

ASSIGNMENT 3.1

2303A52386 Name: Tharun Batch : 40

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.

Promt:

write a python programm to determine whether the user given integer is a perfect number or not give the optimised code and cover all edge cases too and make the code as concise as possible and dont use comments

Code:

```
n = int(input("Enter an integer: "))

s = 1

i = 2

while i * i <= n:

    if n % i == 0:

        s += i + (n // i if i * i != n else 0)

    i += 1

print("Perfect number" if n > 1 and s == n else "Not a perfect number")
```

Output:

```
PS D:\AI_coding> & C:/Python314.  
Could not find platform independent  
Enter an integer: 6  
Perfect number  
PS D:\AI_coding> & C:/Python314.  
Could not find platform independent  
Enter an integer: 100  
Not a perfect number  
PS D:\AI_coding> █
```

Analysis:

Logic is correct for positive integers

Does not handle negative numbers (e.g., -121)

No input validation

Simple and readable

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.

Promt:

Input: 5 Output: 120

Using the above example, write a Python function to calculate the factorial of a given number.

Code

```
def factorial(n):  
    return 1 if n <= 1 else n * factorial(n - 1)  
  
print(factorial(5))
```

Output:

120

Analysis:

Clear iterative approach

No handling of negative numbers

Works well for valid inputs

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.

Promt:

Input: 153 → Output: Armstrong Number
Input: 370 → Output: Armstrong Number
Input: 123 → Output: Not an Armstrong Number

Write a Python function to check whether a given number is an Armstrong number.

Code:

```
def is_armstrong(n):
    digits = str(n)
    power = len(digits)
    total = 0

    for d in digits:
        total += int(d) ** power

    if total == n:
        return "Armstrong Number"
    else:
        return "Not an Armstrong Number"

print(is_armstrong(153))
```

Output:

Armstrong Number

Analysis:

Few-shot examples improve structure
Correct formula used
Does not handle negative or non-numeric input

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.

Promt:

Write an optimized Python program to classify a number as Prime, Composite, or Neither.

Constraints: Validate input, Use efficient logic, Handle edge cases like 0, 1, and negative numbers

Code:

```
def classify_number(n):  
    if not isinstance(n, int):  
        return "Invalid Input"  
  
    if n <= 1:  
        return "Neither"  
  
    for i in range(2, int(n ** 0.5) + 1):  
        if n % i == 0:  
            return "Composite"
```

```
return "Prime"  
  
print(classify_number(25))
```

Output:

Composite

Analysis:

Efficient \sqrt{n} optimization

Proper validation

Best quality among all prompting styles

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.

Task:

- Record the AI-generated code.
- Test the program with multiple inputs.
- Identify any missing conditions or inefficiencies in the logic.

Promt:

Write a Python function to check whether a given number is a perfect number.

Code:

```
def is_perfect(n):  
    total = 0
```

```
for i in range(1, n):
    if n % i == 0:
        total += i
return total == n

print(is_perfect(41))
```

Output:

False

Analysis:

Correct logic

Inefficient ($O(n)$)

No handling of $n \leq 0$

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

Task:

- Analyze how examples improve input handling and output clarity.

- Test the program with negative numbers and non-integer inputs

Promt:

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

Write a Python program to determine whether a number is even or odd with input validation.

Code:

```
def even_or_odd(n):
```

```
    if not isinstance(n, int):
```

```
        return "Invalid Input"
```

```
    if n % 2 == 0:
```

```
        return "Even"
```

```
    else:
```

```
        return "Odd"
```

```
print(even_or_odd(6))
```

Output:

Even

Analysis:

Few-shot improves validation

Clear output

Handles negative numbers correctly

