

ASSIGNMENT -3.1

NAME-HABEEBA KHANAM

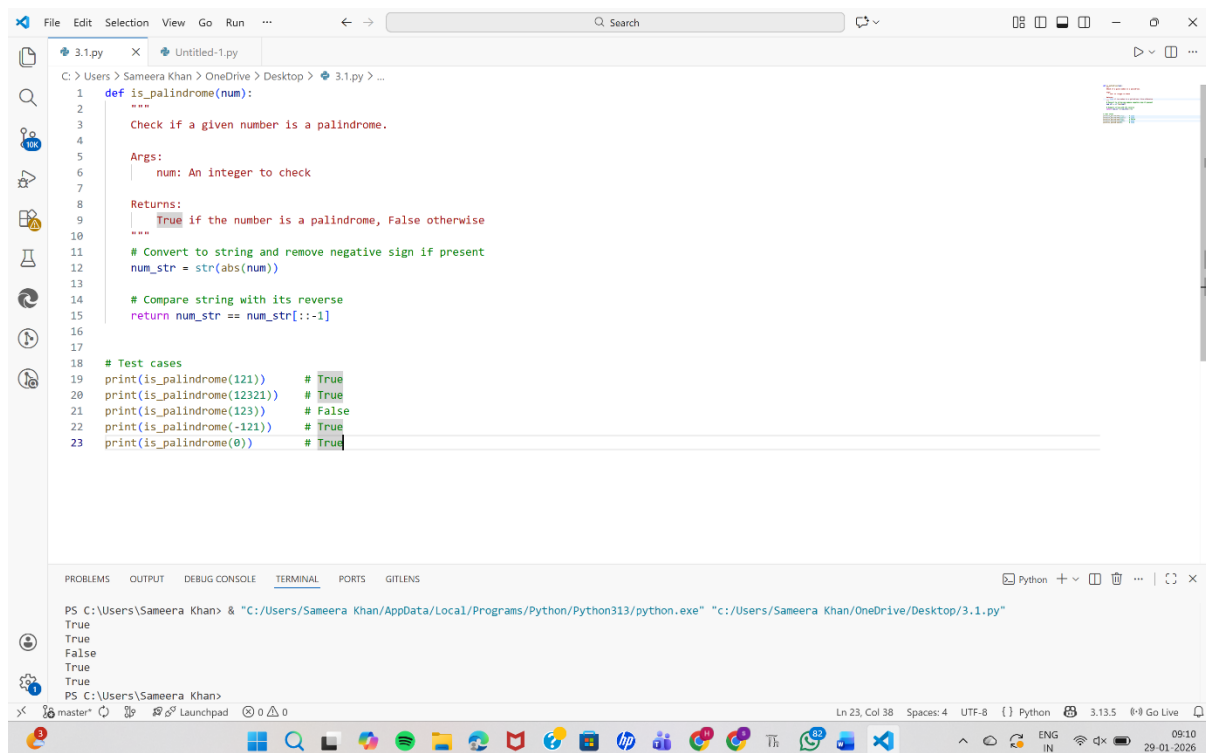
ROLL_NO:2303A51474

BATCH-29

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.



```
1 def is_palindrome(num):
2     """
3     Check if a given number is a palindrome.
4
5     Args:
6         num: An integer to check
7
8     Returns:
9         True if the number is a palindrome, False otherwise
10    """
11    # Convert to string and remove negative sign if present
12    num_str = str(abs(num))
13
14    # Compare string with its reverse
15    return num_str == num_str[::-1]
16
17
18 # Test cases
19 print(is_palindrome(121)) # True
20 print(is_palindrome(12321)) # True
21 print(is_palindrome(123)) # False
22 print(is_palindrome(-121)) # True
23 print(is_palindrome(0)) # True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
True
True
False
True
True
```

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.

