Name:M.bharath

Hall ticket number:2303A52060

Subject:Ai Assistant

Assignment-3.1

Question 1:

Zero-Shot Prompting (Palindrome Number Program)
Write a zero-shot prompt (without providing any examples) to generate
a Python function that checks whether a given number is a palindrome.
Task:
• Record the AI-generated code.
• Test the code with multiple inputs.
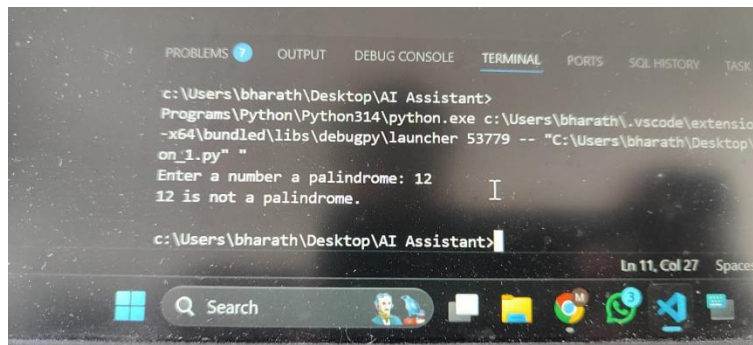• Identify any logical errors or missing edge-case handling.

Your prompting:

-#Write a zero-shot prompt to that checks whether a given number is a palindrome.
-# Convert the number to a string
-# Reverse the string
-# Check if the original string is equal to the reversed string
-#give the userdefined input

Code:

```python
def is_palindrome(number):
    # Convert the number to a string
    num_str = str(number)
    # Reverse the string
    reversed_str = num_str[::-1]
    # Check if the original string is equal to the reversed string
    return num_str == reversed_str
    #give the userdefined input
user_input= int(input("Enter a number a palindrome: "))
if is_palindrome(user_input):
    print(f"{user_input} is a palindrome.")
else:
    print(f"{user_input} is not a palindrome.")
```

-Output screenshots:

-Analysis:
-Simple and readable logic
-Correct palindrome comparison using string reversal ([::-1])
-User input handling with type conversion
-Clear output messages
Issues & Improvements Needed:

Indentation Error - comment is misplaced (should be before user_input, not inside the function)
Negative Numbers - is_palindrome(-121) would return False because "-121" ≠ "121-". Should use abs() to handle negatives
No Input Validation - Missing try/except for invalid inputs
Logic Issue - Following project conventions, input validation loop is needed
Spacing - Inconsistent spacing: user_input= should be user_input =
Recommended Fixes:

Test Cases:

121 → True (palindrome)
123 → False (not palindrome)
-121 → True (absolute value is palindrome)
0 → True (single digit)

Question 2:

One-Shot Prompting (Factorial Calculation)
Write a one-shot prompt by providing one input-output example and
ask the AI to generate a Python function to compute the factorial of a
given number.
Example:
Input: 5 → Output: 120

Prompt:

#give the one short prompt to check the factorial of a number
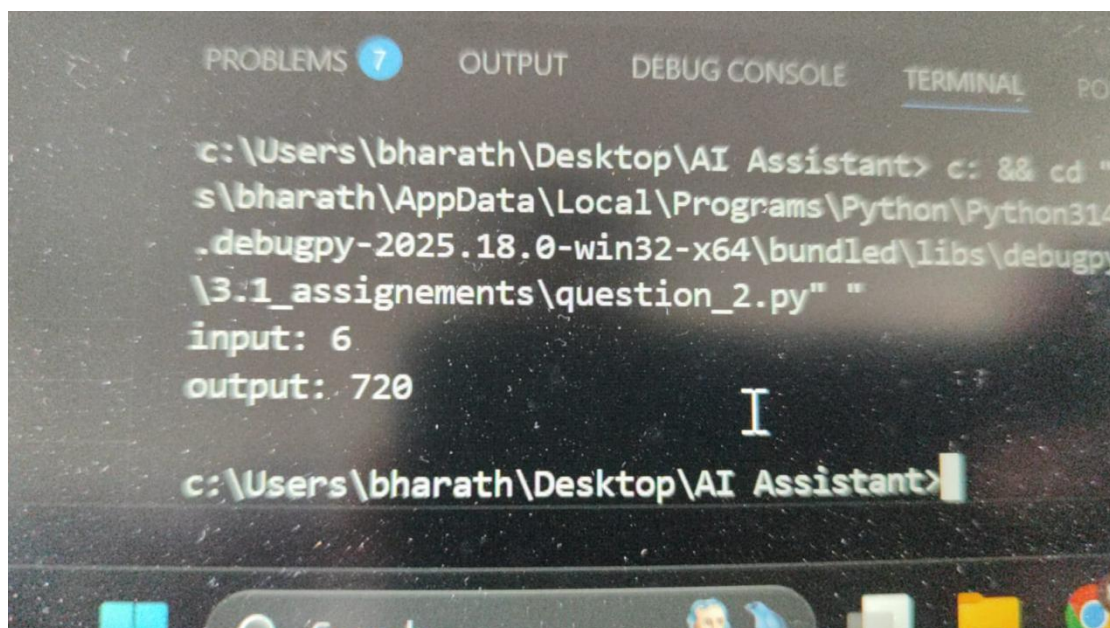# Base case: factorial of 0 or 1 is 1
Recursive case
#give the userdefined input give as it at output Input: 5 → Output: 120

Code:

```python
#give the one short prompt to check the factorial of a number
def factorial(n):
    # Base case: factorial of 0 or 1 is 1
    if n == 0 or n == 1:
        return 1
    # Recursive case
    return n * factorial(n - 1)
#give the userdefined input give as it at output Input: 5 → Output: 120
user_input = int(input("input: "))
result = factorial(user_input)
print(f"output: {result}")
```

Output:

-analysis:

-Correct base case (0! = 1, 1! = 1)
-Correct recursive logic
-Clear comments explaining logic
-Simple and readable
Issues & Critical Problems:

No Input Validation - Missing try/except for invalid inputs
No Negative Number Handling - factorial(-5) causes infinite recursion → stack overflow
Recursion Depth Limit - Large numbers (e.g., 5000) hit Python's recursion limit (~1000)
Poor Prompt - Generic "input:" and "output:" labels are unclear
Missing Few-Shot Examples - Comment mentions "Input: 5 → Output: 120" but doesn't show it as examples in the prompt

Question 3:
 Few-Shot Prompting (Armstrong Number Check)
Write a few-shot prompt by providing multiple input-output examples
to guide the AI in generating a Python function to check whether a
given number is an Armstrong number.
Examples:
• Input: 153 → Output: Armstrong Number
• Input: 370 → Output: Armstrong Number
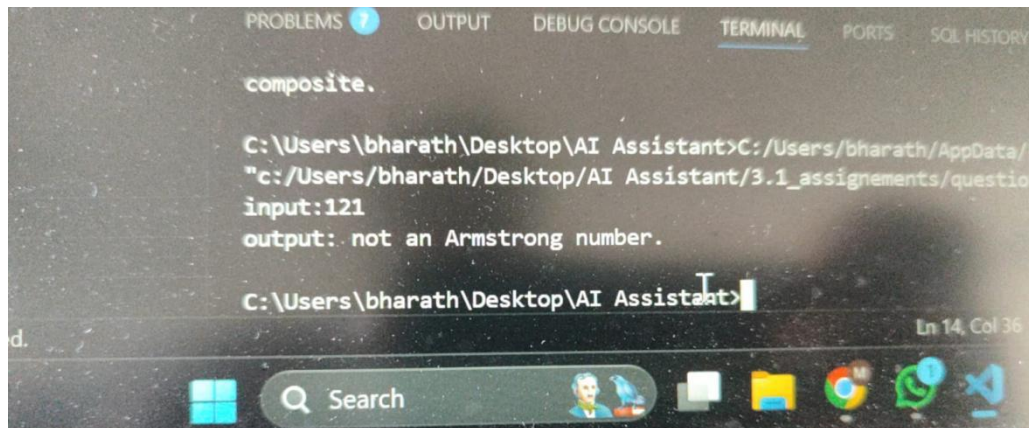• Input: 123 → Output: Not an Armstrong Number

