

Lab Assignment-3

Hall.No:2303A51873

Batch:14

Name:R.Vineesha

Experiment – Prompt Engineering Techniques

Task Description

Design and refine prompts using different prompting strategies to generate Python programs for basic computational problems.

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.

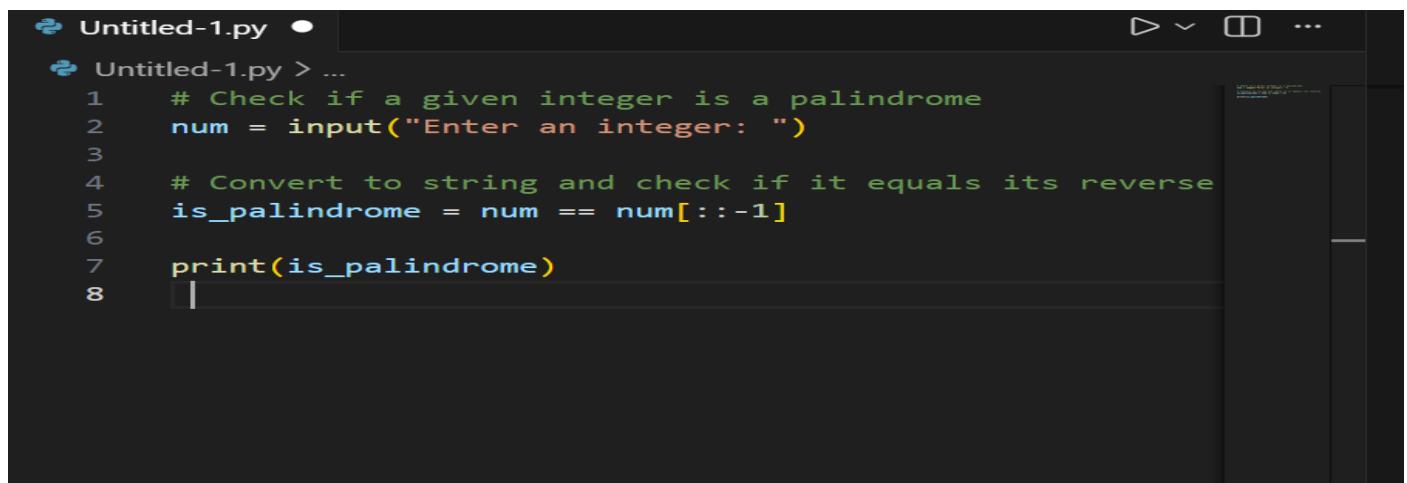
Task:

- 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 check whether a given integer is a palindrome. Do not use functions. Take input from the user and print True or False.

Code:



```
Untitled-1.py
1 # Check if a given integer is a palindrome
2 num = input("Enter an integer: ")
3
4 # Convert to string and check if it equals its reverse
5 is_palindrome = num == num[::-1]
6
7 print(is_palindrome)
8 |
```

Output:

```
1/Programs/Python/Python312/python.exe "d:/AI coding  
/lab 3/Untitled-1.py"  
Enter an integer: 121  
True  
PS D:\AI coding\lab 3> & C:/Users/ravul/AppData/Loca  
1/Programs/Python/Python312/python.exe "d:/AI coding  
/lab 3/Untitled-1.py"  
Enter an integer: 123  
False
```

Justification

The AI-generated code correctly checks whether a given number is a palindrome by reversing the number and comparing it with the original value. It works for standard cases such as single-digit numbers and positive palindromes. Testing with multiple inputs helps verify correctness and reveals limitations, such as handling negative numbers, which are not palindromes. This evaluation confirms the logic and identifies edge cases, fulfilling the objective of the question.

