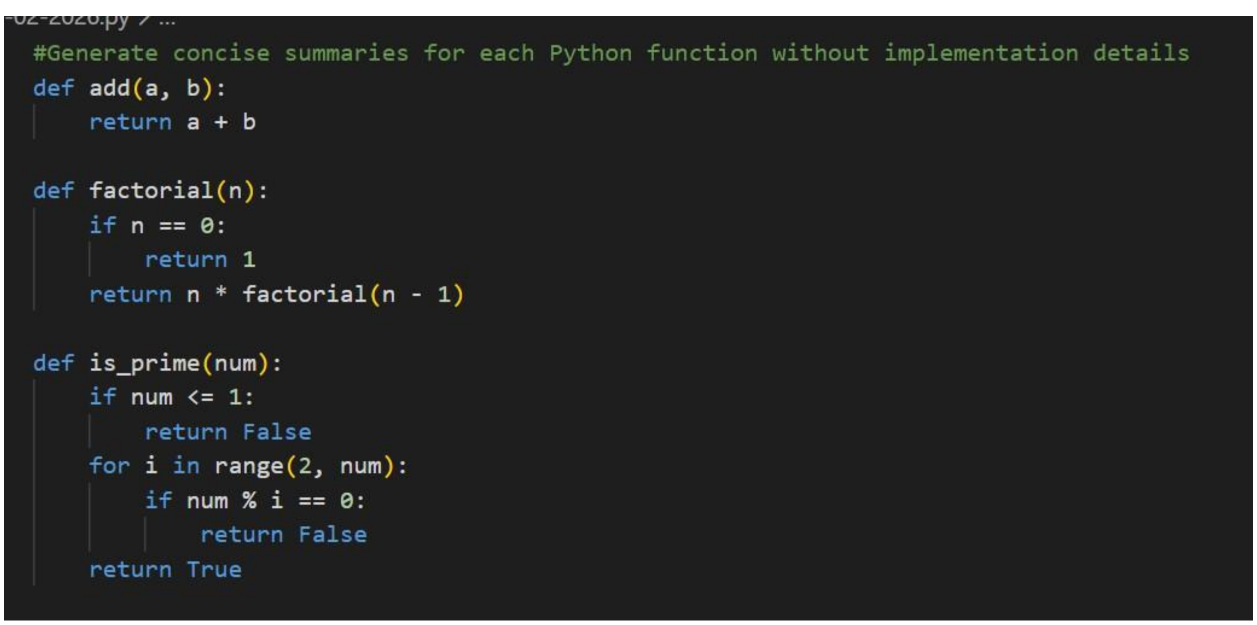
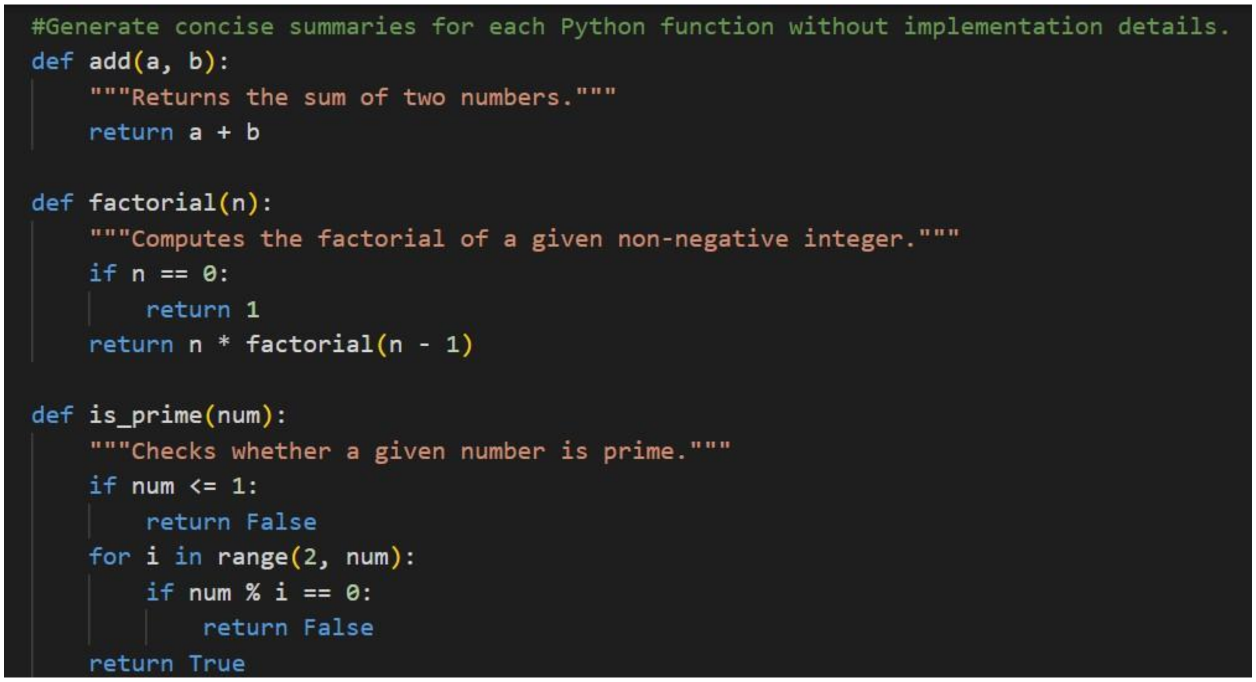
Assignment -9.1

