

AI ASSISTED CODING

ASSIGNMENT-3

Name: G.V.N Hasini

H.T.No: 2303A51099

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.

```
def is_palindrome(num):  
    """  
    Check if the given number is a palindrome or not.  
    A palindrome number reads the same forwards and backwards.  
  
    Args:  
        num: The number to check  
  
    Returns:  
        True if the number is a palindrome, False otherwise  
    """  
    # Convert number to string and remove negative sign if present  
    num_str = str(abs(num))  
  
    # Check if the string is equal to its reverse  
    return num_str == num_str[::-1]  
  
# Main program  
if __name__ == "__main__":  
    try:  
        # Get user input  
        user_input = input("Enter a number to check if it's a palindrome: ")  
        number = int(user_input)  
  
        # Check if the number is a palindrome  
        if is_palindrome(number):  
            print(f"{number} is a palindrome number!")  
        else:  
            print(f"{number} is not a palindrome number.")  
  
    except ValueError:  
        print("Invalid input! Please enter a valid integer.")
```

```
PS C:\Users\hasin\OneDrive\Desktop\java_programming> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "C:/Users/hasin/OneDrive/Desktop/java_programming/assignment_3"
Enter a number to check if it's a palindrome: 121
121 is a palindrome number!
PS C:\Users\hasin\OneDrive\Desktop\java_programming> -121
-121
PS C:\Users\hasin\OneDrive\Desktop\java_programming> 734
734
PS C:\Users\hasin\OneDrive\Desktop\java_programming> 444
444
PS C:\Users\hasin\OneDrive\Desktop\java_programming> 323
323
```

Ln 34, Col 1 Spaces

There are no logical errors in the code generated. Negative Numbers and invalid input(Strings) are handled using try and except.

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

Task:

- Compare the generated code with a zero-shot solution.
- Examine improvements in clarity and correctness.

```
def compute_factorial(n):
    """
    Compute the factorial of a given number.

    One-shot example:
    Input: 5
```

- Examine improvements in clarity and correctness.

```
def compute_factorial(n):
    """
    Compute the factorial of a given number.

    One-shot example:
    Input: 5
    Output: 120
    (because 5! = 5 × 4 × 3 × 2 × 1 = 120)

    Args:
        n: A non-negative integer

    Returns:
        The factorial of n
    ....
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1

    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
```

Comparison of Factorial Programs Using One-Shot and Zero-Shot Prompts

Aspect	Program 1: One-Shot Prompt	Program 2: Zero-Shot Prompt
Definition	Provides a single example and directly computes the factorial	Solves the problem without prior examples using logic and validation
Program Structure	Single block of code	Function-based with main execution logic
Modularity	Not modular	Modular design
Correctness for Positive Integers	Produces correct results	Produces correct results
Handling of Zero (0!)	Correctly returns 1	Correctly returns 1
Handling of Negative Numbers	Not handled; produces incorrect output	Properly handled with validation message
Invalid Input Handling	No handling for invalid input	Uses exception handling (try-except)

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

Task:

- Analyze how multiple examples influence code structure and accuracy.
- Test the function with boundary values and invalid inputs.

```
def is_armstrong(num):  
    num_str = str(num)  
    n = len(num_str)  
    sum_of_powers = 0  
    for digit_char in num_str:  
        sum_of_powers += int(digit_char) ** n  
    return sum_of_powers == num  
  
if __name__ == "__main__":  
    try:  
        number = int(input("Enter a number: "))  
        if is_armstrong(number):  
            print(f"{number} is an Armstrong number.")  
        else:  
            print(f"{number} is not an Armstrong number.")  
    except ValueError:  
        print("Invalid input. Please enter an integer.")
```

```
131 def is_armstrong(num):
132     num_str = str(num)
133     n = len(num_str)
134     sum_of_powers = 0
135     for digit_char in num_str:
136         sum_of_powers += int(digit_char) ** n
137     return sum_of_powers == num
138
139 if __name__ == "__main__":
140     try:
141         number = int(input("Enter a number: "))
142         if is_armstrong(number):
143             print(f"{number} is an Armstrong number.")
144         else:
145             print(f"{number} is not an Armstrong number.")
146     except ValueError:
147         print("Invalid input. Please enter an integer.")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL FOLDS +
```

```
+ CategoryInfo : ObjectNotFound: (abc:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\hasin\OneDrive\Desktop\java programming> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "C:/users/hasin/OneDrive/Desktop/java programming/abc3.py"
Enter a number: 153
153 is an Armstrong number.
PS C:\Users\hasin\OneDrive\Desktop\java programming> 123
123
PS C:\Users\hasin\OneDrive\Desktop\java programming> -153
-153
PS C:\Users\hasin\OneDrive\Desktop\java programming>
```

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.

Task:

- Ensure proper input validation.
- Optimize the logic for efficiency.
- Compare the output with earlier prompting strategies.

