AI ASSISTED CODING

ASSIGNMENT-9.3

M.KEERTHANA

2403A51259

BATCH-11

CSE-GEN

TASK-1

Basic Docstring Generation

• Write python function to return sum of even and odd numbers in the given list

• Incorporate manual docstring in code with Google Style.

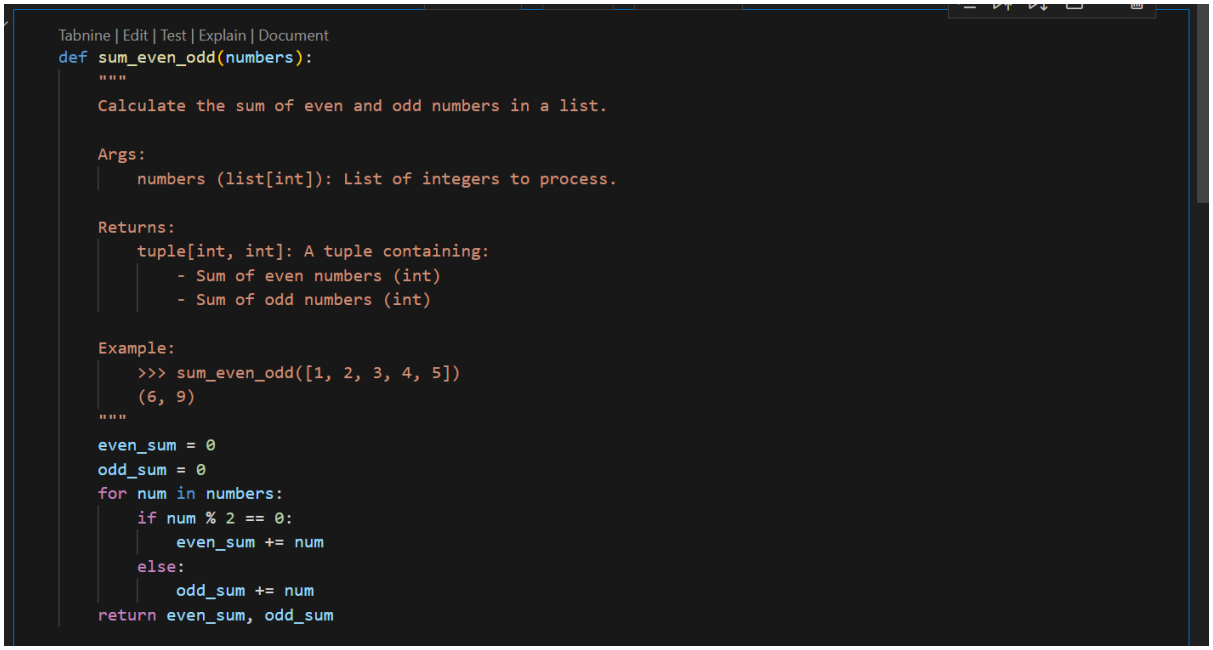
• Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.

• Compare the AI-generated docstring with your manually written one

PROMPT-

Generate a Google-style docstring for the following Python function that calculates and returns the sum of even and odd numbers in a given list. The docstring should include a short description, parameters with type hints, and return values with type hints. Do not include input-output examples.

CODE-



```
Tabnine | Edit | Test | Explain | Document
def sum_even_odd(numbers):
    """
    Calculate the sum of even and odd numbers in a list.

    Args:
        numbers (list[int]): List of integers to process.

    Returns:
        tuple[int, int]: A tuple containing:
            - Sum of even numbers (int)
            - Sum of odd numbers (int)

    Example:
        >>> sum_even_odd([1, 2, 3, 4, 5])
        (6, 9)
    """
    even_sum = 0
    odd_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum
```

AI GENERATED DOCSTRING-

```
# ------------------ AI-Generated Docstring (Simulated) ------------------
ai_docstring = """
This function computes the sum of even and odd numbers from a list of integers.

Parameters:
    numbers (list): A list of integers.

Returns:
    tuple: A tuple with two values:
        - First element is the sum of even numbers.
        - Second element is the sum of odd numbers.
"""

# ------------------ Function Execution ------------------
nums = [10, 15, 20, 25, 30]
```

```
# ------------------ Function Execution ------------------
nums = [10, 15, 20, 25, 30]
result = sum_even_odd(nums)

print("Manual Docstring Output:", result)
print("\n--- AI Generated Docstring ---")
print(ai_docstring)

✓ 0.0s                                                    Python
```

