

## ASSIGNMENT -3.1

BATCH-30

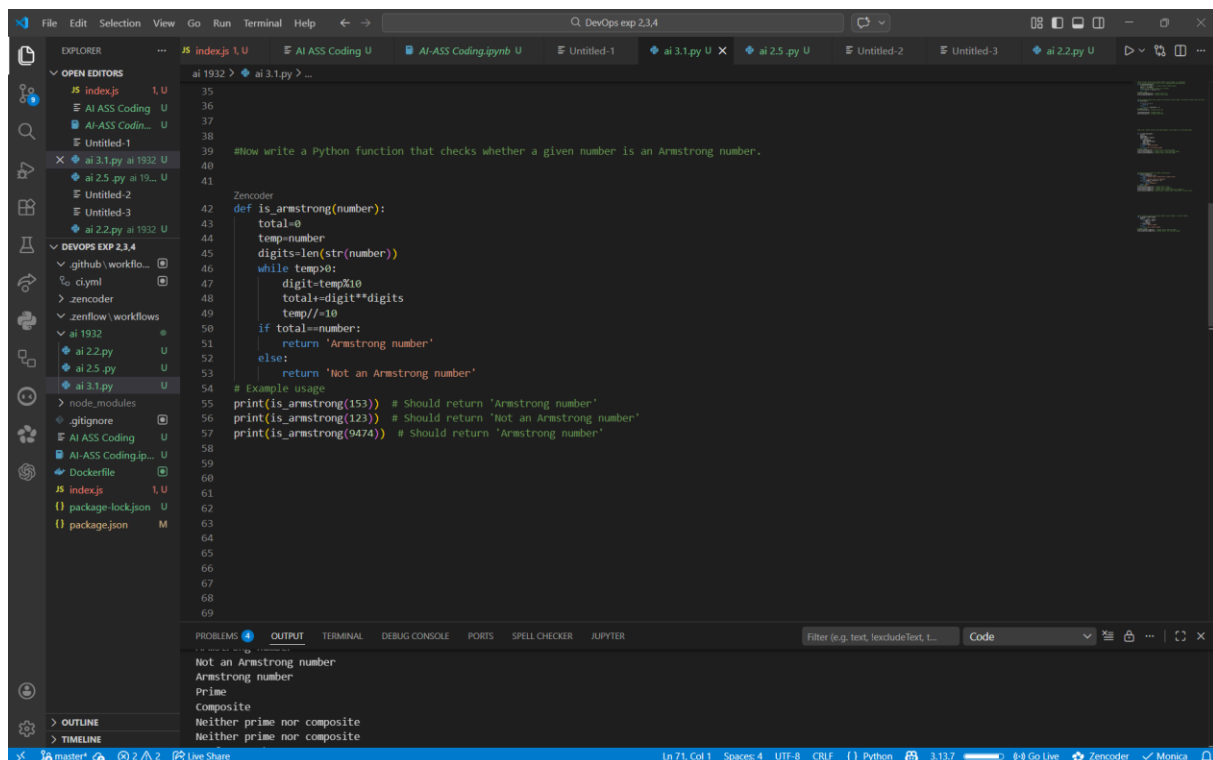
ROLL-NO:2303A51932

NAME-SRAVANI 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 image shows a screenshot of a Visual Studio Code editor window. The Explorer sidebar on the left shows a project structure with folders like 'ai 1932' and 'ai 2.2.py'. The main editor area displays a Python file named 'ai 3.1.py' containing a function 'def is\_armstrong(number):'. The function calculates the sum of the cubes of the digits of the number and compares it to the original number. Below the function, there are example usage lines: 'print(is\_armstrong(153))', 'print(is\_armstrong(123))', and 'print(is\_armstrong(9474))'. The bottom panel shows the 'OUTPUT' tab with the results of these function calls: 'Not an Armstrong number', 'Armstrong number', 'Prime', 'Composite', 'Neither prime nor composite', and 'Neither prime nor composite'. The status bar at the bottom indicates the file is in 'Python' mode, line 71, column 1, with 4 spaces and UTF-8 encoding.