Name : Yashaswini Nagireddi
Htno : 2303A51418
Batch : 21

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.



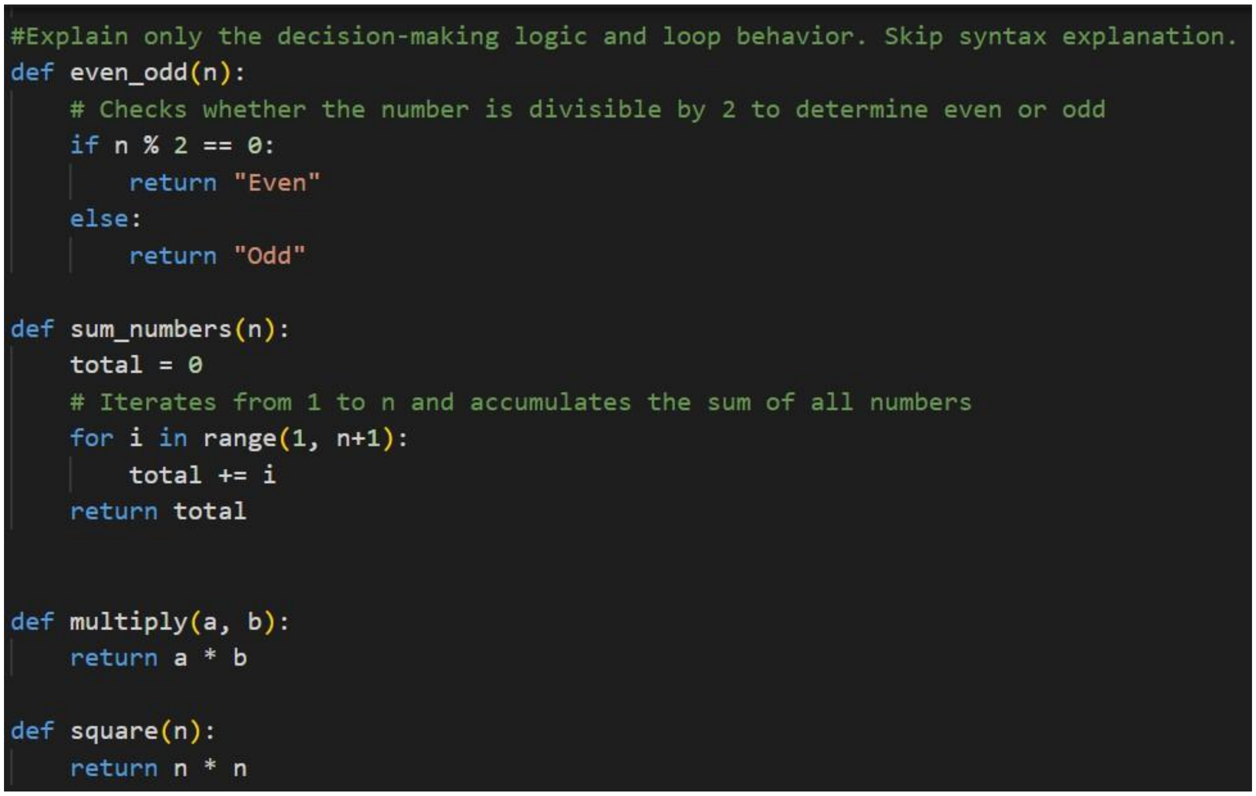
```python
#Generate concise summaries for each Python function without implementation details
def add(a, b):
    return a + b

def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, num):
        if num % i == 0:
            return False
    return True
```



```python
#Generate concise summaries for each Python function without implementation details.
def add(a, b):
    """Returns the sum of two numbers."""
    return a + b

def factorial(n):
    """Computes the factorial of a given non-negative integer."""
    if n == 0:
        return 1
    return n * factorial(n - 1)

def is_prime(num):
    """Checks whether a given number is prime."""
    if num <= 1:
        return False
    for i in range(2, num):
        if num % i == 0:
            return False
    return True
```

Observation:
In this task, AI was used to generate **concise summaries for Python functions**.
The goal was to describe **what each function does**, without explaining the internal implementation details.
A Python script containing multiple functions was
provided to the AI. Using a zero-shot prompt, the AI
generated **docstrings** for each function.
These summaries help developers understand the purpose of functions quickly, improving code readability and maintainability.

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.



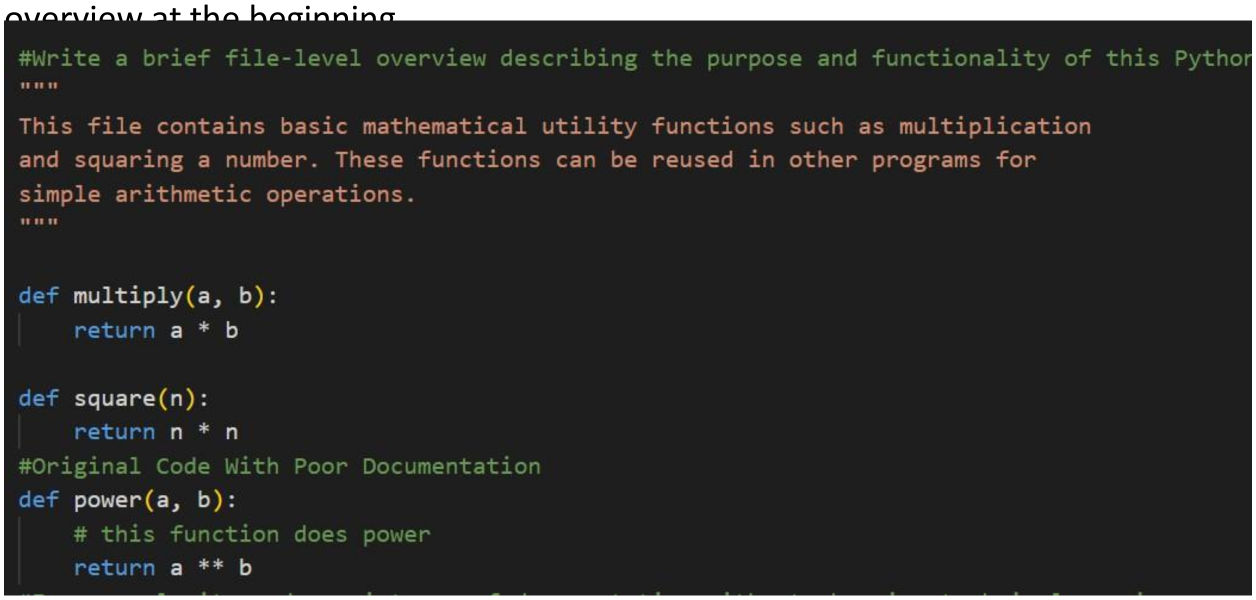
```python
#Explain only the decision-making logic and loop behavior. Skip syntax explanation.
def even_odd(n):
    # Checks whether the number is divisible by 2 to determine even or odd
    if n % 2 == 0:
        return "Even"
    else:
        return "Odd"

def sum_numbers(n):
    total = 0
    # Iterates from 1 to n and accumulates the sum of all numbers
    for i in range(1, n+1):
        total += i
    return total


def multiply(a, b):
    return a * b

def square(n):
    return n * n
```

Explanation:
In this task, AI was used to document **decision-making logic and loop behavior** in Python programs. The program was provided without any comments, and AI was instructed to explain **only the logic**, not Python syntax.
The AI added comments explaining how conditional statements determine outcomes and how loops iterate and accumulate values.
This helps programmers understand program flow and logic, especially in complex programs.