- Compare the output with earlier prompting strategies.

```
89 def is_prime(num):  
90     if num < 2:  
91         return "neither"  
92     for i in range(2, int(num**0.5) + 1):  
93         if num % i == 0:  
94             return "composite"  
95     return "prime"  
96  
97 if __name__ == "__main__":  
98     try:  
99         number = int(input("Enter an integer: "))  
100        result = is_prime(number)  
101        print(f"The number {number} is {result}.")  
102    except ValueError:  
103        print("Invalid input. Please enter an integer.")  
104    except Exception as e:  
105        print(f"An unexpected error occurred: {e}")  
106  
107  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\Yusin\OneDrive\Desktop\Java programming> & C:\Users\Yusin\AppData\Local\Programs\Python\Python33\python.exe "c:/users/yusin/Desktop/Java programming/assignment_3"  
Enter an integer: 7  
The number 7 is prime.  
PS C:\Users\Yusin\OneDrive\Desktop\Java programming> 7  
PS C:\Users\Yusin\OneDrive\Desktop\Java programming> 4  
PS C:\Users\Yusin\OneDrive\Desktop\Java programming> abc  
abc : The term 'abc' is not recognized as the name of a cmdlet, function, script file, or executable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.  
At line:1 char:1
```

```

def is_prime(num):
    if num < 2:
        return "neither"
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return "composite"
    return "prime"

if __name__ == "__main__":
    try:
        number = int(input("Enter an integer: "))
        result = is_prime(number)
        print(f"The number {number} is {result}.")
    except ValueError:
        print("Invalid input. Please enter an integer.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

```

Comparison (optimized vs earlier prompting strategy)

- **Original prompt:** Role: Efficient python programmer Task: Write a python code to classify the given number as prime or composite or neither for only positive numbers.
- **Original outcome:** classify_number used vanilla trial division from 2 to $\text{int}(\text{num}^{**0.5})+1$; accepted zero as positive (if $a < 0$), no type hints, no stdlib helpers.
- **Optimized changes:**
 - **Algorithm:** early-case for 2, early even-number rejection, loop only odd divisors: `for i in range(3, math.isqrt(num)+1, 2)`
 - **Implementation:** uses `math.isqrt`, adds type hints `def classify_number(num: int) -> str`
 - **Validation:** input check tightened to reject $a \leq 0$ (zero treated as non-positive).
- **Performance impact:** same asymptotic complexity $O(\sqrt{n})$ but halves divisor checks (skip evens) and uses integer sqrt for a small constant-speed gain; noticeably faster for large odd numbers.
- **Readability / correctness:** clearer intent, fewer iterations, safer integer sqrt, and explicit types improve maintainability.

```

63
62     def is_perfect_number(number):
63         if number <= 0:
64             return False
65
66         divisors_sum = 1 # 1 is always a divisor
67         for i in range(2, int(number**0.5) + 1):
68             if number % i == 0:
69                 divisors_sum += i
70             if i * i != number: # Avoid adding the same divisor twice for perfect squares
71                 divisors_sum += number // i
72
73         return divisors_sum == number
74
75     def main():
76         try:
77             num = int(input("Enter a number to check if it's a perfect number: "))
78             if is_perfect_number(num):
79                 print(f"{num} is a perfect number.")
80             else:
81                 print(f"{num} is not a perfect number.")
82         except ValueError:
83             print("Invalid input. Please enter an integer.")
84
85     if __name__ == "__main__":
86         main()
87

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL FORTS

```

SyntaxError: unterminated string literal (detected at line 39)
PS C:\Users\hasin\OneDrive\Desktop\java programming> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "C:/Users/hasin/OneDrive/Desktop/java programming/assignment_3"
Enter a number to check if it's a perfect number: 6
6 is a perfect number.
PS C:\Users\hasin\OneDrive\Desktop\java programming> 8
8
PS C:\Users\hasin\OneDrive\Desktop\java programming> -5
-5
PS C:\Users\hasin\OneDrive\Desktop\java programming> abc
abc : The term 'abc' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if it is included, verify that the path is correct and try again.

```

1. The program is **logically correct**, but it redundantly checks for negative numbers both inside the function and in the main code.
2. It does not explicitly handle or explain why **0 and 1 are not perfect numbers**.
3. The divisor calculation is **inefficient ($O(n)$)** as it checks all numbers from 1 to $\text{num}-1$.
4. The logic does not use the **square-root optimization**, causing unnecessary iterations.
5. For large inputs, the program may be **slow and inefficient**, even though the result is correct.

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.

Examples:

- Input: 8 → Output: Even
- Input: 15 → Output: Odd
- Input: 0 → Output: Even

```
109 def is_even_or_odd(number):
110     if number % 2 == 0:
111         return "Even"
112     else:
113         return "Odd"
114
115 def main():
116     while True:
117         user_input = input("Enter an integer: ")
118         try:
119             num = int(user_input)
120             break # exit loop if input is a valid integer
121         except ValueError:
122             print("Invalid input. Please enter an integer.")
123
124     result = is_even_or_odd(num)
125     print(f"The number {num} is {result}.")
126
127 if __name__ == "__main__":
128     main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\hasin\OneDrive\Desktop\java programming> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java
programming/assignment_3"
Enter an integer: 2
The number 2 is Even.
PS C:\Users\hasin\OneDrive\Desktop\java programming> -2
-2
PS C:\Users\hasin\OneDrive\Desktop\java programming> 5
5
PS C:\Users\hasin\OneDrive\Desktop\java programming> abc
abc : The term 'abc' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
```