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 nonintegers values TASK-2:

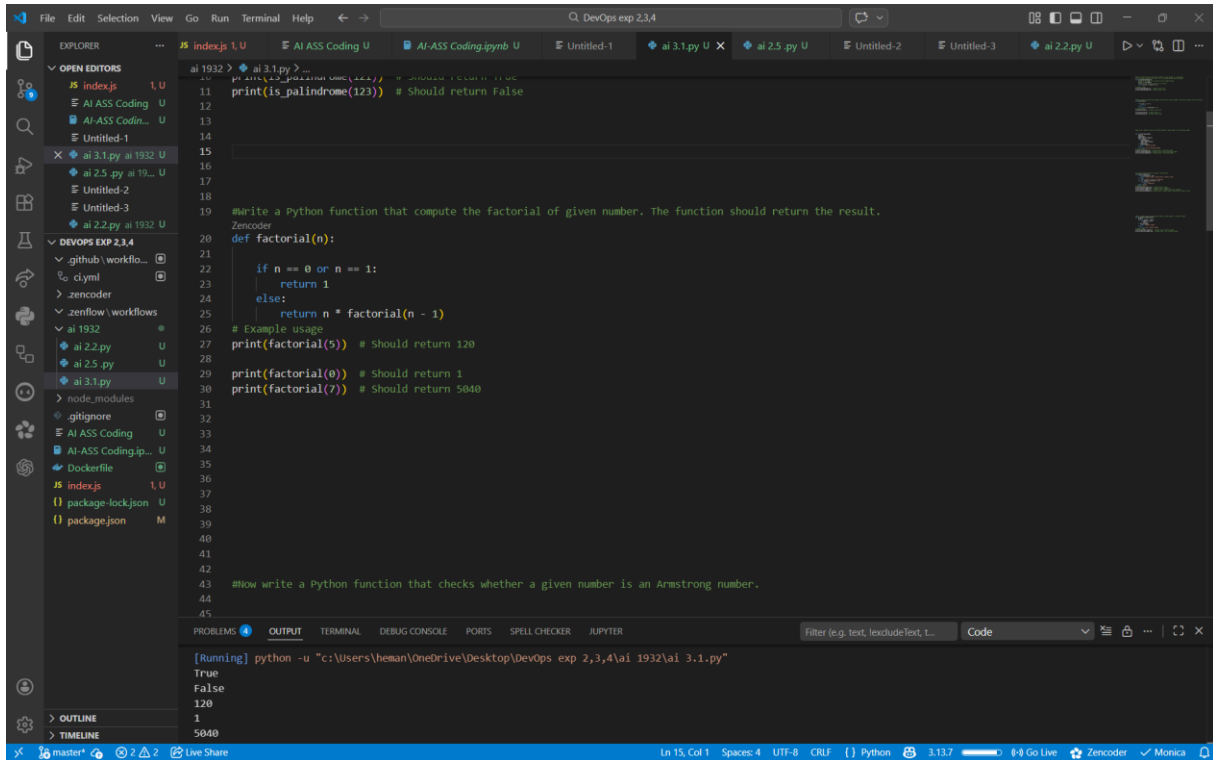
ONE-SHOT PROMPTING (FACTORIAL CALCULATION) PROMPT:

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

Example:

