ASSIGNMENT - 9

Thumma Hasini | 2303A52076 | Batch - 37

Task -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.

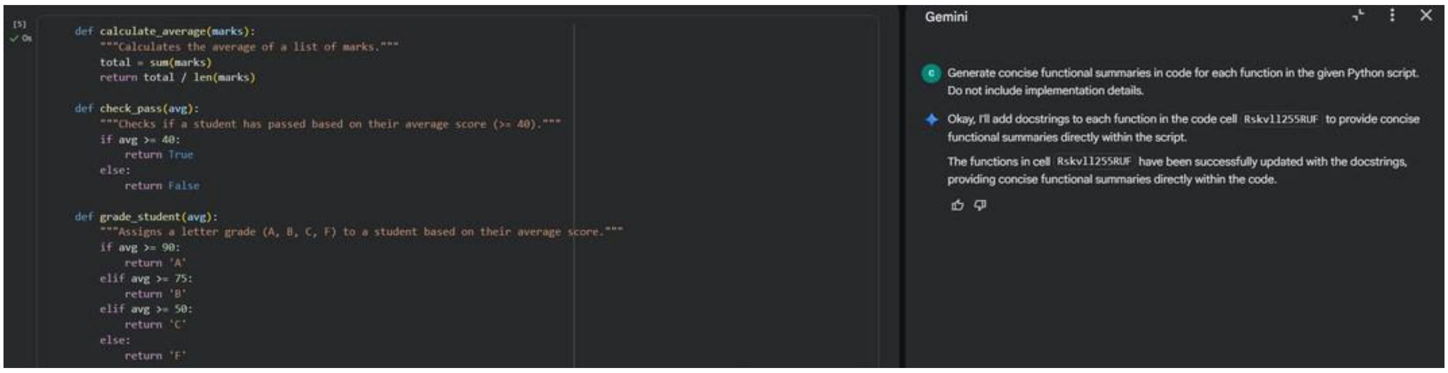
• Ask the AI to write a short summary describing the purpose of each function.

• 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.



```python
def calculate_average(marks):
    """Calculates the average of a list of marks."""
    total = sum(marks)
    return total / len(marks)

def check_pass(avg):
    """Checks if a student has passed based on their average score (>= 40)."""
    if avg >= 40:
        return True
    else:
        return False

def grade_student(avg):
    """Assigns a letter grade (A, B, C, F) to a student based on their average score."""
    if avg >= 90:
        return 'A'
    elif avg >= 75:
        return 'B'
    elif avg >= 50:
        return 'C'
    else:
        return 'F'
```

Gemini

Generate concise functional summaries in code for each function in the given Python script. Do not include implementation details.

Okay, I'll add docstrings to each function in the code cell `Rskvl1255RUF` to provide concise functional summaries directly within the script.

The functions in cell `Rskvl1255RUF` have been successfully updated with the docstrings, providing concise functional summaries directly within the code.

Prompt Used: Generate concise functional summaries in code for each function in the given Python script. Do not include implementation details.

Explanation: Prompt detail affects documentation quality. Detailed prompts give clearer and accurate outputs.

Task -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:

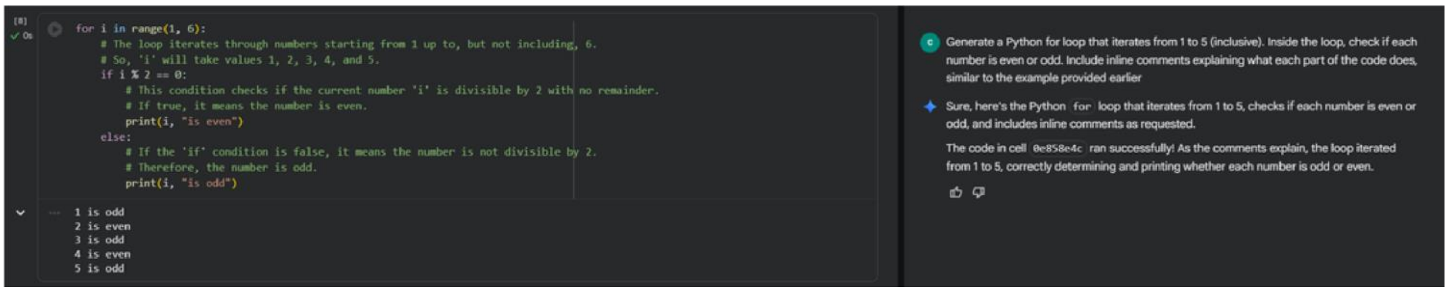
• Provide a Python program without comments.