OUTPUT-

```
Manual Docstring Output: (60, 40)

--- AI Generated Docstring ---

This function computes the sum of even and odd numbers from a list of integers.

Parameters:
    numbers (list): A list of integers.

Returns:
    tuple: A tuple with two values:
        - First element is the sum of even numbers.
        - Second element is the sum of odd numbers.
```

COMPARISON-

- **Manual Docstring (Google Style)**
    - More structured.
    - Includes **type hints** (list[int], tuple[int, int]).
    - Provides an **Example** for clarity.
- **AI-Generated Docstring**
    - Simpler, less detailed.
    - Doesn’t specify **types precisely** (just says list and tuple).
    - Lacks example usage.

Conclusion: The manual version is more professional and precise, while the AI one is shorter and easier to write but misses some details

TASK 2-

Write python program for sru_student class with attributes like name, roll no., hostel_status and fee_update method and display_details method.

- Write comments manually for each line/code block
- Ask an AI tool to add inline comments explaining each line/step.
- Compare the AI-generated comments with your manually written one.

MANUAL COMMENTS-CODE-

```python
class SRUStudent:
    # Constructor to initialize student details
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, name, roll_no, hostel_status):
        # Save student name
        self.name = name
        # Save student roll number
        self.roll_no = roll_no
        # Save hostel status (True/False)
        self.hostel_status = hostel_status
        # Start with base tuition fee
        self.fee = 20000

    # Method to update fee based on hostel choice
    Tabnine | Edit | Test | Explain | Document
    def fee_update(self):
        # If student lives in hostel, add hostel charges
        if self.hostel_status:
            self.fee += 15000
        else:
            # If not in hostel, add transport charges
            self.fee += 5000
        return self.fee

    # Method to print all details
    Tabnine | Edit | Test | Explain | Document
    def display_details(self):
        print("=== Student Details ===")
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", "Yes" if self.hostel_status else "No")
        print("Total Fee:", self.fee)
```

OUTPUT-

```
=== Student Details ===
Name: Ananya
Roll No: 23CS2001
Hostel Status: Yes
Total Fee: 35000

=== Student Details ===
Name: Vikram
Roll No: 23CS2002
Hostel Status: No
Total Fee: 25000
```

PROMPT-

Add inline comments to this Python class SRUStudent which has attributes like name, roll no., hostel_status, and methods fee_update and display_details. The comments should explain each line or step in simple terms

CODE-

```python
class SRUStudent:
    # student class
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, name, roll_no, hostel_status):
        # assign name
        self.name = name
        # assign roll number
        self.roll_no = roll_no
        # assign hostel status
        self.hostel_status = hostel_status
        # set base fee
        self.fee = 20000

    Tabnine | Edit | Test | Explain | Document
    def fee_update(self):
        # check hostel status
        if self.hostel_status:
            # add hostel charges
            self.fee += 15000
        else:
            # add transport charges
            self.fee += 5000
        return self.fee

    Tabnine | Edit | Test | Explain | Document
    def display_details(self):
        # print student details
        print("=== Student Details ===")
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", "Yes" if self.hostel_status else "No")
        print("Total Fee:", self.fee)
```

OUTPUT-

```
=== Student Details ===
Name: Ananya
Roll No: 23CS2001
Hostel Status: Yes
Total Fee: 35000

=== Student Details ===
Name: Vikram
Roll No: 23CS2002
Hostel Status: No
Total Fee: 25000
```

TASK 3-

Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).