Input:5

Output:120 CODE:



```
11 print(is_palindrome(123)) # Should return False
12
13
14
15
16
17
18
19 #write a Python function that compute the factorial of given number. The function should return the result.
20 def factorial(n):
21     # Base case
22     if n == 0 or n == 1:
23         return 1
24     else:
25         return n * factorial(n - 1)
26 # Example usage
27 print(factorial(5)) # Should return 120
28
29 print(factorial(0)) # Should return 1
30 print(factorial(7)) # Should return 5040
31
32
33
34
35
36
37
38
39
40
41
42
43 #Now write a Python function that checks whether a given number is an Armstrong number.
44
45
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS SPELL CHECKER JUPYTER

[Running] python -u "c:\Users\heman\OneDrive\Desktop\DevOps exp 2,3,4\ai 1932\ai 3.1.py"

True  
False  
120  
1  
5040

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:

OBSERVATION:

Clear output formatting. structured way Correct logic selection Easy understanding of code Exact Appropriate answer Optimized and customized solution TASK-4:

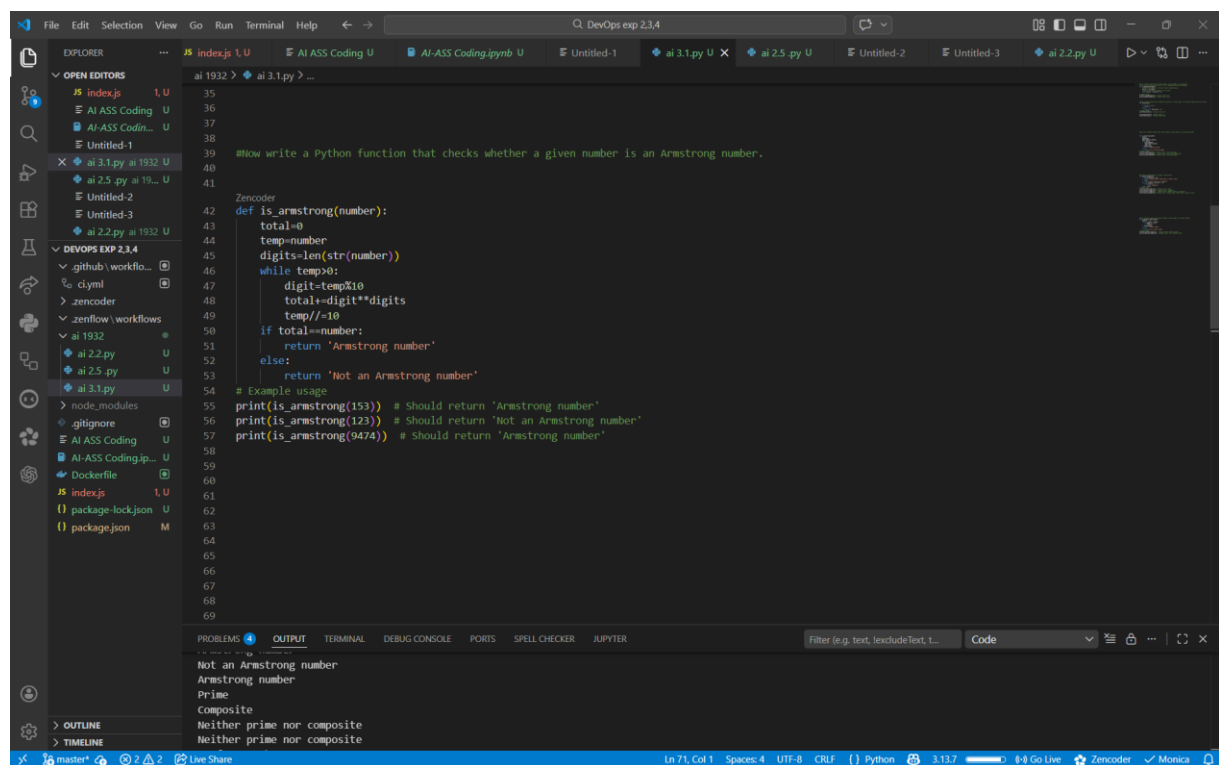
CONTEXT-MANAGED PROMPTING (OPTIMIZED NUMBER CLASSIFICATION) PROMPT:

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:



```
File Edit Selection View Go Run Terminal Help
Q: DevOps exp 2.3.4

EXPLORER
  JS index.js 1, U
  AI ASS Coding U
  AI ASS Coding.ipynb U
  Untitled-1
  ai 3.1.py ai 1932 U
  ai 2.5.py ai 19... U
  Untitled-2
  Untitled-3
  ai 2.2.py ai 1932 U
  DEVOPS EXP 2.3.4
    github/workflows
    ci.yml
    zencoder
    zenflow/workflows
    ai 1932
      ai 2.2.py U
      ai 2.5.py U
      ai 3.1.py U
    node_modules
    .gitignore
    AI ASS Coding U
    AI ASS Coding.ip... U
    Dockerfile
    JS index.js 1, U
    package-lock.json U
    package.json M

ai 1932 > ai 3.1.py > ...
35
36
37
38
39 #Now write a Python function that checks whether a given number is an Armstrong number.
40
41
42 Zencoder
43 def is_armstrong(number):
44     total=0
45     temp=number
46     digits=len(str(number))
47     while temp>0:
48         digit=temp%10
49         total+=digit**digits
50         temp//=10
51     if total==number:
52         return 'Armstrong number'
53     else:
54         return 'Not an Armstrong number'
55
56 # Example usage
57 print(is_armstrong(153)) # Should return 'Armstrong number'
58 print(is_armstrong(123)) # Should return 'Not an Armstrong number'
59 print(is_armstrong(9474)) # Should return 'Armstrong number'
60
61
62
63
64
65
66
67
68
69

