ASSIGNMENT 3.5

J.Sujith

2303A51327

BATCH 20

Task 1: Zero-shot Prompt – Fibonacci Series Generator

Scenario

In this task, a **zero-shot prompting** technique is used. A single comment prompt is provided without any examples, instructing GitHub Copilot to generate a Python function that prints the first N Fibonacci numbers.

Prompt

```
# Write a Python function to print the first N Fibonacci numbers
```

Code

```python
# Task-1 : Fibonacci Series Generator

def fibonacci(n):
    # Generate Fibonacci series up to n terms
    a, b = 0, 1
    for _ in range(n):
        print(a, end=' ')
        a, b = b, a + b
    print()

# Get input from user
num_terms = int(input("Enter number of terms: "))
fibonacci(num_terms)
```

Sample Output

```
Enter number of terms: 7
0 1 1 2 3 5 8
```

Sample Output

```
Enter an email: test@gmail.com
Valid email
```

Sample Output

```
Enter a number: 123
Sum of digits: 6
```

Observations

- The model correctly inferred the logic of the Fibonacci series without any examples.
- The generated solution is simple, readable, and efficient.
- Zero-shot prompting works well for well-known problems with standard logic.
- Suitable for basic algorithmic tasks.

Task 2: One-shot Prompt – List Reversal Function

Scenario

In this task, a **one-shot prompting** technique is used by providing a single example along with the instruction. This helps Copilot generate a correct list reversal function.

Prompt

```
# Write a Python function to reverse a list
# Example: input [1, 2, 3] -> output [3, 2, 1]
```

Code

```python
# Task-2 : List Reversal Function

def reverse_list(lst):
    return lst[::-1]

user_list = [1, 2, 3, 4]
reversed_lst = reverse_list(user_list)
print("Reversed list:", reversed_lst)
```

Output

```
Reversed list: [4, 3, 2, 1]
```

Observations

- One-shot example clearly guides the model toward the expected logic.
- Output confirms correct reversal of the list.
- Python slicing provides an efficient solution.

Task 3: Few-shot Prompt – Even or Odd Checker

Scenario

Few-shot prompting is used by giving multiple examples to guide the model logic.

Code

```
# Task-3 : Even or Odd Checker

def check_even_odd(n):
    if n % 2 == 0:
        return "Even"
    else:
        return "Odd"

num = 7
print("Number:", num)
print("Result:", check_even_odd(num))
```

Output

```
Number: 7
Result: Odd
```

Observations

- Few-shot prompting improves accuracy for conditional problems.
- Output matches expected logical behavior.

Task 4: Zero-shot vs Few-shot – Factorial Program

Code

```
# Task-4 : Factorial using loop

def factorial(n):
    fact = 1
    for i in range(1, n+1):
        fact *= i
    return fact

num = 5
print("Factorial of", num, "is", factorial(num))
```

Output

```
Factorial of 5 is 120
```

Observations

- Few-shot prompting produces more structured code than zero-shot.
- Output verifies correctness of factorial logic.

Task 5: Prompt Tuning – Sum of Natural Numbers

Code

```python
# Task-5 : Sum of first n natural numbers

def sum_natural(n):
    return n * (n + 1) // 2

num = 10
print("Sum of first", num, "natural numbers:", sum_natural(num))
```

Output

```
Sum of first 10 natural numbers: 55
```

Observations

- Prompt tuning helps generate optimized mathematical solutions.
- Output confirms formula-based approach is correct.