AI ASSISTED CODING

ASSIGNMENT – 9.3

Name : Baggi Praneeth

HT NO.: 2403A51246

BATCH:11

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.
Expected Outcome-1: Students understand how AI can produce function-level documentation.

Prompt: Write a Python function to return the sum of even and odd numbers in a list. Add a manual Google-style docstring. Then, use an AI tool to generate a docstring for the same function. Compare both docstrings.

Explanation: Manual docstrings and comments are more detailed, structured, and educational, clearly explaining parameters, return values, examples, and code purpose. AI-generated documentation is concise and readable, providing a quick understanding but often lacking depth and examples. Overall, manual documentation is better for learning and professional code, while AI-generated docs are useful for saving time and quickly understanding the code.

Code:

```
Tabnine | Edit | Test | Explain | Document
def sum_even_odd(numbers):
    """
    Compute the sum of even and odd numbers from a list.
    Args:
        numbers (list of int): List containing integers.

    Returns:
        tuple: A tuple containing:
            - sum_even (int): Sum of all even numbers.
            - sum_odd (int): Sum of all odd numbers.
    Example:
        >>> sum_even_odd([10, 21, 32, 43])
        (42, 64)
    """
    sum_even = 0
    sum_odd = 0
    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num
    return sum_even, sum_odd
# Test the function
if __name__ == "__main__":
    test_list = [10, 21, 32, 43]
    even_sum, odd_sum = sum_even_odd(test_list)
    print(f"Sum of even numbers: {even_sum}")
    print(f"Sum of odd numbers: {odd_sum}")
```

Output:

```
python3.11.exe "c:/Users/sonti/OneDrive/Documents/aicodi
Sum of even numbers: 42
Sum of odd numbers: 64
PS C:\Users\sonti\OneDrive\Documents\aicodind ass-3>
```

Task-2: Automatic Inline Comments

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.

Prompt: Write a Python class sru_student with attributes name, roll_no, hostel_status, and methods fee_update and display_details. Add manual inline comments for each line. Then, use AI to generate inline comments for the same code and compare both sets of comments**.**

Explanation: Manual comments precisely explain the purpose of each line or code block, helping the reader understand the logic clearly and encouraging critical thinking. AI-generated comments also describe each line but tend to be more verbose and sometimes repeat obvious information, which can be helpful for beginners but may feel redundant. Overall, manual comments are more concise and targeted, while AI comments provide extra guidance but with less efficiency**.**

Code:

AI-Generated Inline Comments Version code:

```
class sru_student:
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, name, roll_no, hostel_status):
        self.name = name            # Assign the provided name to the student object
        self.roll_no = roll_no      # Assign the provided roll number to the student object
        self.hostel_status = hostel_status  # Assign the hostel status (True/False)

    Tabnine | Edit | Test | Explain | Document
    def fee_update(self, fee_amount):
        self.fee = fee_amount       # Update the student's fee with the given amount

    Tabnine | Edit | Test | Explain | Document
    def display_details(self):
        print(f"Name: {self.name}")          # Output the student's name
        print(f"Roll No: {self.roll_no}")    # Output the student's roll number
        print(f"Hostel Status: {self.hostel_status}")  # Output whether the student is in hostel
        if hasattr(self, 'fee'):             # Check if the student object has a fee attribute
            print(f"Fee: {self.fee}")       # Output the fee if it exists

student1 = sru_student("Alice", 101, True)  # Instantiate a student object
student1.fee_update(5000)                   # Call method to update the student's fee
student1.display_details()                  # Call method to display all student details
```

Manual Comments Version code:p

```python
# Define the class sru_student
class sru_student:
    # Constructor method to initialize student attributes
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, name, roll_no, hostel_status):
        self.name = name            # Store student's name
        self.roll_no = roll_no      # Store student's roll number
        self.hostel_status = hostel_status  # Store hostel status (True/False)
    # Method to update the student's fee
    Tabnine | Edit | Test | Explain | Document
    def fee_update(self, fee_amount):
        self.fee = fee_amount       # Store the updated fee
    # Method to display all student details
    Tabnine | Edit | Test | Explain | Document
    def display_details(self):
        print(f"Name: {self.name}")          # Print student's name
        print(f"Roll No: {self.roll_no}")    # Print roll number
        print(f"Hostel Status: {self.hostel_status}")  # Print hostel status
        if hasattr(self, 'fee'):             # Check if fee is updated
            print(f"Fee: {self.fee}")       # Print fee if it exists
# Create a student object
student1 = sru_student("Alice", 101, True)
# Update fee for the student
student1.fee_update(5000)
# Display student details
student1.display_details()
```

