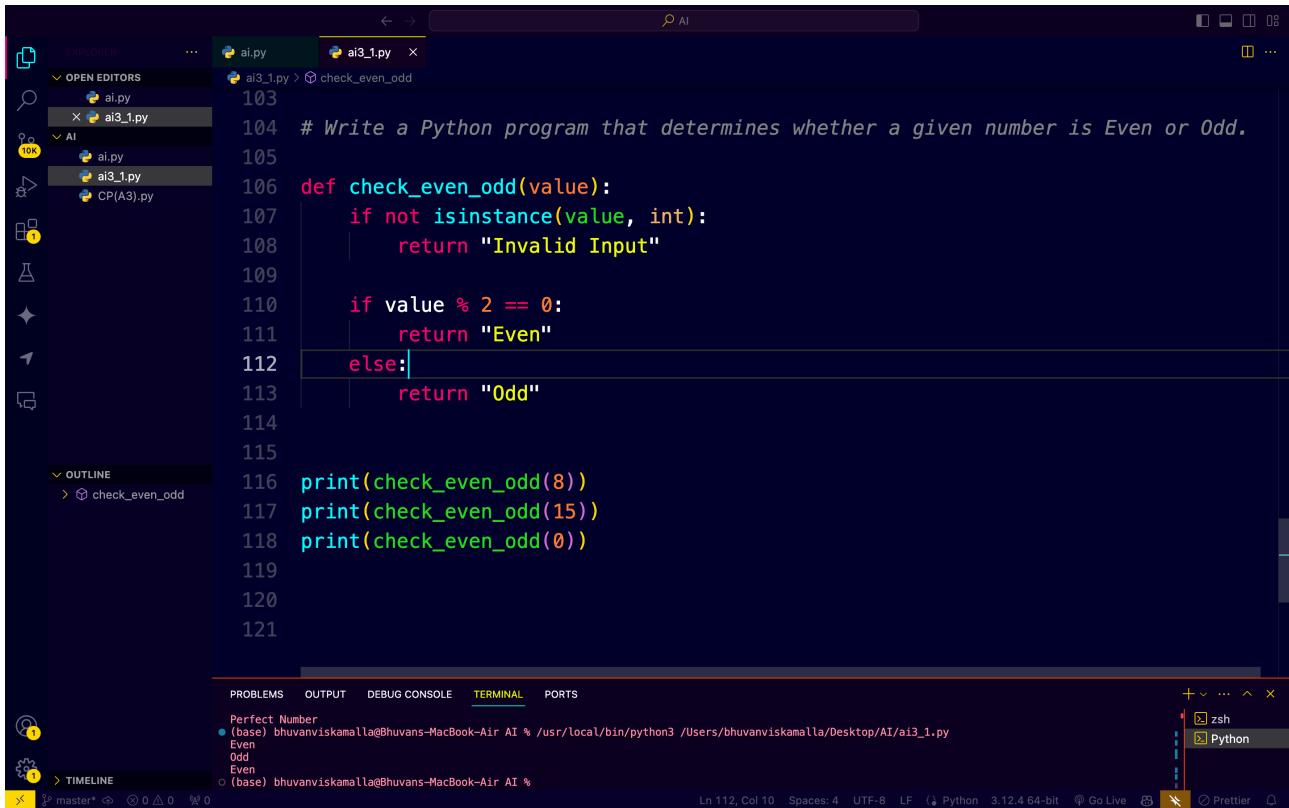
Output: Odd

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:



A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows the Explorer, Open Editors, and Outline panes. The main editor pane displays a Python script named 'ai3\_1.py' with the following code:

```
103 # Write a Python program that determines whether a given number is Even or Odd.
104
105 def check_even_odd(value):
106     if not isinstance(value, int):
107         return "Invalid Input"
108
109     if value % 2 == 0:
110         return "Even"
111     else:
112         return "Odd"
113
114
115
116 print(check_even_odd(8))
117 print(check_even_odd(15))
118 print(check_even_odd(0))
119
120
121
```

The terminal tab at the bottom shows the output of running the script:

```
Perfect Number
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai3_1.py
Even
Odd
Even
```

The status bar at the bottom indicates the file is 112 lines long, has 4 spaces per tab, is in UTF-8 encoding, and is a 3.12.4 64-bit Python file.

## OBSERVATION:

Negative integer are handled correctly Program safely rejected non integer inputs

Improve input handling Clear and consistent output