• 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.



```
for i in range(1, 6):
    # The loop iterates through numbers starting from 1 up to, but not including, 6.
    # So, 'i' will take values 1, 2, 3, 4, and 5.
    if i % 2 == 0:
        # This condition checks if the current number 'i' is divisible by 2 with no remainder.
        # If true, it means the number is even.
        print(i, "is even")
    else:
        # If the 'if' condition is false, it means the number is not divisible by 2.
        # Therefore, the number is odd.
        print(i, "is odd")
```

```
1 is odd
2 is even
3 is odd
4 is even
5 is odd
```

Generate a Python for loop that iterates from 1 to 5 (inclusive). Inside the loop, check if each number is even or odd. Include inline comments explaining what each part of the code does, similar to the example provided earlier

Sure, here's the Python `for` loop that iterates from 1 to 5, checks if each number is even or odd, and includes inline comments as requested.

The code in cell `0e858e4c` ran successfully! As the comments explain, the loop iterated from 1 to 5, correctly determining and printing whether each number is odd or even.

Prompt Used: Generate a Python for loop that iterates from 1 to 5 (inclusive). Inside the loop, check if each number is even or odd. Include inline comments explaining what each part of the code does.

Explanation: AI can identify decision-making logic and loop behavior in programs and explain them clearly. This improves understanding of program flow without focusing on syntax, making it easier for developers to interpret conditional and iterative logic.

Task - 3 (Documentation – File-Level Overview)

Task: Use AI to generate a high-level overview describing the functionality of an entire Python file.

Instructions:

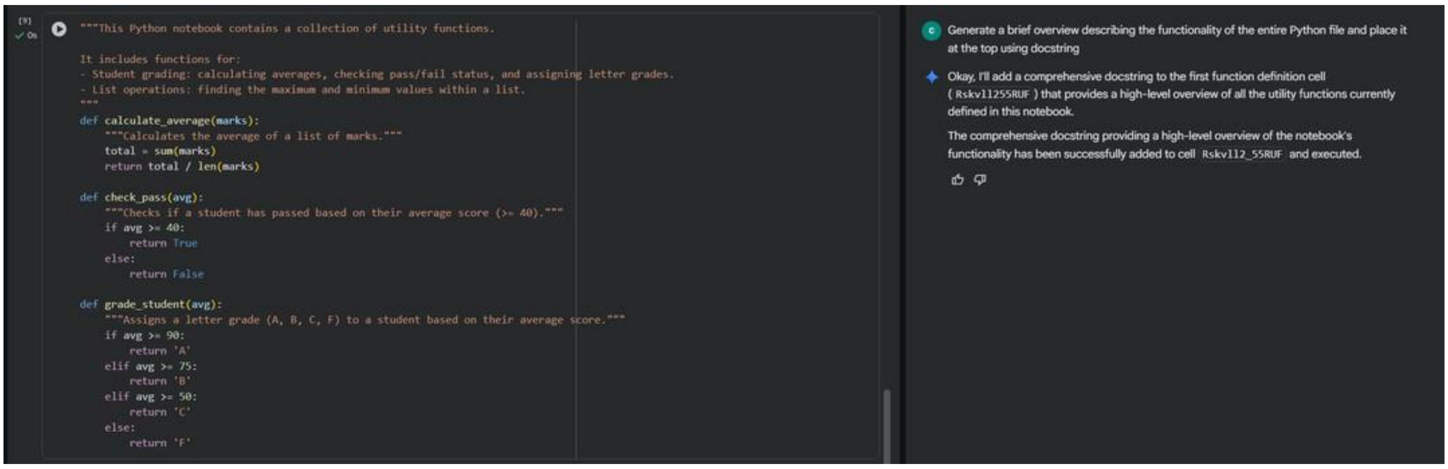
• Provide the complete Python file to AI.