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



```
#Write a brief file-level overview describing the purpose and functionality of this Pytho
"""
This file contains basic mathematical utility functions such as multiplication
and squaring a number. These functions can be reused in other programs for
simple arithmetic operations.
"""

def multiply(a, b):
    return a * b

def square(n):
    return n * n
#Original Code With Poor Documentation
def power(a, b):
    # this function does power
    return a ** b
```

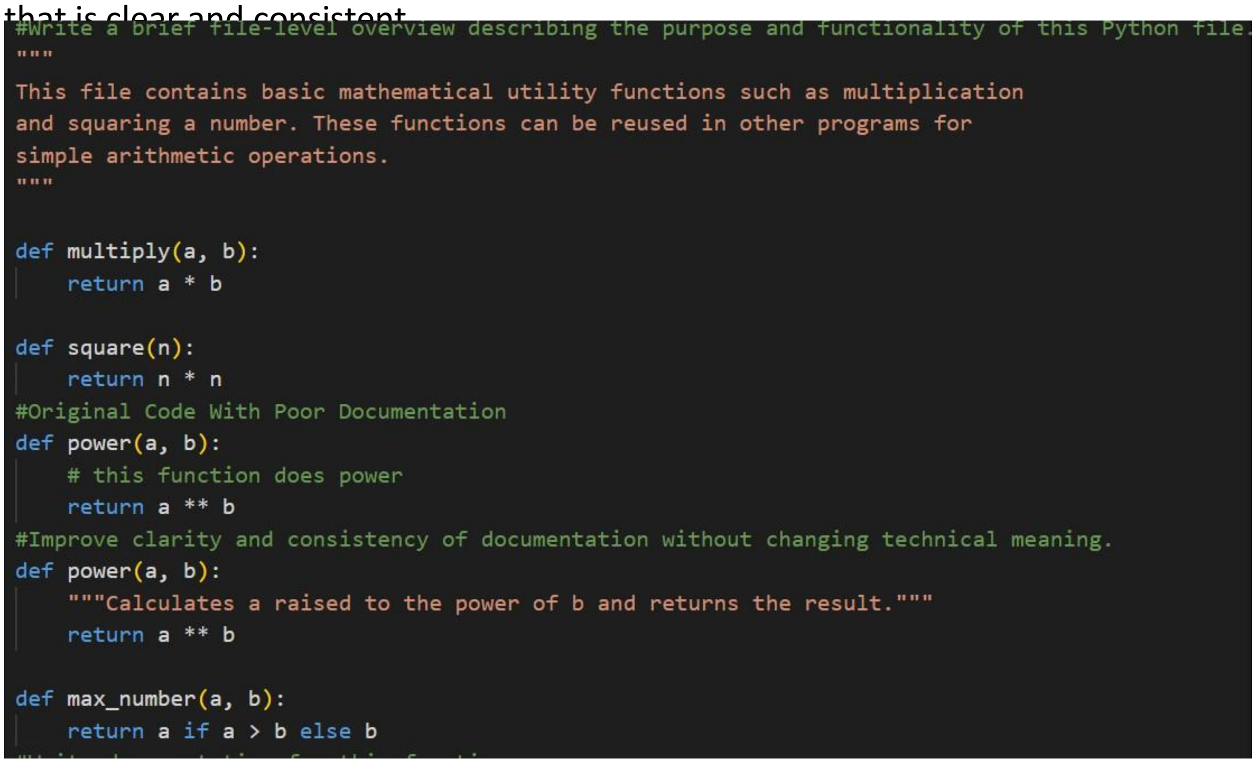
Explanation
In this task, AI generated a **high-level description of an entire Python file**.
The complete Python file was given, and AI created a brief overview explaining the purpose and functionality of the file.
The overview was placed at the top of the file as a module-level docstring.
This type of documentation helps developers understand the file’s role in a project without reading all
the code.

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.

