AI ASSISTED CODING ASSIGNMENT - 3.1

Lalith Kumar 2303a51761

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 Description**

  ➢ Record the AI-generated code.

  ➢ Test the code with multiple inputs.

  ➢ Identify any logical errors or missing edge-case handling.

```
# generate a Python function that checks whether a given number is a palindrome
def is_palindrome(n):
    if not isinstance(n, int):
        return "Invalid number"
    elif n < 0:
        return "Not a palindrome"
    else:
        original = n
        reversed_num = 0
        while n > 0:
            digit = n % 10
            reversed_num = reversed_num * 10 + digit
            n = n // 10
        if original == reversed_num:
            return f"{original} is a palindrome"
        else:
            return f"{original} is not a palindrome"
n=int(input("Enter a number: "))
print(is_palindrome(n))
```

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 Description**

➢ Compare the generated code with a zero-shot solution.

➢ Examine improvements in clarity and correctness.

```python
#n = 5
#print factorial of n is 120
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
num = int(input("Enter a number: "))
print("Factorial of", num, "is", factorial(num))
```

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.

- **Example:**

➢ Input: 153 → Output: Armstrong Number

➢ Input: 370 → Output: Armstrong Number

➢ Input: 123 → Output: Not an Armstrong Number

- **Task Description**

➢ Analyse how multiple examples influence code structure and accuracy.

➢ Test the function with boundary values and invalid inputs.

assain3.py > ...

```
n = 153
display Armstrong number
n = 370
display Armstrong number
n = 123
display not Armstrong number
'''

def is_armstrong(n):
    if not isinstance(n, int):
        return "Invalid number"
    elif n < 0:
        return "Not an Armstrong number"
    else:
        num_str = str(n)
        num_digits = len(num_str)
        sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
        if sum_of_powers == n:
            return f"{n} is an Armstrong number"
        else:
            return f"{n} is not an Armstrong number"
try:
    n = int(input("Enter a number: "))
    print(is_armstrong(n))
except ValueError:
    print("Invalid 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 Description

➢ Ensure proper input validation.

➢ Optimize the logic for efficiency.

➢ Compare the output with earlier prompting strategies.

```
readinput a number and
check the number is prime or composite or neither'''

def check_prime_composite(n):
    if not isinstance(n, int):
        return "Invalid number"
    elif n < 0:
        return "Invalid number"
    elif n == 0 or n == 1:
        return f"{n} is neither prime nor composite"
    else:
        for i in range(2, int(n**0.5) + 1):
            if n % i == 0:
                return f"{n} is a composite number"
        return f"{n} is a prime number"
try:
    n = int(input("Enter a number: "))
    print(check_prime_composite(n))
except ValueError:
    print("Invalid number")
```

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 Description

➢ Record the AI-generated code.

➢ Test the program with multiple inputs.

➢ Identify any missing conditions or inefficiencies in the logic.

```python
#generate a Python function that checks whether a given number is a perfect number

def is_perfect_number(n):
    if not isinstance(n, int):
        return "Invalid number"
    elif n < 1:
        return "Not a perfect number"
    else:
        divisors_sum = sum(i for i in range(1, n) if n % i == 0)
        if divisors_sum == n:
            return f"{n} is a perfect number"
        else:
            return f"{n} is not a perfect number"
try:
    n = int(input("Enter a number: "))
    print(is_perfect_number(n))
except ValueError:
    print("Invalid number")
```

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.

- **Example:**
  - ➢ Input: 8 → Output: Even
  - ➢ Input: 15 → Output: Odd
  - ➢ Input: 0 → Output: Even
- **Task Description**
  - ➢ Analyse how examples improve input handling and output clarity.
  - ➢ Test the program with negative numbers and non-integer inputs.

```
'''
num = 28
display even number
num = 7
display odd number
num = -5
display invalid number
num = sfsd
display invalid number
'''
def check_odd_even_extended(number):
    if not isinstance(number, int):
        return "Invalid number"
    elif number < 0:
        return "Invalid number"
    elif number % 2 == 0:
        return f"{number} is an even number"
    else:
        return f"{number} is an odd number"
try:
    number = int(input("Enter a number: "))
    print(check_odd_even_extended(number))
except ValueError:
    print("Invalid number")
```