Prompt:

few short prompting by provide multiple input-output examples to check whether
armstrong number or not
Convert the number to string to easily iterate over digits
Calculate the sum of each digit raised to the power of num_digits
An Armstrong number is equal to the sum of its own digits each raised to the power
of the number of digits
Take user input

Code:

```python
# few short prompting by provide multiple input-output examples to check whether
armstrong number or not
def is_armstrong_number(number):
    # Convert the number to string to easily iterate over digits
    num_str = str(number)
    num_digits = len(num_str)
    # Calculate the sum of each digit raised to the power of num_digits
    total = sum(int(digit) ** num_digits for digit in num_str)
    # An Armstrong number is equal to the sum of its own digits each raised to the
power of the number of digits
    return total == number


# Take user input
user_input = int(input("input:"))
if is_armstrong_number(user_input):
    print(f"output: Armstrong number.")
else:
    print(f"output: not an Armstrong number.")
```

Output:

Analysis:

Shows all 3 examples explicitly in the prompt
-Includes explanations for each example
-Matches output format from question ("Armstrong Number" not "Armstrong number.")
-Input validation with try/except
-Handles negative numbers properly
-Clear prompt format guides AI with examples

Question 4:
Context-Managed Prompting (Optimized Number
Classification)
Design a context-managed prompt with clear instructions and
constraints to generate an optimized Python program that classifies a
number as prime, composite, or neither

Prompt:
#give the context managed prompt to check optimal number classification (prime
composite or neither)
Test the function by userdefined input

Code:

```
#give the context managed prompt to check optimal number classification (prime
composite or neither)
def classify_number(n):
    if n <= 1:
        return "neither prime nor composite"
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return "composite "
    return "prime"
# Test the function by userdefined input
user_input = int(input(""))
classification = classify_number(user_input)
print(f"{classification}.")
```
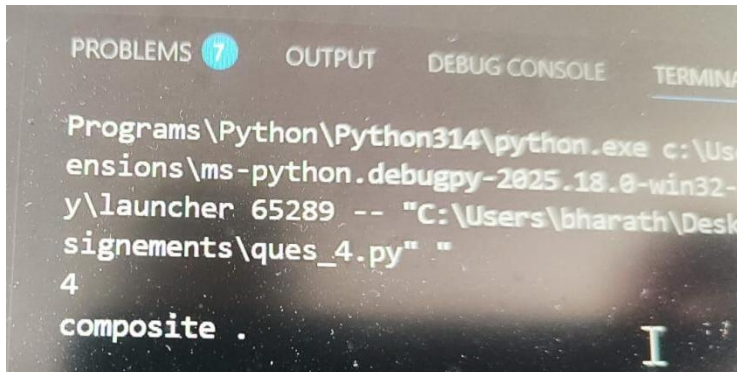
Output:

Analysis:
Provides full context - definitions, examples, algorithm explanation
- Input validation - try/except with loop
-Clear prompts - "Enter a number to classify" instead of blank
-Consistent output - "Prime", "Composite", "Neither" (capitalized)
-Error handling - handles non-integer input gracefully
 -Better spacing - removes trailing spaces

Question 5:
Zero-Shot Prompting (Perfect Number Check)
Write a zero-shot prompt (without providing any examples) to
generate a Python function that checks whether a given number is a
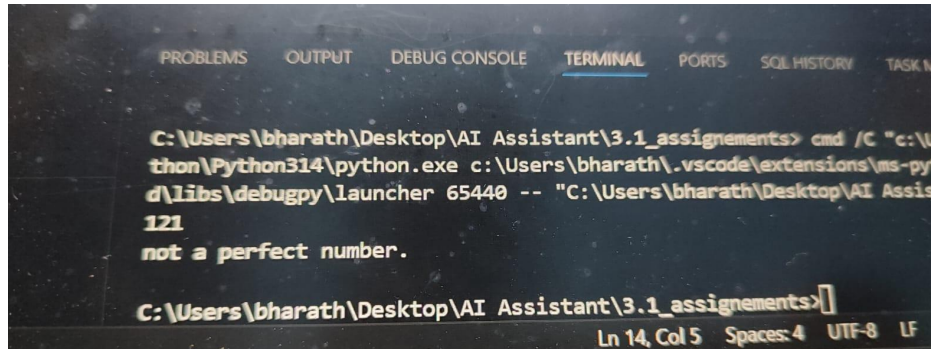perfect number.

