

# AI ASSISTANT CODING ASSIGNMENT -9.2

---

**NAME : MOGILI KEERTHI**

**ROLL NO : 2303A51427**

**BATCH NO : 21**

---

## **Lab 9 – Documentation Generation: Automatic Documentation and Code Comments**

### **Lab Objectives**

- To use AI-assisted coding tools for generating Python documentation and code comments.
- To apply zero-shot, few-shot, and context-based prompt engineering for documentation creation.
- To practice generating and refining docstrings, inline comments, and module-level documentation.
- To compare outputs from different prompting styles for quality analysis.

### **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 a short functional summary as a docstring for every function in this file. The summary must explain the function purpose only, without describing logic or implementation steps.

**Code:**

```
'''Add a short functional summary as a docstring for every function in this file.
The summary must explain the function purpose only, without describing logic
or implementation steps.'''
def is_prime(num):
    """Check if a number is prime."""
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True
number = int(input("Enter a number: "))
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

**Output:**

- PS C:\Users\keert> python -u "C:\Users\keert\AppData\Local\Programs\Python\Python310\test.py"
Enter a number: 1
1 is not a prime number.
- PS C:\Users\keert> python -u "C:\Users\keert\AppData\Local\Programs\Python\Python310\test.py"
Enter a number: 5
5 is a prime number.
- PS C:\Users\keert> python -u "C:\Users\keert\AppData\Local\Programs\Python\Python310\test.py"
Enter a number: 2
2 is a prime number.

**Observation:**

The AI-generated function summaries enhance code readability by clearly explaining the purpose and functionality of each method. These summaries enable developers to quickly understand the overall structure and flow of the program without needing to examine the implementation in detail. This approach also facilitates easier maintenance, improves collaboration among team members, and ensures consistency in documentation across the project.

**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:

Add a brief explanation of the logic behind the each condition and loop statements.  
Explain only decision - making and loop statements

### Code:

```
'''Add a brief explanation of the logic behind the each condition and loop statements. Explain only decision - making and loop statements'''
def is_prime(num):
    """Check if a number is prime."""
    if num <= 1:
        return False # 0 and 1 are not prime numbers
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False # If num is divisible by any number other than 1 and itself, it's not prime
    return True # If no divisors are found, num is prime
number = int(input("Enter a number: "))
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

### Output:

```
● PS C:\Users\keert> python -u "C:\Users\keert\AppData\Local\Temp\tempCodeRunnerFile.python"
Enter a number: 0
0 is not a prime number.
PS C:\Users\keert> python -u "C:\Users\keert\AppData\Local\Temp\tempCodeRunnerFile.python"
Enter a number: -2
-2 is not a prime number.
PS C:\Users\keert> python -u "C:\Users\keert\AppData\Local\Temp\tempCodeRunnerFile.python"
Enter a number: 2
2 is a prime number.
PS C:\Users\keert> []
```

### **Observation:**

AI-generated explanations for conditional statements and loops improve clarity by clearly outlining the program's decision-making logic and iteration processes. These descriptive comments make it easier to understand how different execution paths are controlled within the code. Consequently, this enhances code comprehension, simplifies debugging, and supports more efficient program analysis.

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

Task:

Use AI to generate 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:**

Generate a file summarizing the overall functionality of the script, including the purpose of the main function and how it interacts with the user.

**Code:**

```
'''Generate a file summarizing the overall functionality of the script, including the
purpose of the main function and how it interacts with the user.'''
def is_prime(num):
    """Check if a number is prime."""
    if num <= 1:
        return False # 0 and 1 are not prime numbers
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False # If num is divisible by any number other than 1 and itself, it's not prime
    return True # If no divisors are found, num is prime
number = int(input("Enter a number: "))
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

**Output:**

