

ASSIGNMENT -3.1

BATCH-30

ROLL-NO:2303A510G7

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 screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files `ai.py` and `Assignment-3.1.py`.
- Code Editor:** Displays the `Assignment-3.1.py` file containing the following Python code:

```
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(123))    # Expected: True
16 print(is_palindrome(1221))   # Expected: False
17 print(is_palindrome(10))     # Expected: False
18 print(is_palindrome(7))      # Expected: True
19
```
- Terminal:** Shows the command `PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py` and its output:

```
True
False
True
False
True
True
PS C:\Users\Jashwanth\AI coding>
```
- Status Bar:** Shows "Line 4, Col 1" and "Python 3.12.10 (Microsoft Store)".

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:

```
# Write a python function that compute the factorial of given number. The function should return the result.
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Example usage
print(factorial(5)) # Expected: 120
print(factorial(0)) # Expected: 1
print(factorial(1)) # Expected: 1
print(factorial(4)) # Expected: 24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
True
False
True
False
True
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
File "d:/Assignment-3.1.py", line 2
    Example:
SyntaxError: invalid syntax
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
120
1
1
24
PS C:\Users\Jashwanth\AI 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 the VS Code interface with the following details:

- Explorer:** Shows files in the workspace, including `a.py` and `Assignment-3.1.py`.
- Code Editor:** Displays the Python code for an Armstrong number checker. The code defines a function `is_armstrong` that calculates the sum of digits raised to the power of their count and compares it to the original number.
- Terminal:** Shows the command `python Assignment-3.1.py` being run, followed by the output:

```
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Armstrong Number
Armstrong Number
Not an Armstrong Number
Not an Armstrong Number
Not an Armstrong Number
PS C:\Users\Jashwanth\AI coding>
```
- Status Bar:** Shows the current file is `Assignment-3.1.py`, line 1, column 84, and the Python version is 3.12.10 (Microsoft Store).

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 the VS Code interface with the following details:

- File Explorer:** Shows a folder named "AI CODING" containing ".vscode" and "ai.py".
- Code Editor:** Displays the file "Assignment3.1.py" with the following code:

```
1 #Writing a python program for number classification.
2 def classify_number(n):
3     if not isinstance(n, int):
4         return "Invalid input"
5
6     if n <= 1:
7         return "Neither Prime nor Composite"
8
9     for i in range(2, int(n ** 0.5) + 1):
10        if n % i == 0:
11            return "Composite"
12
13    return "Prime"
14
15 print(classify_number(2))
16 print(classify_number(1))
17 print(classify_number(10))
18 print(classify_number(1))
19 print(classify_number(0))
```
- Terminal:** Shows the output of running the script:

```
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Armstrong Number
Armstrong Number
Not Armstrong Number
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Prime
Prime
Composite
Neither Prime nor Composite
Neither Prime nor Composite
PS C:\Users\Jashwanth\AI coding>
```
- Bottom Status Bar:** Shows "Ln 18, Col 26" and "Python 3.12.10 (Microsoft Store)".

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 Microsoft Visual Studio Code (VS Code) interface. The top bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and AI coding. The Explorer sidebar on the left shows a folder structure with 'Assignment-3.1.py' selected. The main editor area contains the following Python code:

```
D:\> ai.py Assignment-3.1.py
1 Write a Python function that checks whether a given number is a perfect number.
2
3     def is_perfect(number):
4         total = 0
5
6         for i in range(1, number):
7             if number % i == 0:
8                 total += i
9
10        if total == number:
11            return "Perfect Number"
12        else:
13            return "Not a Perfect Number"
14
15 print(is_perfect(6)) # 1 + 2 + 3 = 6
16 print(is_perfect(28)) # 1 + 2 + 4 + 7 + 14 =
```

The terminal at the bottom shows command-line history:

```
PS C:\Users\JashwanthAI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Armstrong Number
Armstrong Number
Not an Armstrong Number
PS C:\Users\JashwanthAI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Prime
Prime
Composite
Neither Prime nor Composite
Neither Prime nor Composite
PS C:\Users\JashwanthAI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Perfect Number
Perfect Number
PS C:\Users\JashwanthAI coding>
```

Bottom status bar: Ln 15, Col 51 | Spaces: 4 | UTF-8 | CRLF | Python | 3.12.10 (Microsoft Store)

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

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder named "AI CODING" containing ".vscode" and "ai.py".
- Code Editor:** Displays the file "Assignment3.1.py" with the following code:

```
1  #Write a python program that determines whether a given number is Even or Odd
2
3  def check_even_odd(value):
4      if not isinstance(value, int):
5          return "Invalid input"
6
7      if value % 2 == 0:
8          return "Even"
9      else:
10         return "Odd"
11
12 print(check_even_odd(1))
13 print(check_even_odd(15))
14 print(check_even_odd(0))
```
- Terminal:** Shows the output of running the script:

```
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Armstrong Number
Armstrong Number
Not Armstrong Number
Prime
Prime
Composite
Neither Prime nor Composite
Neither Prime nor Composite
Perfect Number
Perfect Number
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Odd
Even
PS C:\Users\Jashwanth\AI coding>
```
- Output Panel:** Shows the following messages:
 - "powershell" terminal has 1 message.
 - "Python" terminal has 1 message.
 - "Python" terminal has 1 message.
- Status Bar:** Shows "In 1, Col 79" and "3.12.10 (Microsoft Store)".

OBSERVATION:

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