Output:

```
PS C:\Users\sonti\OneDrive\Documents\aicodind ass-3> & C:
python3.11.exe "c:/Users/sonti/OneDrive/Documents/aicodin
Name: Alice
Roll No: 101
Hostel Status: True
Fee: 5000
PS C:\Users\sonti\OneDrive\Documents\aicodind ass-3>
```

Task-3: Calculator

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.
Expected Output-3: Students learn structured documentation for multi-function scripts

Prompt: Write a Python script with functions add, subtract, multiply, and divide. Add manual NumPy-style docstrings for each function. Then, use AI to generate module-level and function-level docstrings. Compare AI-generated docstrings with manual ones.

Explanation: The manually written docstrings are detailed, structured, and follow a standard format like Google or NumPy style, including parameters, return values, examples, and error handling. In contrast, the AI-generated docstrings are concise and easy to read but often lack depth, structure, and examples. Overall, manual docstrings are more professional and educational, while AI-generated ones are quicker for basic understanding.

Code:

Manual NumPy-style Docstrings Version code:

```
"""
calculator_module.py
A simple calculator module with basic arithmetic operations.
This module provides functions to add, subtract, multiply, and divide numbers.
"""
Tabnine | Edit | Test | Explain | Document
def add(a, b):
    """
    Add two numbers.
    Parameters
    ----------
    a : float
        First number.
    b : float
        Second number.
    Returns
    -------
    float
        Sum of a and b.
    Examples
    --------
    >>> add(2, 3)
    5
    """
    return a + b
Tabnine | Edit | Test | Explain | Document
def subtract(a, b):
    """
    Subtract second number from first number.
```

```
        Minuend.
    b : float
        Subtrahend.
    Returns
    -------
    float
        Difference (a - b).

    Examples
    --------
    >>> subtract(5, 3)
    2
    """
    return a - b
Tabnine | Edit | Test | Explain | Document
def multiply(a, b):
    """
    Multiply two numbers.
    Parameters
    ----------
    a : float
        First factor.
    b : float
        Second factor.
    Returns
    -------
    float
        Product of a and b.
```

```
    >>> multiply(2, 3)
    6
    """
    return a * b
Tabnine | Edit | Test | Explain | Document
def divide(a, b):
    """
    Divide first number by second number.
    Parameters
    ----------
    a : float
        Dividend.
    b : float
        Divisor (must not be zero).
    Returns
    -------
    float
        Quotient (a / b).
    Raises
    ------
    ValueError
        If b is zero.
    Examples
    --------
    >>> divide(6, 3)
    2.0
```

```
    """
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
# Example usage
if __name__ == "__main__":
    print("Add:", add(5, 3))
    print("Subtract:", subtract(5, 3))
    print("Multiply:", multiply(5, 3))
    print("Divide:", divide(5, 3))
```

AI-Generated Docstrings version code:

```
"""
Calculator Module

This module contains basic arithmetic functions: add, subtract, multiply, and divide.
Each function takes two numbers as input and returns the computed result.
"""

Tabnine | Edit | Test | Explain | Document
def add(a, b):
    """Return the sum of two numbers a and b."""
    return a + b

Tabnine | Edit | Test | Explain | Document
def subtract(a, b):
    """Return the difference when b is subtracted from a."""
    return a - b

Tabnine | Edit | Test | Explain | Document
def multiply(a, b):
    """Return the product of two numbers a and b."""
    return a * b

Tabnine | Edit | Test | Explain | Document
def divide(a, b):
    """Return the result of dividing a by b. Raises error if b is zero."""
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
```

Output:

```
python3.11.exe "c:/Users/sonti/OneDrive/Documents/aicodind ass-3/a
Add: 8
Subtract: 2
Multiply: 15
Divide: 1.6666666666666667
PS C:\Users\sonti\OneDrive\Documents\aicodind ass-3> & C:\Users\so
```