```
Enter a number: -2
-2 is not a prime number.
PS C:\Users\keert> python -u "C:\Users\keert\A
Enter a number: 2
2 is a prime number.
PS C:\Users\keert> python -u "C:\Users\keert\A
Enter a number: 11
11 is a prime number.
● Enter a number: -2python -u "C:\Users\keert\Ap
```

### Script Summary:

This script defines a function named `is_prime` that determines whether a given number is prime. The main function prompts the user to enter a number and then passes that input to the `is_prime` function for evaluation. Based on the function's return value, the program displays a message indicating whether the number is prime or not. The script uses fundamental control flow constructs, such as conditional statements and loops, to perform the primality check and provide appropriate feedback to the user.

### Observation:

The AI effectively generated a clear and concise file-level summary outlining the overall objective and functionality of the Python program. The summary was positioned at the beginning of the file and described the program from a high-level perspective without delving into implementation specifics. This enhanced the code's readability and enabled users to quickly grasp the program's purpose.

### 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:**

Rewrite the existing comments to improve clarity and consistency while keeping the original meaning unchanged. Do not modify the program logic.

## Code:

```
'''Rewrite the existing comments to improve clarity and consistency while
keeping the original meaning unchanged. Do not modify the program logic.'''
def is_prime(num):
    """Check if a number is prime."""
    if num <= 1:
        return False # 0 and 1 are not considered prime numbers
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False # If num is divisible by any number other than 1 and itself, it is not prime
    return True # If no divisors are found, num is prime
number = int(input("Enter a number: "))
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.)
```

## Output:

```
PS C:\Users\keert> python -u "C:\Users\keert\Appl
Enter a number: -987
-987 is not a prime number.
PS C:\Users\keert> python -u "C:\Users\keert\Appl
Enter a number: 65
65 is not a prime number.
PS C:\Users\keert> 
```

## Observation:

The AI enhanced the clarity and uniformity of the existing comments while keeping the program's functionality unchanged. The revised documentation offered more precise and structured explanations without modifying the original technical intent. This improvement increased the overall readability of the code and made it easier to understand, review, and maintain.

## 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.

### Brief Prompt:

```
'''Write a short one-line documentation comment describing the purpose of the
following function.'''
def factorial(n):
    """Calculate the factorial of a number."""
    if n == 0:
        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\keert> python -u "C:\Users\keert\Ap
Enter a number: 5
The factorial of 5 is 120.
PS C:\Users\keert> python -u "C:\Users\keert\Ap
Enter a number: 1
The factorial of 1 is 1.
● PS C:\Users\keert> python -u "C:\Users\keert\Ap
Enter a number: 0
The factorial of 0 is 1.
PS C:\Users\keert> █
```

### Detailed Prompt:

Write a clear 3–5 line documentation comment for the following function describing its purpose, inputs, outputs, and overall behavior.

### Code:

```

'''Write a clear 3-5 line documentation comment for the following function
describing its purpose, inputs, outputs, and overall behavior.'''
def factorial(n):
    """
    Calculate the factorial of a non-negative integer n.

    The factorial of a non-negative integer n is the product of all positive integers less than or equal to n.
    It is denoted as n!. For example, 5! = 5 * 4 * 3 * 2 * 1 = 120.
    The function uses recursion to compute the factorial, where the base case is when n is 0, returning 1.

    Parameters:
    n (int): A non-negative integer for which the factorial is to be calculated.

    Returns:
    int: The factorial of the input number n.
    """
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
number = int(input("Enter a number: "))
print(f"The factorial of {number} is {factorial(number)}.")

```

## Output:

```

file.py
Enter a number: 0
The factorial of 0 is 1.
PS C:\Users\keert> python -u "C:\Users\keert\AppData\Local\Temp\tempCodeRunnerFile.python"
Enter a number: 5
The factorial of 5 is 120.
PS C:\Users\keert> python -u "C:\Users\keert\AppData\Local\Temp\tempCodeRunnerFile.python"
Enter a number: 54
The factorial of 54 is 2308436973392413804720927426830275810832785645718079411322880000000000000000.
PS C:\Users\keert>

```

## Observation:

Criteria	Brief Prompt Output	Detailed Prompt Output
<b>Completeness</b>	States only the basic purpose of the function.	Provides a comprehensive description including purpose, input parameters, return values, and behavioral details.
<b>Clarity</b>	Clear but very concise and minimal in explanation.	More detailed, well-structured, and easier to understand.
<b>Accuracy</b>	Technically correct but limited in depth and context.	Highly accurate with complete technical context and detailed parameter information.