Assignment 3.1

Perumalla Sushwanth

batch 29

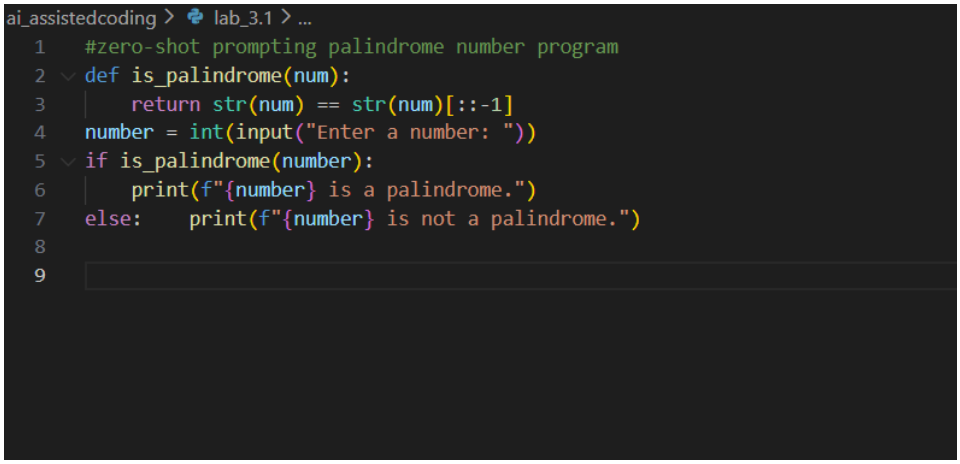
2303a51567

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.

code:



output

```
PS C:\Users\perum\OneDrive\Desktop\New folder>  & 'c:\Users\perum\AppData\Local\Programs\Python\Pyt
xtensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '49396' '--' 'C:\Use
stedcoding\lab_3.1'
Enter a number: 17
17 is not a palindrome.
```

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.

CODE :

```python
#one - shot prompting factorial program
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
number = int(input("Enter a number: "))
print(f"The factorial of {number} is {factorial(number)}.")
```

OUTPUT:

```
PS C:\Users\perum\OneDrive\Desktop\New folder>  c:; cd 'c:\Users\perum\OneDrive\Desktop\New folder' ; & 'C:\Users\perum\AppData\Local\Programs\P
ython\Python312\python.exe' 'c:\Users\perum\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61869' '-
-' 'C:\Users\perum\OneDrive\Desktop\New folder\ai_assistedcoding\lab_3.1'
Enter a number: 22
The factorial of 22 is 1124000727777607680000.
```

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

CODE:

```
    #few-shot prompting armstrong number check
def is_armstrong(num):
    num_str = str(num)
    num_digits = len(num_str)
    armstrong_sum = sum(int(digit) ** num_digits for digit in num_str)
    return armstrong_sum == num
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.")
```

OUTPUT:

```
ython\Python312\python.exe' 'c:\Users\perum\.vscode\extensions\ms-python.debugpy-2025.18.0
' 'C:\Users\perum\OneDrive\Desktop\New folder\ai_assistedcoding\lab_3.1'
Enter a number: 21
21 is not an Armstrong number.
```

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.

CODE:

```python
    #context -managed prompting optimized number check classification
def classify_number(num):
    if num < 0:
        return "Negative"
    elif num == 0:
        return "Zero"
    else:
        return "Positive"
number = int(input("Enter a number: "))
classification = classify_number(number)
print(f"{number} is classified as: {classification}.")
```

OUTPUT:

```
IndentationError: unexpected indent
PS C:\Users\perum\OneDrive\Desktop\New folder>  c:; cd 'c:\Users\perum\OneDrive\Desktop\New folder'; & 'c:\Users\perum\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\perum\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60391' '--' 'C:\Users\perum\OneDrive\Desktop\New folder\ai_assistedcoding\lab_3.1'
Enter a number: 23
23 is classified as: Positive.
```

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.

CODE:

```python
    #context -managed prompting optimized number check classification
def classify_number(num):
    if num < 0:
        return "Negative"
    elif num == 0:
        return "Zero"
    else:
        return "Positive"
number = int(input("Enter a number: "))
classification = classify_number(number)
print(f"{number} is classified as: {classification}.")

#zero-shot prompting perfect number check
def is_perfect(num):
    if num < 1:
        return False
    divisors_sum = sum(i for i in range(1, num) if num % i == 0)
    return divisors_sum == num
number = int(input("Enter a number: "))
if is_perfect(number):
    print(f"{number} is a perfect number.")
else:
    print(f"{number} is not a perfect number.")
```

OUTPUT:

```
PS C:\Users\perum\OneDrive\Desktop\New folder>  c:; cd 'c:\Users\perum\OneDrive\Desktop\New folder'; & 'c:\Users\perum\AppDat
ython\Python312\python.exe' 'c:\Users\perum\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\lau
' 'C:\Users\perum\OneDrive\Desktop\New folder\ai_assistedcoding\lab_3.1'
Enter a number: 33
33 is classified as: Positive.
```

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.

CODE:

```
    #FEW-SHOT PROMPTING EVEN OR ODD NUMBER CHECK WITH VALIDATION

def is_even_or_odd(num): ···
while True:
    try:
        number = int(input("Enter a number: "))
        break
    except ValueError:
        print("Invalid input. Please enter an integer.")
classification = is_even_or_odd(number)
print(f"{number} is classified as: {classification}.")
```

OUTPUT:

```
PS C:\Users\perum\OneDrive\Desktop\New folder>  c:; cd 'c:\Users\perum\OneDrive\Desktop\New folder'; & 'c:\Users\perum\AppData\Local\Programs\P
ython\Python312\python.exe' 'c:\Users\perum\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53427' '--
' 'C:\Users\perum\OneDrive\Desktop\New folder\ai_assistedcoding\lab_3.1'
Enter a number: 56
56 is classified as: Even.
PS C:\Users\perum\OneDrive\Desktop\New folder>
```