The screenshot displays a Windows 10 desktop environment. The primary focus is the Visual Studio Code (VS Code) application, which is open with a Python file named '3.1.py'. The code in the editor defines a 'factorial' function that calculates the factorial of a given non-negative integer 'n'. It includes a docstring, argument description, and a return statement. The function uses a loop to calculate the factorial and raises a 'ValueError' for negative numbers. Below the function definition, there is an example usage line: 'print(factorial(4)) # Output: 24'. The VS Code interface shows the 'TERMINAL' tab is active, displaying the command prompt output for the script execution, which shows 'True' and '24'. The taskbar at the bottom shows various application icons, including the Start menu, Search, File Explorer, and several instances of VS Code. The system tray in the bottom right corner shows the date and time as 09:31 on 29-01-2022.

```

C:\> Users > Sameera Khan > OneDrive > Desktop > 3.1.py > ...
1  def factorial(n):
2      """
3      Calculate the factorial of a given number.
4
5      Args:
6          n: A non-negative integer
7
8      Returns:
9          The factorial of n
10     """
11
12     if n < 0:
13         raise ValueError("Factorial is not defined for negative numbers")
14     if n == 0 or n == 1:
15         return 1
16
17     result = 1
18     for i in range(2, n + 1):
19         result *= i
20     return result
21
22 # Example usage
23 print(factorial(4)) # Output: 24

```

```

True
True
PS C:\Users\Sameera Khan> "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
PS C:\Users\Sameera Khan> "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
PS C:\Users\Sameera Khan> "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
24
PS C:\Users\Sameera Khan>

```

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:

```
1 def check_armstrong(num):
2     """
3     Check whether a given number is an Armstrong number.
4     An Armstrong number is a number that is equal to the sum of its own digits
5     each raised to the power of the number of digits.
6     """
7     num_str = str(num)
8     num_digits = len(num_str)
9     sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
10
11     if sum_of_powers == num:
12         return "Armstrong Number"
13     else:
14         return "Not an Armstrong Number"
15
16 # Test cases
17 print(check_armstrong(153)) # Armstrong Number
18 print(check_armstrong(370)) # Armstrong Number
19 print(check_armstrong(123)) # Not an Armstrong Number
```

```
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "C:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
24
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "C:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
Armstrong Number
Armstrong Number
Not an Armstrong Number
PS C:\Users\Sameera Khan>
```

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.