• 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.



```
"""This Python notebook contains a collection of utility functions.

It includes functions for:
- Student grading: calculating averages, checking pass/fail status, and assigning letter grades.
- List operations: finding the maximum and minimum values within a list.
"""
def calculate_average(marks):
    """Calculates the average of a list of marks."""
    total = sum(marks)
    return total / len(marks)

def check_pass(avg):
    """Checks if a student has passed based on their average score (>= 40)."""
    if avg >= 40:
        return True
    else:
        return False

def grade_student(avg):
    """Assigns a letter grade (A, B, C, F) to a student based on their average score."""
    if avg >= 90:
        return 'A'
    elif avg >= 75:
        return 'B'
    elif avg >= 50:
        return 'C'
    else:
        return 'F'
```

Generate a brief overview describing the functionality of the entire Python file and place it at the top using docstring

Okay, I'll add a comprehensive docstring to the first function definition cell (`Rskvl1255RUF`) that provides a high-level overview of all the utility functions currently defined in this notebook.

The comprehensive docstring providing a high-level overview of the notebook's functionality has been successfully added to cell `Rskvl12_55RUF` and executed.

Prompt Used: Generate a brief overview describing the functionality of the entire Python file and place it at the top using docstring

Explanation: AI can generate a high-level overview of an entire Python file. It is useful for summarizing the overall functionality of a program, which improves documentation quality and helps new users quickly understand the purpose of the file.

Task - 4 (Documentation – Refine Existing Documentation)

Task: Use AI to improve clarity and consistency of existing documentation in Python code.

Instructions:

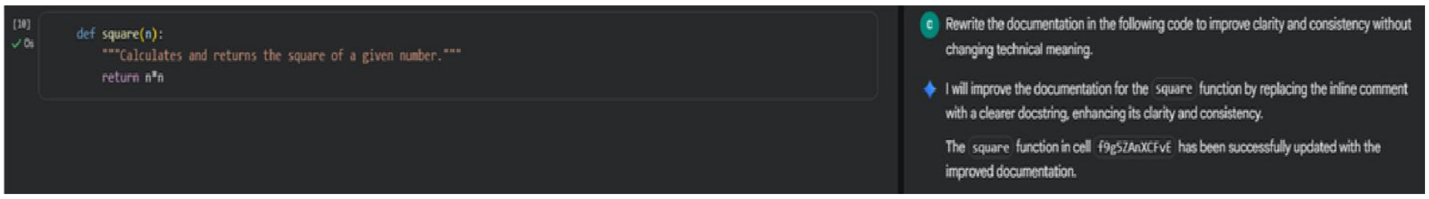
• Provide Python code containing basic or unclear comments.

• 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.



```
[10]
def square(n):
    """Calculates and returns the square of a given number."""
    return n*n
```

Rewrite the documentation in the following code to improve clarity and consistency without changing technical meaning.

I will improve the documentation for the `square` function by replacing the inline comment with a clearer docstring, enhancing its clarity and consistency.

The `square` function in cell `f9gSZAnXCFvE` has been successfully updated with the improved documentation.

Prompt Used: Rewrite the documentation in the following code to improve clarity and consistency without changing technical meaning.

Explanation: AI can refine existing documentation by improving clarity and consistency while preserving the original technical meaning. This helps in maintaining standardized and professional code comments.

Task - 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.