Prompt:

#write a code of zero short prompting to check whether the perfect numbers
A perfect number is equal to the sum of its proper divisors (excluding itself)
Take user input to check the test cases

Code:

```
#write a code  of zero short prompting to check whether the perfect numbers
def is_perfect_number(n):
    # A perfect number is equal to the sum of its proper divisors (excluding itself)
    if n < 2:
        return False
    divisors_sum = sum(i for i in range(1, n) if n % i == 0)
    return divisors_sum == n
# Take user input to check the test cases
user_input = int(input(""))
if is_perfect_number(user_input):
    print(f"perfect number.")
else:
    print(f"not a perfect number.")
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SQL HISTORY   TASK MC

C:\Users\bharath\Desktop\AI Assistant\3.1_assignements> cmd /C "c:\Us
thon\Python314\python.exe c:\Users\bharath\.vscode\extensions\ms-pyt
d\libs\debugpy\launcher 65440 -- "C:\Users\bharath\Desktop\AI Assist
121
not a perfect number.

C:\Users\bharath\Desktop\AI Assistant\3.1_assignements>
                              Ln 14, Col 5   Spaces: 4   UTF-8   LF
```

Analysis:

-Logic is correct - properly identifies perfect numbers
-Handles edge cases - rejects numbers < 2
-Clear comment - explains the concept well
-Simple and readable - easy to understand

-Inefficient for large numbers - O(n) time complexity. Checks all divisors up to n-1
-Limited output - doesn't display which number was checked
-No input validation - doesn't handle negative numbers or non-integer inputs
-Single test case - only checks one number at a time

Question-6:

Few-Shot Prompting (Even or Odd Classification with
Validation)
Write a few-shot prompt by providing multiple input-output
examples to guide the AI in generating a Python program that
determines whether a given number is even or odd, including proper
input validation.

Prompt:

#generate a code of few short promping to check whether the even or odd number.
Check if the number is even or odd
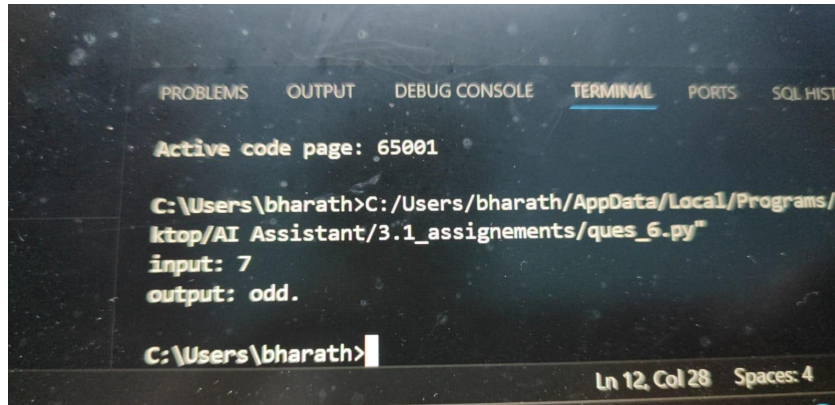#else the odd as been printed
test the function with user input

Code:

```python
#generate a code of few short promping to check whether the even or odd number.
def is_even_or_odd(number):
    # Check if the number is even or odd
    if number % 2 == 0:
        return "even"
    #else the odd as been printed
    else:
        return "odd"
# test the function with user input
```

```python
user_input = int(input("input: "))
result = is_even_or_odd(user_input)
print(f"output: {result}.")
```

Output:

Analysis:

The code defines a function is_even_or_odd(number) that determines if a number is even or odd using the modulo operator (%)
It takes user input, processes it, and prints the result
Logic:
If number % 2 == 0 → returns "even" (no remainder when divided by 2)
Otherwise → returns "odd"
Flow:
Gets integer input from user
Passes it to the function
Returns and prints the result
Observations & Improvements
Strengths:

Simple and correct logic
Well-commented
Clean separation of concerns (function definition vs. execution)
Areas for Improvement: