

## ASSIGNMENT-5

### Task-1: Zero-shot Prompt – Fibonacci Series Generator

**Prompt:** #write a python function to print the first N Fibonacci numbers.

**Code and Output:**

```
❸ 3.4(fibonacci).py > ...
1  #write a python function to print the first N Fibonacci numbers.
2  def fibonacci_sequence(n):
3      fib_sequence = []
4      a, b = 0, 1
5      for _ in range(n):
6          fib_sequence.append(a)
7          a, b = b, a + b
8  return fib_sequence
9 # Get user input
10 N = int(input("Enter the number of Fibonacci numbers to generate: "))
11 # Generate and print the Fibonacci sequence
12 fib_numbers = fibonacci_sequence(N)
13 print(fib_numbers)
```

```
Enter the number of Fibonacci numbers to generate: 7
[0, 1, 1, 2, 3, 5, 8]
```

**Justification:**

- Copilot correctly understood the task without any example
- Generated a valid Fibonacci logic using iteration
- Shows that zero-shot prompting works well for common problems

### Task-2: One-shot Prompt – List Reversal Function

**Prompt:** #write a python function to reverse a list

**#Example: Input:** [3,2,1] **Output:** [1,2,3]

**Code and Output:**

```
3.4(reverseList).py > ...
1 #write a python function to reverse a list
2 #Example: Input: [3,2,1] Output: [1,2,3]
3 def reverse_list(lst):
4     return lst[::-1]
5 # Get user input
6 user_input = input("Enter a list of numbers separated by commas: ")
7 # Convert the input string to a list of integers
8 input_list = [int(x.strip()) for x in user_input.split(',')]
9 # Reverse the list and print the result
10 reversed_list = reverse_list(input_list)
11 print(reversed_list)
```

```
Enter a list of numbers separated by commas: 3,2,1
[1, 2, 3]
```

### Justification:

- Adding one example improved accuracy
- Copilot used Python slicing, which is clean and efficient
- One-shot prompts guide Copilot toward better solutions

## Task-3: String Pattern Matching

### Prompt:

```
#write a python function o check if a string starts with a capital letter and ends with
a period.
```

```
#Example: Input: "Coding." Output: True
```

```
#Example: Input: "coding." Output: False
```

```
#Example: Input: "Coding" Output: False
```

### Code and Output:

```

❷ 3.4(stringPattern).py > ...
1  #write a python function o check if a string starts with a capital letter and ends with a period.
2  #Example: Input: "Coding." Output: True
3  #Example: Input: "coding." Output: False
4  #Example: Input: "Coding" Output: False
5  import re
6  def check_string_pattern(s):
7      pattern = r'^[A-Z].*\.$'
8      if re.match(pattern, s):
9          return True
10     else:
11         return False
12 # Get user input
13 user_input = input("Enter a string: ")
14 # Check the string pattern and print the result
15 result = check_string_pattern(user_input)
16 print(result)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/AI(vs)/stringPattern.py"

Enter a string: Coding.

True

PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/AI(vs)/stringPattern.py"

Enter a string: coding

False

PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/AI(vs)/stringPattern.py"

Enter a string: Coding

False

PS C:\Users\akhil\OneDrive\Documents\AI(vs)>

### Justification:

- Multiple examples helped Copilot understand the pattern clearly
- Few-shot prompting produces more accurate and specific logic

### Task-4: Zero-shot vs Few-shot – Email Validator

**Prompt for zero-shot:** write a python function to check an email validation.

### Code and Output:

```
3.4(emailValidation.py) > ...
1  #write a python function to check an email validation.
2  def is_valid_email(email):
3      import re
4      pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$
5      if re.match(pattern, email):
6          return True
7      else:
8          return False
9  # Get user input
10 user_input = input("Enter an email address: ")
11 # Check email validation and print the result
12 validation_result = is_valid_email(user_input)
13 print(validation_result)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + ▾
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/mailValidation.py"
Enter an email address: coding@gmail.com
True
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/mailValidation.py"
Enter an email address: coding@
False
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/mailValidation.py"
Enter an email address: gmail.com
False
PS C:\Users\akhil\OneDrive\Documents\AI(vs)>
```

**Prompt for few-shot:** #write a python function to check an email validation.

#Example: Input: "coding@gmail.com" Output: True

#Example: Input: "codinggmail.com" Output: False

#Example: Input: "coding@gmailcom" Output: False

### Code and Output:

```
3.4(emailValidation.py) > ...
1  #write a python function to check an email validation.
2  #Example: Input: "coding@gmail.com" Output: True
3  #Example: Input: "codinggmail.com" Output: False
4  #Example: Input: "coding@gmailcom" Output: False
5  import re
6  def validate_email(email):
7      pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$
8      if re.match(pattern, email):
9          return True
10     else:
11         return False
12 # Get user input
13 user_input = input("Enter an email address: ")
14 # Validate the email and print the result
15 is_valid = validate_email(user_input)
16 print(is_valid)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + ▾
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/mailValidation.py"
Enter an email address: coding@gmail.com
True
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/mailValidation.py"
Enter an email address: codinggmail.com
False
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/mailValidation.py"
Enter an email address: coding
False
PS C:\Users\akhil\OneDrive\Documents\AI(vs)>
```

### **Justification:**

- Zero-shot prompting gives basic and incomplete validation because no examples are provided.
- Few-shot prompting gives better logic and more accurate results by checking username and domain using examples.

Zero-shot -- Basic, incomplete validation

Few-shot -- Better logic, checks username & domain

### **Task-5: Prompt Tuning – Summing Digits of a Number**

**Prompt1:** write a python function that returns the sum of digits of a number.

**Code and Output:**

The screenshot shows a terminal window with the following content:

```
3.4(sumOfDigits).py > ...
1 # write a python function that returns the sum of digits of a number.
2 def sum_of_digits(n):
3     total = 0
4     for digit in str(n):
5         total += int(digit)
6     return total
7 # Get user input
8 user_input = int(input("Enter a number: "))
9 # calculate the sum of digits and print the result
10 result = sum_of_digits(user_input)
11 print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/AI(vs)/sumOfDigits.py"
Enter a number: 15
6
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/AI(vs)/sumOfDigits.py"
Enter a number: 157
13
PS C:\Users\akhil\OneDrive\Documents\AI(vs)>
```

0

**Prompt2:** #generate a python function to calculate sum\_of\_digits

#Example Input: sum\_of\_digits(123) → 6

**Code and Output:**

The screenshot shows a Python script named `sum.py` in a code editor. The code defines a function `sum_of_digits` that calculates the sum of digits of a given number. It prompts the user for input, calculates the result, and prints it. The terminal below shows the script being run and the output of two test cases.

```
3.4(sum.py) > ...
1 #generate a python function to calculate sum_of_digits
2 #Example Input: sum_of_digits(123) → 6
3 def sum_of_digits(n):
4     total = 0
5     for digit in str(n):
6         total += int(digit)
7     return total
8 # Get user input
9 user_input = int(input("Enter a number: "))
10 # Calculate the sum of digits and print the result
11 result = sum_of_digits(user_input)
12 print(result)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + ▾ □ ▶

PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/sum.py"
Enter a number: 12
3
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/sum.py"
Enter a number: 157
13
PS C:\Users\akhil\OneDrive\Documents\AI(vs)>
```

### Justification:

- Prompt2 produced cleaner and optimized code
- Example guided Copilot to use Pythonic one-line solution
- Prompt tuning improves code quality and readability