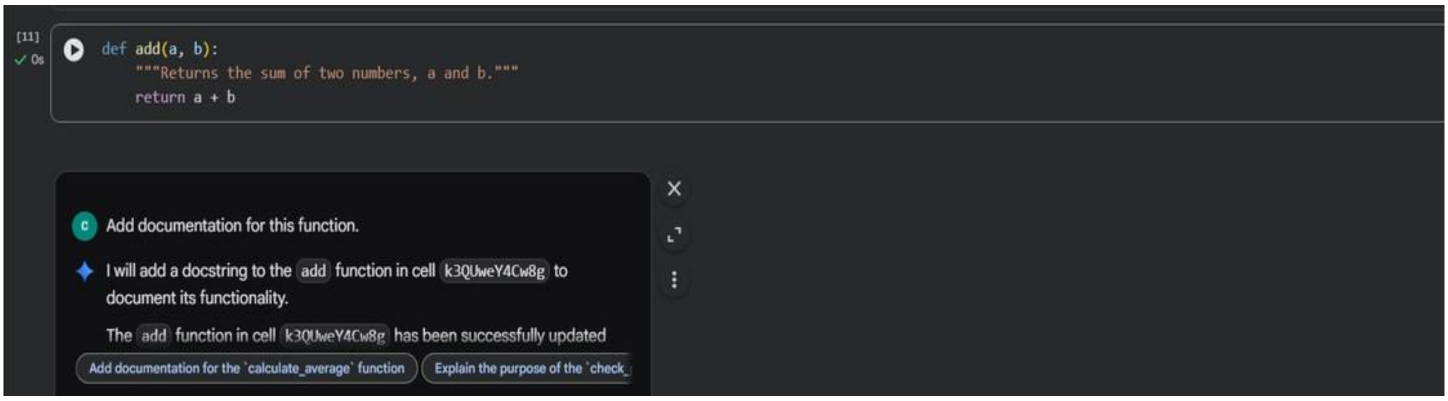
• Use both prompts to document the same Python function.

• Compare the generated outputs.

Expected Output -5:

A comparison table highlighting differences in completeness, clarity, and accuracy of documentation.

Brief Prompt: Add documentation for this function.



```
[11]
 0s
def add(a, b):
    """Returns the sum of two numbers, a and b."""
    return a + b
```

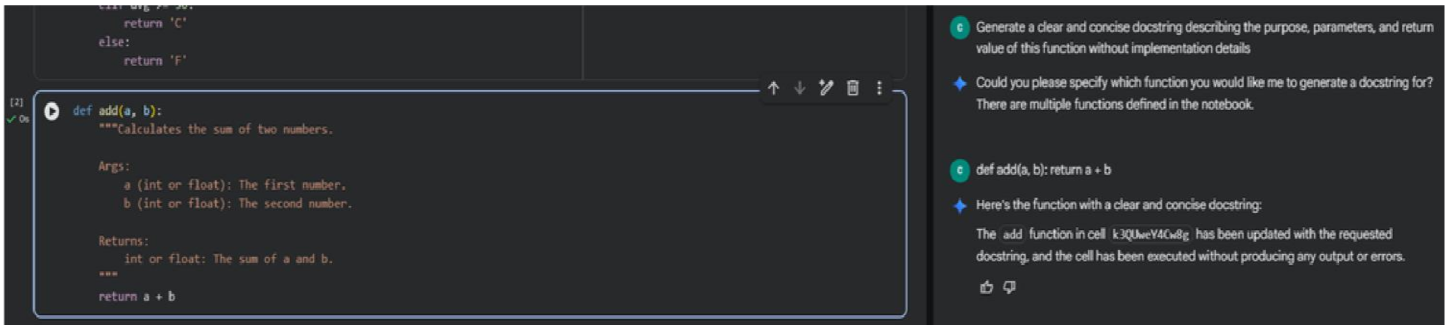
Add documentation for this function.

I will add a docstring to the `add` function in cell `k3QUweY4Cw8g` to document its functionality.

The `add` function in cell `k3QUweY4Cw8g` has been successfully updated

Add documentation for the `calculate_average` function
Explain the purpose of the `check_

Detailed Prompt: Generate a clear and concise docstring describing the purpose, parameters, and return value of this function without implementation details.



```
        return 'C'
    else:
        return 'F'
```

```
[2]
 0s
def add(a, b):
    """Calculates the sum of two numbers.

    Args:
        a (int or float): The first number.
        b (int or float): The second number.

    Returns:
        int or float: The sum of a and b.
    """
    return a + b
```

Generate a clear and concise docstring describing the purpose, parameters, and return value of this function without implementation details

Could you please specify which function you would like me to generate a docstring for? There are multiple functions defined in the notebook.

def add(a, b): return a + b

Here's the function with a clear and concise docstring:

The `add` function in cell `k3QUweY4Cw8g` has been updated with the requested docstring, and the cell has been executed without producing any output or errors.

Explanation: The level of detail in prompts directly affects the quality of AI-generated documentation. Detailed prompts produce more complete, accurate, and readable outputs compared to brief prompts, highlighting the importance of effective prompt engineering.