

## Assignment 3.1

2303A51250

I.Indu

Batch – 18

### 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.

**Prompt :**

#python program to check a number is a palindrome or not

**Code :**

```
palindrome.ipyn > ...
1  #python program to check a number is a palindrome or not
2  num = int(input("Enter a number: "))
3  original_num = num
4  reversed_num = 0
5  while num > 0:
6      digit = num % 10
7      reversed_num = reversed_num * 10 + digit
8      num = num // 10
9  if original_num == reversed_num:
10     print("The number is a palindrome.")
11 else:
12     print("The number is not a palindrome.")
```

**output**

```
PROBLEMS 58 OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/palindrome.ipyn
Enter a number: Traceback (most recent call last):
  File "c:\Users\user1\Desktop\python\palindrome.ipyn", line 2, in <module>
    num = int(input("Enter a number: "))
KeyboardInterrupt
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/palindrome.ipyn
Enter a number: 222
The number is a palindrome.
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/palindrome.ipyn
Enter a number: 121
The number is a palindrome.
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/palindrome.ipyn
Enter a number: 233
The number is not a palindrome.
PS C:\Users\user1\Desktop\python> s
```

## 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.

**Prompt :**

#program on a python function to compute the factorial of a number

**Code:**

```

factorial.ipyn > ...
1  #program on a python function to compute the factorial of a number
2  num = int(input("Enter a number to compute its factorial: "))
3  fact = 1
4  while num > 1:
5      fact *= num
6      num -= 1
7  print("Factorial is:", fact)
8
9

```

## Output:

```

PS C:\Users\user1\Desktop\python>
PS C:\Users\user1\Desktop\python>
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/user1/Desktop/python/factorial.ipyn
Enter a number to compute its factorial: 5
Factorial is: 120
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/user1/Desktop/python/factorial.ipyn
Enter a number to compute its factorial: 9
Factorial is: 362880
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/user1/Desktop/python/factorial.ipyn

```

## 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.

(Optional Extension)

### Prompt :

# Write 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

### Code :

```
# Write a Python function to check whether a given number is an Armstrong number. Armstrong numbers are those numbers that are
# Examples:
# Input: 153
# Output: Armstrong Number
# Input: 370
# Output: Armstrong Number
# Input: 123
# Output: Not an Armstrong Number

def is_armstrong(num):
    if num < 0:
        return "Invalid input"

    digits = str(num)
    power = len(digits)
    total = 0

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

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

try:
    n = int(input("Enter a number: "))
    print(is_armstrong(n))
except ValueError:
    print("Invalid input")
```

### Output:

```
PROBLEMS 58 OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/fact
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/arms
trong.ipy
Enter a number: 153
Armstrong Number
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/arms
trong.ipy
Enter a number: 370
Armstrong Number
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/arms
trong.ipy
Enter a number: 123
Not an Armstrong Number
PS C:\Users\user1\Desktop\python> 
```

## 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.

**Prompt :**

```
# Write an optimized Python program to classify a number as
# prime, composite, or neither.
# Print clear and correct output messages
# Avoid unnecessary loops
```

**Code :**

```

# Write an optimized Python program to classify a number as
# prime, composite, or neither.
# Print clear and correct output messages
# Avoid unnecessary loops
num = int(input("Enter a number to classify: "))
if num <= 1:
    print("Neither prime nor composite")
else:
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            print("Composite number")
            break
    else:
        print("Prime number")

```

## Output

```

Not an Armstrong Number
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/primecomposite.ipyn
Enter a number to classify: 2
Prime number
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/primecomposite.ipyn
Enter a number to classify: 4
Composite number
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/python/primecomposite.ipyn
Enter a number to classify: 0
Neither prime nor composite
PS C:\Users\user1\Desktop\python> 

```

## 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.

**Prompt :**



# Write a Python function that checks whether a given number is a perfect number.

**Code :**

```
perfectnumber.ipyn
1  # Write a Python function that checks whether a given number is a perfect number.
2  def is_perfect(num):
3      total = 0
4      for i in range(1, num):
5          if num % i == 0:
6              total += i
7      if total == num:
8          return True
9      else:
10         return False
11
12  num = int(input("Enter a number: "))
13  if is_perfect(num):
14      print("Perfect number")
15  else:
16      print("Not a perfect number")
```

**Output :**

```
Enter a number: 6
Perfect number
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktopnumber.ipyn
Enter a number: 32
Not a perfect number
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktopnumber.ipyn
Enter a number: 28
Perfect number
PS C:\Users\user1\Desktop\python> |
```

## 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.

**Prompt:**

# Write a Python program to determine whether a given number is even or odd.

# Examples:

# Input: 8

# Output: Even

# Input: 15

# Output: Odd

# Input: 0

# Output: Even

# The program should validate input and handle non-integer values gracefully.

**Code:**



```

evenodd.py
1  # Write a Python program to determine whether a given number is even or odd.
2  # Examples:
3  # Input: 8
4  # Output: Even
5  # Input: 15
6  # Output: Odd
7  # Input: 0
8  # Output: Even
9  # The program should validate input and handle non-integer values gracefully.
10 try:
11     num = int(input("Enter a number to check even or odd: "))
12     if num % 2 == 0:
13         print("Even")
14     else:
15         print("Odd")
16 except ValueError:
17     print("Invalid input. Please enter an integer value.")

```

## Output:

```

even.py
Enter a number to check even or odd: 8
Even
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/py
odd.py
Enter a number to check even or odd: 15
Odd
PS C:\Users\user1\Desktop\python> & C:/Users/user1/AppData/Local/Programs/Python/Python313/python.exe c:/Users/user1/Desktop/py
odd.py
Enter a number to check even or odd: 0
Even
PS C:\Users\user1\Desktop\python>

```

## Analysis:

- I gave clear and descriptive prompts to GitHub Copilot in VS Code to generate Python programs.
- When I included examples, instructions, and validation requirements, Copilot produced more accurate and well-structured code.
- This shows that providing proper context and clear requirements improves the quality of AI-generated code.