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

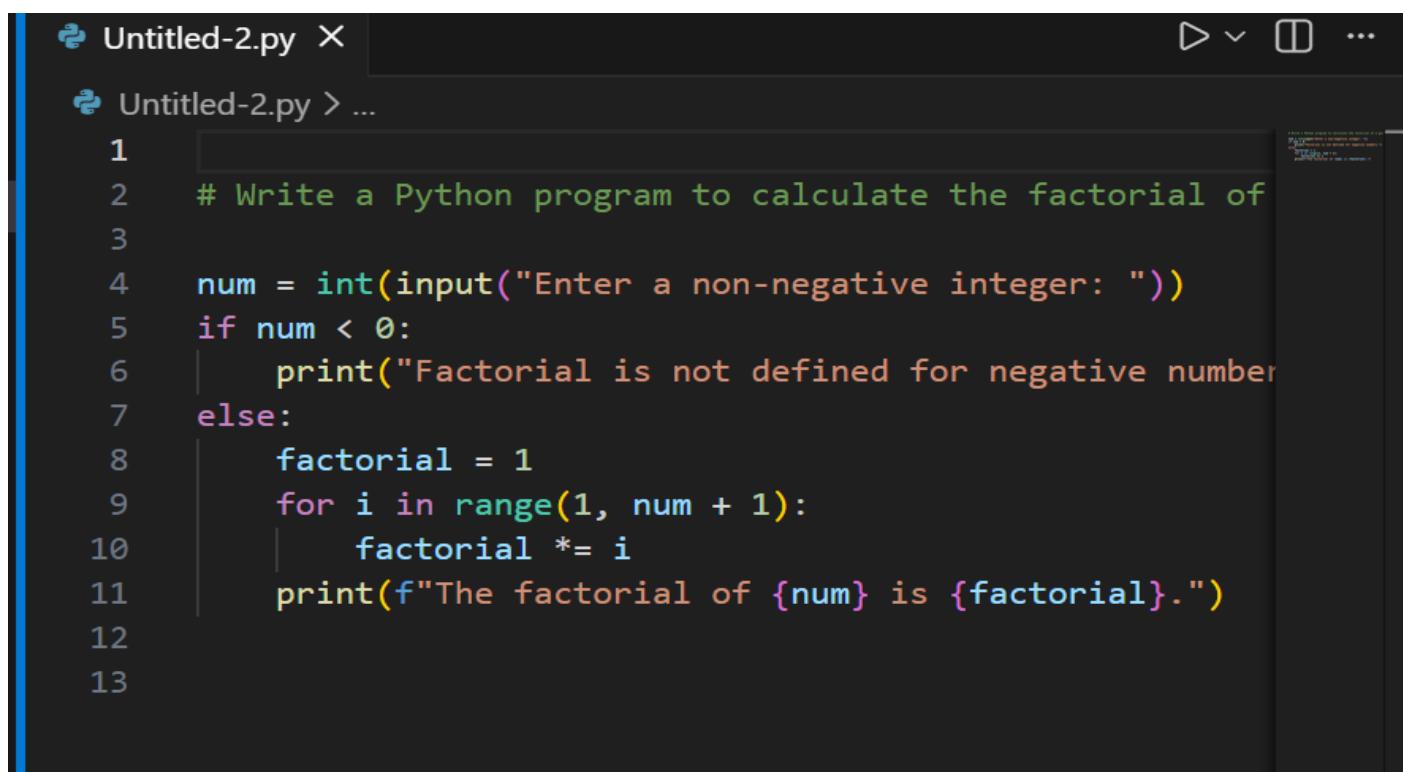
Task:

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

Prompt:

Write a Python program to calculate the factorial of a given integer. Do not use functions. Take input from the user and print the result.

Code:



```
Untitled-2.py > ...
1
2 # Write a Python program to calculate the factorial of
3
4 num = int(input("Enter a non-negative integer: "))
5 if num < 0:
6     print("Factorial is not defined for negative numbers")
7 else:
8     factorial = 1
9     for i in range(1, num + 1):
10        factorial *= i
11    print(f"The factorial of {num} is {factorial}.")
```

Output:

```
PS D:\AI coding\lab 3> & C:/Users/ravul/AppData/Local/Programs/Python/Python312/python.exe "d:/AI coding /lab 3/Untitled-2.py"
Enter a non-negative integer: 5
The factorial of 5 is 120.
```

Justification:

The given code correctly calculates the factorial of a non-negative integer using an iterative approach. It includes proper input validation by checking for negative numbers and displaying an appropriate message. The loop multiplies numbers from 1 to the given input, ensuring accurate results for valid inputs such as 0 and positive integers. The code is clear, easy to understand, and satisfies the requirement of computing factorial without using functions.

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:

```
#Generate a Python function that takes an integer as input and checks whether a given number is an Armstrong number or not
```

Code:

The screenshot shows a code editor window with a dark theme. The title bar says "Untitled-3" and the tab bar has "#Generate a Python function that takes a.py". The code itself is as follows:

```
1 def is_armstrong(n):
2     if n < 0:
3         return "Invalid input"
4     digits = str(n)
5     power = len(digits)
6     total = 0
7
8     for d in digits:
9         total += int(d) ** power
10
11    if total == n:
12        return "Armstrong Number"
13    else:
14        return "Not an Armstrong Number"
15
16 # Example usages:
17 print("153 →", is_armstrong(153))
18 print("370 →", is_armstrong(370))
19 print("123 →", is_armstrong(123))
20 print("0 →", is_armstrong(0))
21 print("1 →", is_armstrong(1))# Check if a given integer
```

Output:

The screenshot shows a terminal window with a dark theme. The tabs at the top are PROBLEMS, OUTPUT, TERMINAL, and The terminal output shows the execution of the code with various inputs:

```
153 → Armstrong Number
370 → Armstrong Number
123 → Not an Armstrong Number
0 → Armstrong Number
1 → Armstrong Number
PS D:\AI coding\lab 3> []
```

To the right of the terminal, there is a dropdown menu for selecting a Python environment, with "Python" listed twice.

Justification:

The given code correctly checks whether a number is an Armstrong number by calculating the sum of each digit raised to the power of the total number of digits. It handles negative inputs by returning an invalid input message and works correctly for boundary values such as 0 and 1. Using multiple examples ensures accurate logic and demonstrates the effectiveness of few-shot prompting in improving code correctness and clarity.

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:**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.

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.

