AI - Assisted Coding-Assignment -9

HTNO:2303A52228
Batch:37

Lab 9 – Documentation Generation: Automatic Documentation and Code Comments

Task Description-1
Documentation – Function Summary Generation
Expected Output:
A Python script where each function contains a clear and concise
summary explaining its purpose.
Prompt:
Generate a short summary for each function,the summary should clearly explain what the function does, not how it works.Keep each summary to one line and ensure it is technically correct.
CODE :

doc.py > is_even

```python
def is_even(n):
    """
    Check if a number is even.
    Args:
        n: An integer to check.
    Returns:
        bool: True if the number is even, False otherwise.
    """
    return n % 2 == 0
def find_max(numbers):
    """
    Find and return the maximum value in a list of numbers.
    Args:
        numbers (list): A list of numeric values.
    Returns:
        The maximum value from the input list.
    Raises:
        IndexError: If the list is empty.
    Example:
        >>> find_max([3, 1, 4, 1, 5, 9])
        9
    """
    max_val = numbers[0]
    for num in numbers:
        if num > max_val:
            max_val = num
    return max_val

```

Explanation:
In this task, AI is used to automatically generate short summaries for each function in a Python program. The goal is to describe what the function does rather than how it works internally. These summaries improve code readability and make it easier for developers to quickly understand the purpose of each function without analyzing the full implementation.

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

Expected Output:
Python code with clear explanations describing the logic of conditions
and loops
Prompt:
Generate documentation for the following Python program that clearly explains the logic behind its conditional statements and loops. The documentation should focus on how decisions are made and how repetition is controlled in the program.
CODE :

doc.py > ...

```python
#generate docstr for the following function which explains
# the conditional statements and loops clearly.

def count_even_odd(numbers):
    """
    Count the number of even and odd numbers in a list.
    Args:
        numbers (list): A list of integers.
    Returns:
        tuple: A tuple containing two integers, the count of e
    Example:
        >>> count_even_odd([1, 2, 3, 4, 5])
        (2, 3)
    """
    even = 0
    odd = 0
    for num in numbers:
        if num % 2 == 0:
            even += 1
        else:
            odd += 1
    return even, odd
```

Explanation:
In this task, AI is used to automatically generate documentation that explains the reasoning behind conditional statements and loops in a Python program. Instead of describing syntax, the focus is on understanding how the program makes decisions and how repetition is controlled through loops. This type of documentation helps readers quickly grasp the program's logical flow, making the code easier to understand, debug, and maintain.

Task Description-3

Documentation – File-Level Overview.

Expected Output:

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

Prompt:

Generate a high-level overview for the following Python file that summarizes its overall purpose and functionality. The overview should describe what the program does at a general level without explaining implementation details.

CODE :

doc.py > ...

```python
#write a brief overview summarizing the purpose
#of this code and its functionality.
# This code defines two functions: `add` and `average`.
# The `add` function takes two arguments and returns their sum
#  The `average` function takes a list of numbers as an argume
# by the length of the list. Finally, the code demonstrates th
def add(a, b):
    return a + b

def average(nums):
    return sum(nums) / len(nums)

print(average([2,4,6,8]))
```

Explanation:

In this task, AI is used to create a file-level overview that summarizes the purpose of an entire Python program. Instead of documenting individual functions or statements, the AI provides a general description of what the program does and what problem it solves. This overview is placed at the beginning of the file so that anyone reading the code can quickly understand its objective and functionality.

Task Description-4

Documentation – Refine Existing Documentation

Expected Output:

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

Prompt:

Review the following Python code and refine the existing comments and documentation to improve clarity, consistency, and readability. Rewrite the documentation so that it is clear and professionally written while preserving the original technical meaning.

CODE :

doc.py > ...

```python
#rewrite the documentation to improve clarity andconsistency
# This function calculates the area of a circle given its radi
# Parameters:
# r (float): The radius of the circle.
# Returns:
# float: The area of the circle.
# Example usage:
# area(5)  # returns 78.5
# Note: The value of pi is approximated as 3.14 in this functi
# This loop prints the numbers from 0 to 4.
# The range function generates a sequence of numbers, starting
# Example usage:
# for i in range(5):
#   print(i)  # prints 0, 1, 2, 3
def area(r):
    # this gives area
    return 3.14 * r * r
# loop for printing numbers
for i in range(5):
    # print i
    print(i)
```

Explanation:
In this task, AI is used to enhance the quality of existing documentation in a Python program. Instead of generating new explanations, the AI rewrites unclear or inconsistent comments so they become easier to understand and more uniform in style. The goal is to improve readability while keeping the original technical meaning unchanged.

Task Description-5
Documentation – Prompt Detail Impact Study.
Expected Output:
A comparison table highlighting differences in completeness, clarity, and accuracy of documentation.
Prompt:
Generate a clear docstring for this Python function explaining its purpose, input parameter, return value, and behavior. Keep it concise but technically accurate. Do not include implementation details.
CODE :

doc.py > factorial

```python
#Add documentation for this Python function
# This function calculates the factorial of a given number n.
# Parameters:
# n (int): The number for which factorial is to be calculated.
# Returns:
# int: The factorial of n.
# Example usage:
# factorial(5)  # returns 120
def factorial(n):
    if n == 0:
        return 1
    result = 1
    for i in range(1, n+1):
        result *= i
    return result
#Generate a clear docstring for this Python function explainir
def factorial(n):
    """
    Calculate the factorial of a given non-negative integer n.
    Parameters:
    n (int): A non-negative integer for which the factorial is
    Returns:
    int: The factorial of n, defined as the product of all pos
        For n = 0, the factorial is defined as 1.
    Raises:
    ValueError: If n is a negative integer.
        Example usage:
    >>> factorial(5)
    120
    """
```

Explanation:
This task demonstrates how the level of detail in prompts affects the quality of AI-generated documentation. A brief prompt produces minimal and less informative documentation, while a detailed prompt results in structured, clear, and complete explanations.