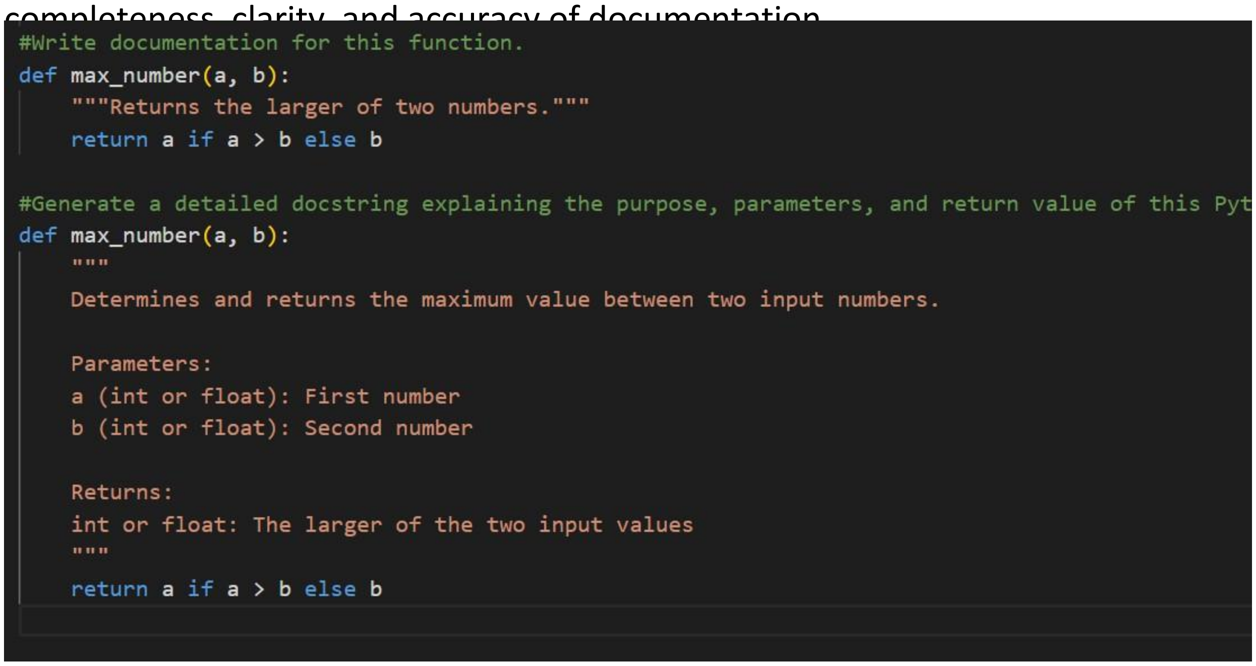


```
#Write a brief file-level overview describing the purpose and functionality of this Python file.
"""
This file contains basic mathematical utility functions such as multiplication
and squaring a number. These functions can be reused in other programs for
simple arithmetic operations.
"""

def multiply(a, b):
    return a * b

def square(n):
    return n * n
#Original Code With Poor Documentation
def power(a, b):
    # this function does power
    return a ** b
#Improve clarity and consistency of documentation without changing technical meaning.
def power(a, b):
    """Calculates a raised to the power of b and returns the result."""
    return a ** b

def max_number(a, b):
    return a if a > b else b
```

Explanation
In this task, AI was used to **improve unclear or basic comments** in Python code. Existing comments were rewritten to be clear, consistent, and professional while keeping the technical meaning unchanged.
This task demonstrates how AI can help in **code documentation refinement**, making comments more readable and standardized.

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.



```python
#Write documentation for this function.
def max_number(a, b):
    """Returns the larger of two numbers."""
    return a if a > b else b

#Generate a detailed docstring explaining the purpose, parameters, and return value of this Pyt
def max_number(a, b):
    """
    Determines and returns the maximum value between two input numbers.

    Parameters:
    a (int or float): First number
    b (int or float): Second number

    Returns:
    int or float: The larger of the two input values
    """
    return a if a > b else b
```

Explanation

In this task, the impact of **prompt detail on AI-generated documentation** was studied. Two prompts were used:

1. A brief prompt
2. A detailed prompt

The same Python function was documented using both prompts.
The outputs were compared based on completeness, clarity, and accuracy.
The results showed that **detailed prompts produce more structured and informative documentation**, while brief prompts generate short summaries.