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS SPELL CHECKER JUPYTER
Filter (e.g. text, excludeText, L...) Code
Not an Armstrong number
Armstrong number
Prime
Composite
Neither prime nor composite
Neither prime nor composite

Ln 71, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.7 Go Live Zencoder Monica
```

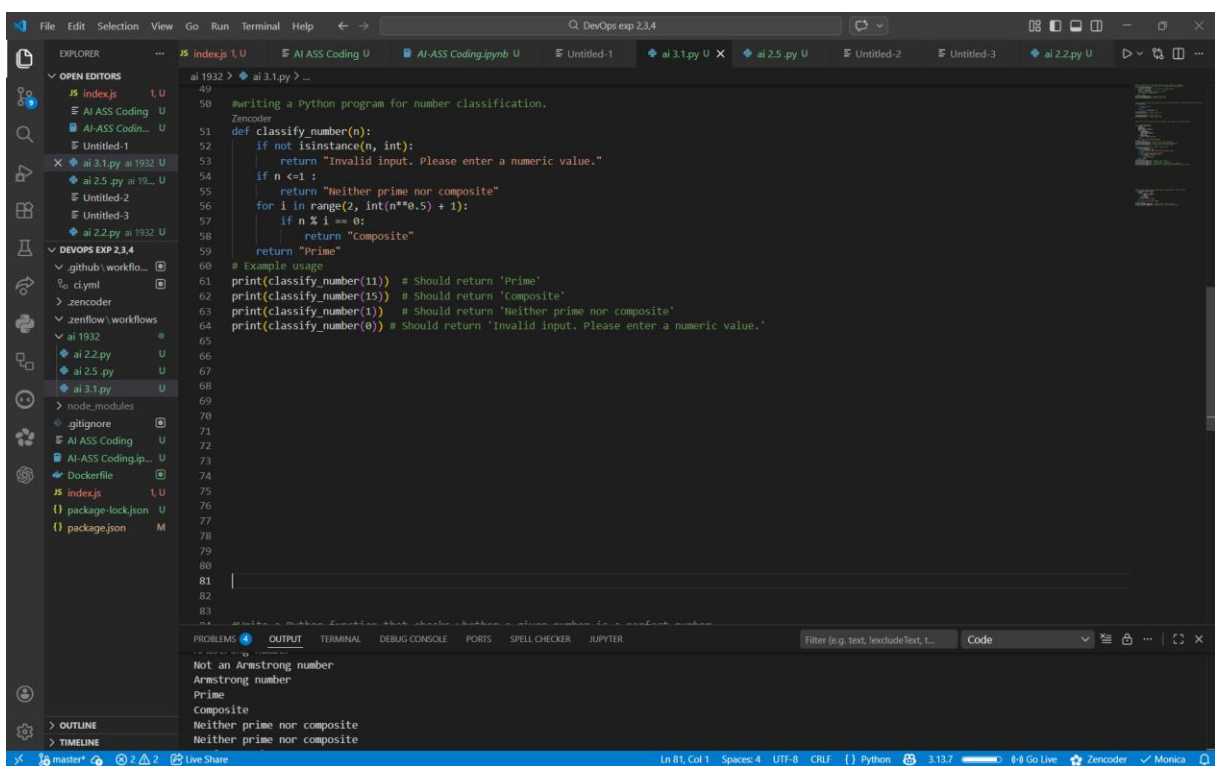
## OBSERVATION:

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:



```
49
50 #writing a Python program for number classification.
51
52 def classify_number(n):
53     if not isinstance(n, int):
54         return "Invalid input. Please enter a numeric value."
55     if n <= 1:
56         return "Neither prime nor composite"
57     for i in range(2, int(n**0.5) + 1):
58         if n % i == 0:
59             return "Composite"
60     return "Prime"
61
62 # Example usage
63 print(classify_number(11)) # Should return 'Prime'
64 print(classify_number(15)) # Should return 'Composite'
65 print(classify_number(1)) # Should return 'Neither prime nor composite'
66 print(classify_number(0)) # Should return 'Invalid input. Please enter a numeric value.'
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS SPELL CHECKER JUPYTER

