ASSIGNMENT – 9.2

B.Sri Laxmi Gayathri

2303A52033

Batch – 38

Task Description -1 (Documentation – Function Summary Generation)

Task:

Use AI to generate concise functional summaries for each Python

function in a given script.

Instructions:

• Provide a Python script to the AI.

• Ask the AI to write a short summary describing the purpose of

each function.

• Ensure summaries are brief and technically accurate.

• Do not include code implementation details.

Expected Output -1:

A Python script where each function contains a clear and concise

summary explaining its purpose.

PROMPT:

Add the summary as a docstring inside each function and return the updated script without explaining the code.

```python
def add(a, b):
    """Calculates the sum of two numerical inputs."""
    return a + b

def is_even(num):
    """Checks if a given number is even."""
    return num % 2 == 0

def find_max(numbers):
    """Finds the maximum value in a list of numbers."""
    return max(numbers)

def greet(name):
    """Generates a greeting message for a given name."""
    return f"Hello, {name}!"
```

EXPLANATION:

AI generated short docstrings that explain the purpose of each function in the script. The summaries describe what the function does, not the internal logic or steps. Simple technical language is used so the meaning is clear and easy to understand. This makes the code more readable and helps others quickly understand the functions.

Task Description -2 (Documentation – Logical Explanation for Conditions and Loops)

Task:

Use AI to document the logic behind conditional statements and loops in a Python program.

Instructions:

• Provide a Python program without comments.

• Instruct AI to explain only decision-making logic and loop behavior.

• Skip basic syntax explanations.

Expected Output -2:

Python code with clear explanations describing the logic of conditions and loops

PROMPT:

Read the Python program and add comments explaining the logic of conditional statements and loops.
Return the updated code with comments, without explaining basic syntax.

OUTPUT:

```
numbers = [3, 8, 5, 10, 2]

# Iterate through each number in the 'numbers' list
for num in numbers:
    # Check if the current number is even
    if num % 2 == 0:
        # If the number is even, print it along with the message 'is even'
        print(num, "is even")
    # If the number is not even (i.e., it's odd)
    else:
        # Print the number along with the message 'is odd'
        print(num, "is odd")
```

EXPLANATION:

Read the Python program and add comments explaining the logic of conditional statements and loops.
Return the updated code with comments, without explaining basic syntax**.**

Task Description -3 (Documentation – File-Level Overview)

Task:

Use AI to generate a high-level overview describing the functionality of

an entire Python file.

Instructions:

• Provide the complete Python file to AI.

• Ask AI to write a brief overview summarizing the file's purpose

and functionality.

• Place the overview at the top of the file.

Expected Output -3:

A Python file with a clear and concise file-level overview at the

beginning.

PROMPT:

Read the Python file and write a brief overview summarizing its purpose and functionality.
Add the overview as a comment at the top and return the updated file.

OUTPUT:

```
numbers = [3, 8, 5, 10, 2]

# Iterate through each number in the 'numbers' list
for num in numbers:
    # Check if the current number is even
    if num % 2 == 0:
        # If the number is even, print it along with the message 'is even'
        print(num, "is even")
    # If the number is not even (i.e., it's odd)
    else:
        # Print the number along with the message 'is odd'
        print(num, "is odd")
```

EXPLANATION:

AI generates a file-level summary that explains the overall purpose of the program. The overview is placed at the top of the file for documentation. This helps readers quickly understand what the Python file does.

Task Description -4 (Documentation – Refine Existing Documentation)

Task:

Use AI to improve clarity and consistency of existing documentation in

Python code.

Instructions:

• Provide Python code containing basic or unclear comments.

• Ask AI to rewrite the documentation to improve clarity and

consistency.

• Ensure technical meaning remains unchanged.

Expected Output -4:

Python code with refined and improved documentation that is clear and consistent.

PROMPT:

Read the Python code and rewrite the existing comments to improve clarity and consistency without changing their meaning. Return the updated code with refined documentation.

OUTPUT:

```
def calculate_average(numbers):
    # Calculates the average of a list of numbers.
    total = sum(numbers)
    # Divides the total by the count of numbers to get the average.
    return total / len(numbers)

# Determines if a score indicates a 'Pass' or 'Fail'.
def check_result(score):
    # If the score is 40 or greater, it's a 'Pass'.
    if score >= 40:
        return "Pass"
    else:
        return "Fail"
```

EXPLANATION:

AI rewrites unclear comments to make them more professional and consistent. The technical meaning remains the same, but the wording becomes clearer. This improves overall code readability and documentation quality.

Task Description -5 (Documentation – Prompt Detail Impact Study)

Task:

Study the impact of prompt detail on AI-generated documentation quality.

Instructions:

Create two prompts: one brief and one detailed.

• Use both prompts to document the same Python function.

• Compare the generated outputs.

Expected Output -5:

A comparison table highlighting differences in completeness, clarity, and

accuracy of documentation.

PROMPT 1:

Write a short docstring for the following Python function**.**

OUTPUT:

```
def calculate_discount(price, discount_percent):
    """Calculates the final price after applying a discount.

    Args:
        price (float): The original price of the item.
        discount_percent (float): The discount percentage to apply.

    Returns:
        float: The final price after the discount.
    """
    final_price = price - (price * discount_percent / 100)
    return final_price
```

PROMPT 2:

Write a clear and concise docstring for the given Python function explaining its purpose. Include a brief description of the parameters and the return value.

OUTPUT:

```
def calculate_discount(price, discount_percent):
    """Calculates the final price of an item after applying a percentage discount.

    Args:
        price (float): The original price of the item before any discount.
        discount_percent (float): The percentage of discount to be applied (e.g., 10 for 10%).

    Returns:
        float: The calculated final price after the discount has been applied.
    """
    final_price = price - (price * discount_percent / 100)
    return final_price
```

EXPLANATION:

The brief prompt produces minimal documentation with limited detail. The detailed prompt generates clearer, more structured, and complete documentation. This shows that more specific prompts improve the quality and usefulness of AI-generated documentation.

COMPARSION TABLE:

Aspect	Brief Prompt Output	Detailed Prompt Output
Completeness	Very short, only states purpose	Includes purpose, parameters, and return value
Clarity	Basic and minimal	Clear and well-structured
Accuracy	Correct but limited detail	Correct and more informative