

## **Assignment 3**

**P Sree Varshini**

**2303A52336**

**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.

**Prompt:**

```
#Write a python program to find whether the given number is palindrome or not  
# Check if the string representation of the number is equal to its reverse  
# Take input from the user  
# Check and display whether the number is a palindrome
```

## Code and Output:

```
#Question-1
#Write a python program to find whether the given number is palindrome or not
# Check if the string representation of the number is equal to its reverse
# Take input from the user
# Check and display whether the number is a palindrome
def is_palindrome(number):
    str_num = str(number)
    return str_num == str_num[::-1]
user_input = int(input("Enter a number: "))
if is_palindrome(user_input):
    print(f"{user_input} is a palindrome.")
else:
    print(f"{user_input} is not a palindrome.")

Enter a number: 577
577 is not a palindrome.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number: 232
232 is a palindrome.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number: 111
111 is a palindrome.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst>
```

## Analysis:

Works correctly for basic positive numbers

Negative numbers fail due to string behavior, not real logic

No input type checking is done

Relies only on string conversion

Suitable only for simple or beginner-level tasks

## 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:

```
#Write a python program to find given number is factorial or not e.g input=5 output=120
```

```
# Base case: factorial of 0 or 1 is 1
```

```
# Recursive case
```

```
# Take input from the user
```

### Code and Output:

```
Assignment-3.py
14 #Question-2
15 #Write a python program to find given number is factorial or not e.g input=5 output=120
16 def factorial(n):
17     # Base case: factorial of 0 or 1 is 1
18     if n == 0 or n == 1:
19         return 1
20     else:
21         # Recursive case
22         return n * factorial(n - 1)
23 # Take input from the user
24 user_input = int(input("Enter a number to find its factorial: "))
25 result = factorial(user_input)
26 print(f"The factorial of {user_input} is: {result}")

Enter a number to find its factorial: 5
The factorial of 5 is: 120
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number to find its factorial: 6
The factorial of 6 is: 720
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number to find its factorial: 7
The factorial of 7 is: 5040
PS C:\Users\varsh\OneDrive\Desktop\AI Asst>
```

### Analysis:

One-shot code clearly handles the base case ( $0! = 1$ )

Zero-shot version misses explicit handling of zero

One-shot solution is easier to understand and more structured

One-shot result is mathematically more correct

Example helps the AI generate safer and clearer logic.

### **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.

#### **Prompt:**

```
#write a python code to check whether the given number is armstrong or not  
#Input: 153 → Output: Armstrong Number  
#Input: 370 → Output: Armstrong Number  
#Input: 123 → Output: Not an Armstrong Number  
  
# Calculate the sum of each digit raised to the power of the number of digits  
# Check if the sum of powers is equal to the original number  
# Take input from the user  
# Check and display whether the number is an Armstrong number
```

#### **Code and Output:**

```

#Question-3
#write a python code to check whether the given number is armstrong or not
#Input: 153 -> Output: Armstrong Number
#Input: 370 -> Output: Armstrong Number
#Input: 123 -> Output: Not an Armstrong Number
# Calculate the sum of each digit raised to the power of the number of digits
# Check if the sum of powers is equal to the original number
# Take input from the user
# Check and display whether the number is an Armstrong number
def is_armstrong(number):
    num_str = str(number)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
    return sum_of_powers == number
user_input = int(input("Enter a number: "))
if is_armstrong(user_input):
    print(f"{user_input} is an Armstrong Number.")
else:
    print(f"{user_input} is not an Armstrong Number.")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Enter a number: 153  
153 is an Armstrong Number.  
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"  
Enter a number: 370  
370 is an Armstrong Number.  
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"  
Enter a number: 434  
434 is not an Armstrong Number.  
PS C:\Users\varsh\OneDrive\Desktop\AI Asst>

Ln 45, Col 55 Spaces: 4 UTF-8 CRLF { } Python 3.13.7 (v) Go Live

## Analysis:

Giving examples helps the AI understand what kind of answer is expected.

Multiple examples make the code cleaner and more logical.

Showing both correct and incorrect cases avoids confusion.

Testing small numbers like 0 and 1 ensures the function works properly.

Checking wrong inputs makes the program safer and more reliable.

## 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 a python program to find the given number is prime,compositeor neither  
# Check if the number is less than 2 (neither prime nor composite)  
# Check for factors from 2 to the square root of the number  
# Take input from the user  
# Check and display whether the number is prime, composite, or neither
```