• Incorporate manual docstring in code with NumPy Style

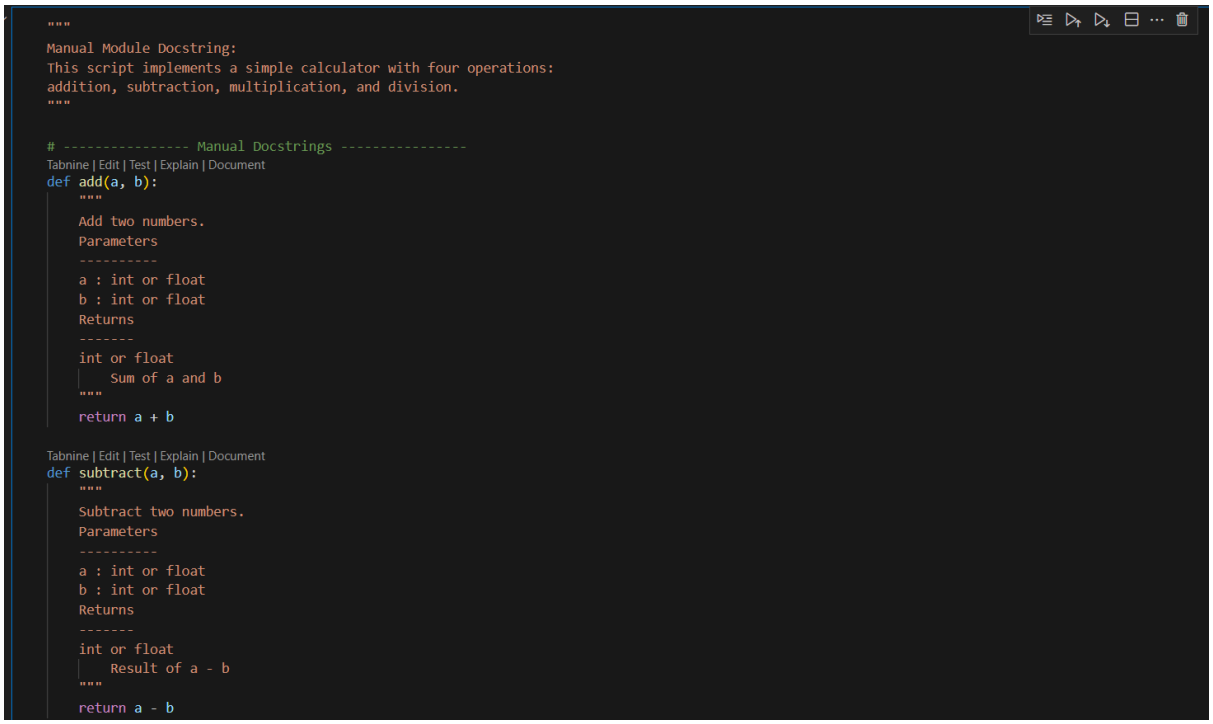
• Use AI assistance to generate a module-level docstring + individual function docstrings.

• Compare the AI-generated docstring with your manually written one.

PROMPT-

Write a module-level and function-level docstring in NumPy style for a Python calculator with functions add, subtract, multiply, and divide. Include parameter types, return types, and error handling.