Not an Armstrong number  
Armstrong number  
Prime  
Composite  
Neither prime nor composite  
Neither prime nor composite

## 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:

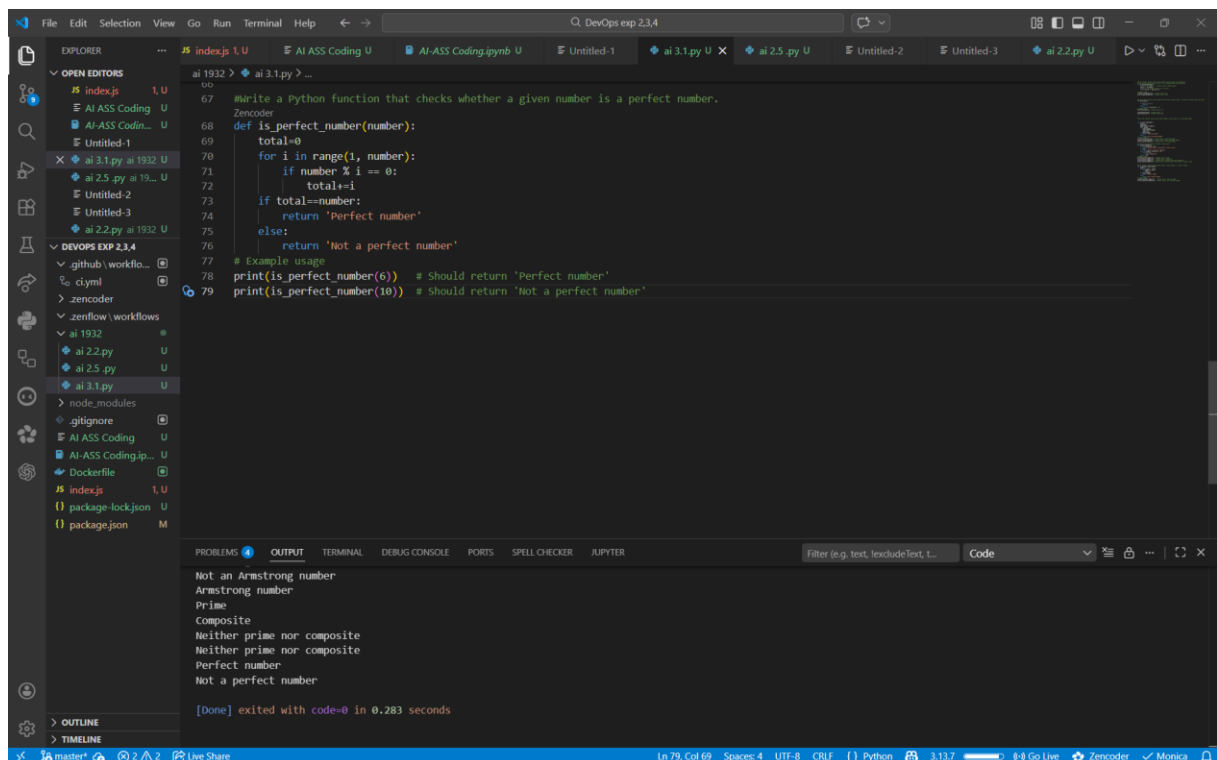
Input: 15 Output:

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:



```
67 #Write a Python function that checks whether a given number is a perfect number.
68 def is_perfect_number(number):
69     total=0
70     for i in range(1, number):
71         if number % i == 0:
72             total+=i
73     if total==number:
74         return 'Perfect number'
75     else:
76         return 'Not a perfect number'
77 # Example usage
78 print(is_perfect_number(6)) # Should return 'Perfect number'
79 print(is_perfect_number(10)) # Should return 'Not a perfect number'
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS SPELL CHECKER JUPYTER

Filter (e.g. test, leetcode, test, L...

Code

Not an Armstrong number  
Armstrong number  
Prime  
Composite  
Neither prime nor composite  
Neither prime nor composite  
Perfect number  
Not a perfect number

[Done] exited with code=0 in 0.283 seconds

OBSERVATION:

Negative integer are handled correctly Program safely rejected non integer inputs  
Improve input handling Clear and consistent output