### Code and Output:

The screenshot shows the VS Code interface. The left sidebar has icons for Explorer, AI ASST, Search, Open, and Help. The main area shows the file 'Assignment-3.py' with the following code:

```
46  #Question-4
47  #Write a python program to find the given number is prime,compositeor neither
48  # Check if the number is less than 2 (neither prime nor composite)
49  # Check for factors from 2 to the square root of the number
50 def check_prime_composite(number):
51     if number < 2:
52         return "Neither prime nor composite"
53     for i in range(2, int(number**0.5) + 1):
54         if number % i == 0:
55             return "Composite"
56     return "Prime"
57 # Take input from the user
58 user_input = int(input("Enter a number: "))
59 # Check and display whether the number is prime, composite, or neither
60 result = check_prime_composite(user_input)
61 print(f"{user_input} is {result}.")
```

The terminal tab at the bottom shows the execution of the program:

```
Enter a number: 3
3 is Prime.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/va
rsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number: 18
18 is Composite.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/va
rsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number: 22
22 is Composite.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst>
```

At the bottom right, status bar shows: Ln 46, Col 12 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 (F) Go Live

### Analysis:

Clear instructions help the AI understand exactly what is needed.

Input validation avoids crashes from wrong inputs.

Efficient logic makes the program faster and smarter.

The AI performs better than with simple prompts.

Results are clearer and more reliable.

### 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 python program to find whether given input is perfect number or not
```

```
# Calculate the sum of divisors of the number
```

```
# Check if the sum of divisors is equal to the original number
```

```
# Take input from the user
```

```
# Check and display whether the number is a perfect number
```

**Code and Output:**

The screenshot shows a code editor interface with a dark theme. The left sidebar has icons for file operations, search, and navigation. The main area displays a Python file named 'Assignment-3.py'. The code defines a function 'is\_perfect\_number' that calculates the sum of divisors of a given number and checks if it equals the number itself. It also prompts the user for input and prints the result. The terminal below shows the execution of the script with two different inputs: 12 and 28, demonstrating that 6 is a perfect number while 12 is not.

```
62 #Question-5
63 #Write python program to find whether given input is perfect number or not
64 # Calculate the sum of divisors of the number
65 # Check if the sum of divisors is equal to the original number
66 def is_perfect_number(number):
67     if number < 1:
68         return False
69     divisors_sum = sum(i for i in range(1, number) if number % i == 0)
70     return divisors_sum == number
71 # Take input from the user
72 user_input = int(input("Enter a number: "))
73 # Check and display whether the number is a perfect number
74 if is_perfect_number(user_input):
75     print(f"{user_input} is a perfect number.")
76 else:
77     print(f"{user_input} is not a perfect number.")

Enter a number: 12
12 is not a perfect number.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number: 6
6 is a perfect number.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number: 28
28 is a perfect number.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst>
```

**Analysis:**

The AI works only with instructions, no examples.

The code usually works but may miss some cases.

Testing with different numbers shows if it's correct.

The logic may be slower without optimization.

Extra checks improve accuracy.

**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 code to check whether given input is even or odd number  
#e.g input=4 output=Even  
#e.g input=15 output=Odd  
# Take input from the user  
# Check and display whether the number is even or odd
```

**Code and Output:**

```
File Edit Selection View Go Run ... ← → Q AI Asst
EXPLORER ... Assignment-3.py ...
AI ASST Assignment-3.py ...
78 #Question-6
79 #Write a python code to check whether given input is even or odd number
80 #e.g input=4 output=Even
81 #e.g input=15 output=Odd
82 def check_even_odd(number):
83     if number % 2 == 0:
84         return "Even"
85     else:
86         return "Odd"
87 # Take input from the user
88 user_input = int(input("Enter a number: "))
89 # Check and display whether the number is even or odd
90 result = check_even_odd(user_input)
91 print(f"{user_input} is {result}.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter a number: 36
36 is Even.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number: 11
11 is Odd.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/varsh/OneDrive/Desktop/AI Asst/Assignment-3.py"
Enter a number: 52
52 is Even.
PS C:\Users\varsh\OneDrive\Desktop\AI Asst>
```

Ln 78, Col 12 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 (v) Go Live

## Analysis:

Examples make the task easy to understand.

The AI gives clear even or odd results.

Input validation improves with examples.

Negative numbers are handled properly.

Wrong inputs are easier to detect.