CODE:

```
1 def classify_number(num):
2
3     # Validate input
4     if not isinstance(num, int) or isinstance(num, bool):
5         return "Error: Please enter a valid integer."
6
7     if num < 0:
8         return "Error: Please enter a non-negative integer."
9
10    # Handle special cases
11    if num == 0 or num == 1:
12        return f"{num} is Neither prime nor composite."
13
14    if num == 2:
15        return "2 is Prime."
16
17    # Check if even
18    if num % 2 == 0:
19        return f"{num} is Composite."
20
21    # Check odd divisors up to sqrt(num)
22    i = 3
23    while i * i <= num:
24        if num % i == 0:
25            return f"{num} is Composite."
26        i += 2
27
28    return f"{num} is Prime."
29
30
31
32 # Main program
33 if __name__ == "__main__":
34     try:
35         user_input = int(input("Enter an integer: "))
36         result = classify_number(user_input)
37         print(result)
38     except ValueError:
39         print("Error: Please enter a valid integer.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS

Enter an integer: 25
25 is Composite.
PS C:\Users\Sameera Khan> 29
PS C:\Users\Sameera Khan> 28
PS C:\Users\Sameera Khan> 28

OBSERVATION:

The role is defined

Constraints are clearly stated

Efficiency and validation of the code

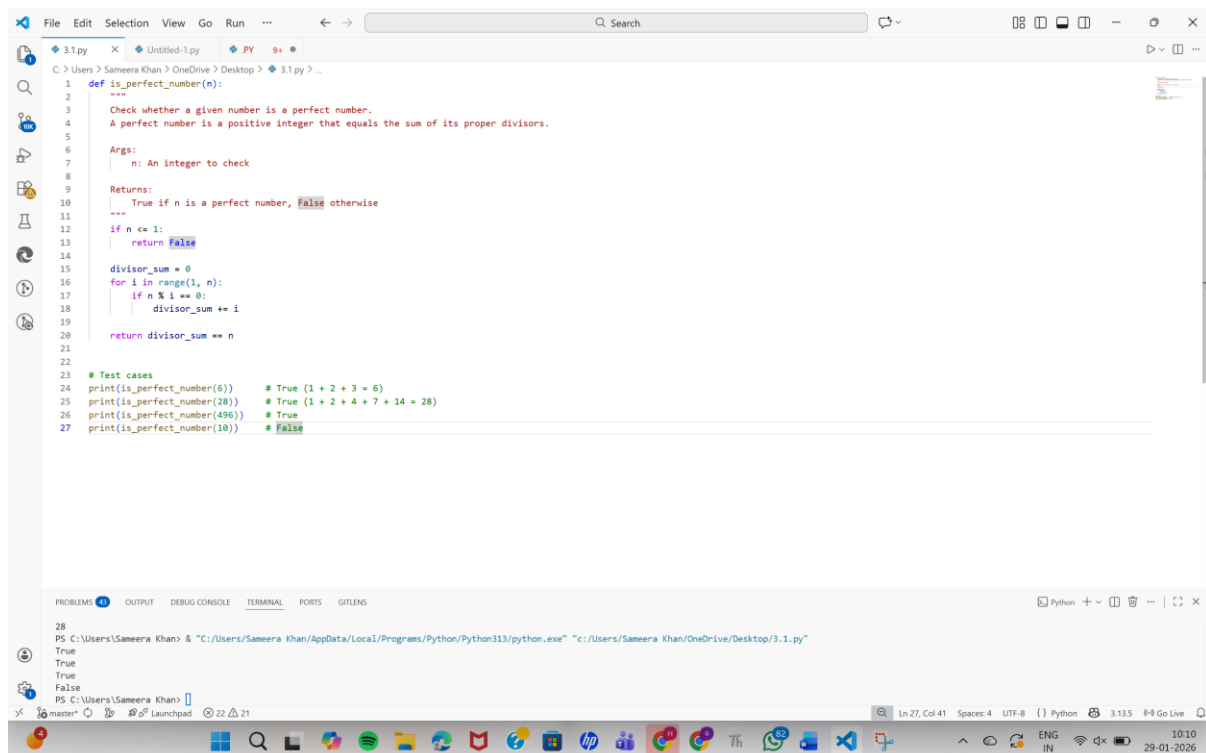
But the inputs should be specified more clearly mentioned

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:



```
1 def is_perfect_number(n):
2     """
3     Check whether a given number is a perfect number.
4     A perfect number is a positive integer that equals the sum of its proper divisors.
5
6     Args:
7         n: An integer to check
8
9     Returns:
10         True if n is a perfect number, False otherwise
11     """
12     if n <= 1:
13         return False
14
15     divisor_sum = 0
16     for i in range(1, n):
17         if n % i == 0:
18             divisor_sum += i
19
20     return divisor_sum == n
21
22
23 # Test cases
24 print(is_perfect_number(6))    # True (1 + 2 + 3 = 6)
25 print(is_perfect_number(28))  # True (1 + 2 + 4 + 7 + 14 = 28)
26 print(is_perfect_number(496)) # True
27 print(is_perfect_number(10))  # False
```

Terminal Output:

```
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
True
True
False
PS C:\Users\Sameera Khan>
```

OBSERVATION:

No input validation – if negative or float any..

Inefficient for large input

Did not specify input constraints

No edge case handling seen

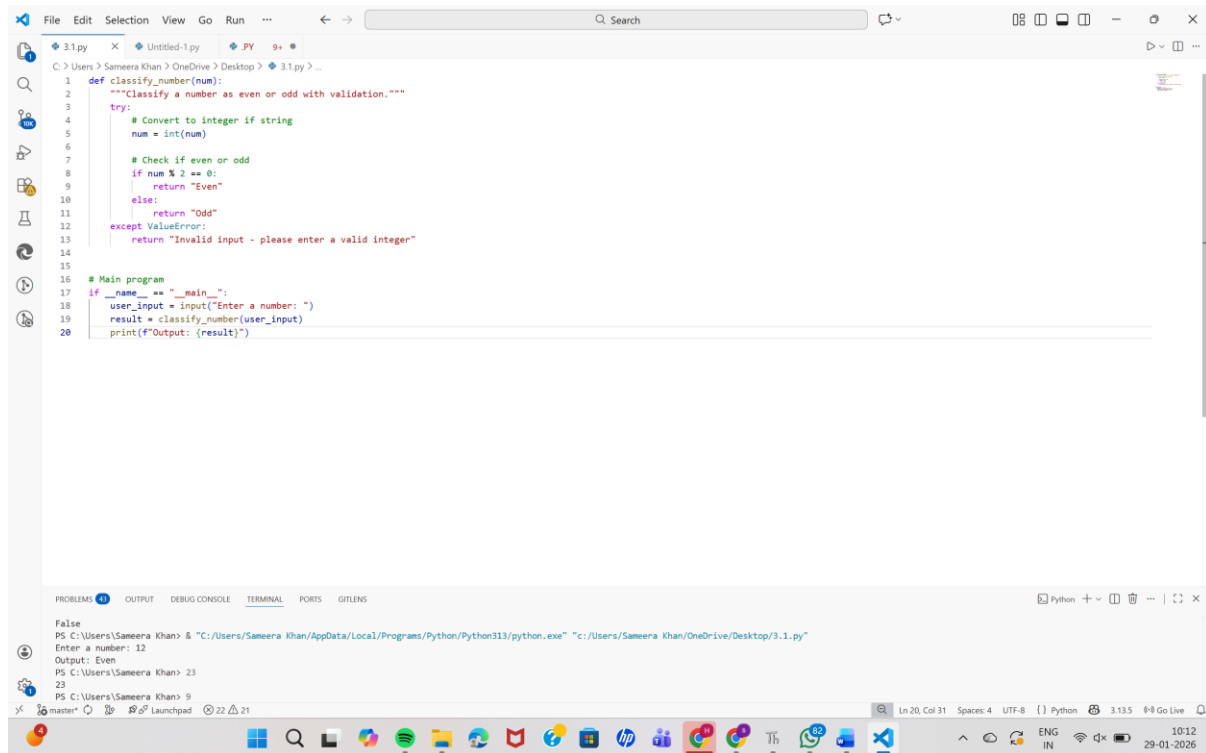
TASK-6: FEW-SHOT PROMPTING (EVEN OR ODD CLASSIFICATION WITH VALIDATION)

PROMPT:

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:



```
1 def classify_number(num):
2     """Classify a number as even or odd with validation."""
3     try:
4         # Convert to integer if string
5         num = int(num)
6
7         # Check if even or odd
8         if num % 2 == 0:
9             return "Even"
10        else:
11            return "Odd"
12    except ValueError:
13        return "Invalid input - please enter a valid integer"
14
15
16 # Main program
17 if __name__ == "__main__":
18     user_input = input("Enter a number: ")
19     result = classify_number(user_input)
20     print(f"Output: {result}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
False
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
Enter a number: 12
Output: Even
PS C:\Users\Sameera Khan>
23
PS C:\Users\Sameera Khan> 9
```

OBSERVATION:

Negative integer are handled correctly

Program safely rejected non integer inputs

Improve input handling

Clear and consistent output