CODE-



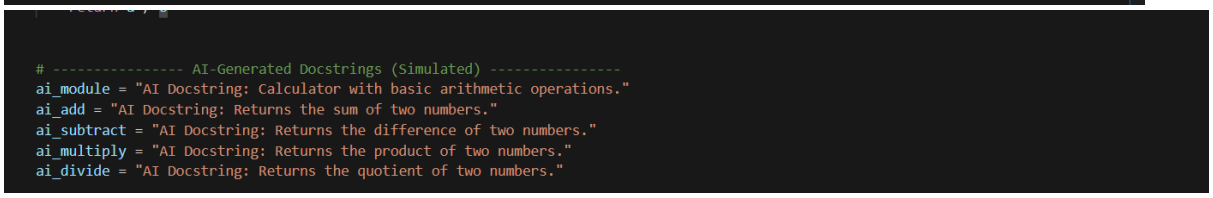
```
"""
Manual Module Docstring:
This script implements a simple calculator with four operations:
addition, subtraction, multiplication, and division.
"""

# ---------------- Manual Docstrings ----------------
Tabnine | Edit | Test | Explain | Document
def add(a, b):
    """
    Add two numbers.
    Parameters
    ----------
    a : int or float
    b : int or float
    Returns
    -------
    int or float
        Sum of a and b
    """
    return a + b

Tabnine | Edit | Test | Explain | Document
def subtract(a, b):
    """
    Subtract two numbers.
    Parameters
    ----------
    a : int or float
    b : int or float
    Returns
    -------
    int or float
        Result of a - b
    """
    return a - b
```

```
Tabnine | Edit | Test | Explain | Document
def multiply(a, b):
    """
    Multiply two numbers.
    Parameters
    ----------
    a : int or float
    b : int or float
    Returns
    -------
    int or float
        Product of a and b
    """
    return a * b

Tabnine | Edit | Test | Explain | Document
def divide(a, b):
    """
    Divide a by b.
    Parameters
    ----------
    a : int or float
    b : int or float
    Returns
    -------
    float
        Result of a / b
    Raises
    ------
    ValueError
        If b = 0
    """
    if b == 0:
        raise ValueError("Division by zero not allowed.")
    return a / b
```



```
# ---------------- AI-Generated Docstrings (Simulated) ----------------
ai_module = "AI Docstring: Calculator with basic arithmetic operations."
ai_add = "AI Docstring: Returns the sum of two numbers."
ai_subtract = "AI Docstring: Returns the difference of two numbers."
ai_multiply = "AI Docstring: Returns the product of two numbers."
ai_divide = "AI Docstring: Returns the quotient of two numbers."
```

```python
# ---------------- Main Execution ----------------
if __name__ == "__main__":
    # Run calculations
    print("=== Calculator Results ===")
    print("Add(8, 2):", add(8, 2))
    print("Subtract(8, 2):", subtract(8, 2))
    print("Multiply(8, 2):", multiply(8, 2))
    print("Divide(8, 2):", divide(8, 2))

    # Show docstrings
    print("\n=== Module Docstring Comparison ===")
    print(__doc__)
    print(ai_module)

    print("\n=== Function Docstring Comparison ===")
    print("Manual add:", add.__doc__)
    print(ai_add)
    print("Manual subtract:", subtract.__doc__)
    print(ai_subtract)
    print("Manual multiply:", multiply.__doc__)
    print(ai_multiply)
    print("Manual divide:", divide.__doc__)
    print(ai_divide)

    # Summary
    print("\n=== Comparison Summary ===")
    print("Manual docstrings (NumPy style) are structured, detailed, and include types/errors.")
    print("AI docstrings are shorter, less detailed, but faster to generate.")
```

OUTPUT-

```
=== Calculator Results ===
Add(8, 2): 10
Subtract(8, 2): 6
Multiply(8, 2): 16
Divide(8, 2): 4.0

=== Module Docstring Comparison ===

Manual Module Docstring:
This script implements a simple calculator with four operations:
addition, subtraction, multiplication, and division.

AI Docstring: Calculator with basic arithmetic operations.

=== Function Docstring Comparison ===
Manual add:
    Add two numbers.
    Parameters
    ----------
    a : int or float
    b : int or float
    Returns
    -------
    int or float
        Sum of a and b
```

```
AI Docstring: Returns the sum of two numbers.
Manual subtract:
    Subtract two numbers.
    Parameters
    ----------
    a : int or float
    b : int or float
    Returns
    -------
    int or float
        Result of a - b

AI Docstring: Returns the difference of two numbers.
Manual multiply:
    Multiply two numbers.
    Parameters
    ----------
    a : int or float
    b : int or float
    Returns
    -------
    int or float
        Product of a and b

AI Docstring: Returns the product of two numbers.
Manual divide:
    Divide a by b.
    Parameters
    ----------
```

```
        Product of a and b

AI Docstring: Returns the product of two numbers.
Manual divide:
    Divide a by b.
    Parameters
    ----------
    a : int or float
    b : int or float
    Returns
    -------
    float
        Result of a / b
    Raises
    ------
    ValueError
        If b = 0

AI Docstring: Returns the quotient of two numbers.

=== Comparison Summary ===
Manual docstrings (NumPy style) are structured, detailed, and include types/errors.
AI docstrings are shorter, less detailed, but faster to generate.
```