2403a51276
Mohammed Ibrahim
Batch 12

AI-ASSISTED-CODING
Assignment-9.3

Task Description#1:

Basic Docstring Generation

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

```
def sum_even_odd(numbers):
    """
    Calculates the sum of even and odd
        numbers in a given list.

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

    Returns:
        Tuple[int, int]: A tuple
            containing the sum of even
            numbers and the sum of odd
            numbers.
    """
    even_sum = sum(num for num in numbers
        if num % 2 == 0)
    odd_sum = sum(num for num in numbers
        if num % 2 != 0)
    return even_sum, odd_sum
```

• Incorporate manual docstring in code with Google Style

```
"""
Calculates the sum of even and odd numbers
    in a given list.

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

Returns:
    Tuple[int, int]: A tuple containing
        the sum of even numbers and the
        sum of odd numbers.
"""
```

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

Manually Written Docstring (Google Style)

Python Copy

```
"""
Calculates the sum of even and odd numbers in

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

Returns:
    Tuple[int, int]: A tuple containing the su
"""
```

AI-Generated Docstring

Python Copy

```
"""
Returns the sum of even and odd numbers from t

Parameters:
    numbers (List[int]): List of integers to p

Returns:
    Tuple[int, int]: A tuple where the first e
    and the second is the sum of odd numbers.
"""
```

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

Aspect	Manual Docstring	AI-Generated Docstring
Clarity	Clear and concise	Also clear, slightly more conversational
Structure	Strict Google Style	Mostly Google Style, but uses "Parameters"
Detail	Precise and formal	Slightly more casual but informative
Terminology	Uses "Args" and "Returns"	Uses "Parameters" instead of "Args"

Task Description#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.

```
class sru_student:
    # Constructor to initialize student attributes
    def __init__(self, name, roll_no, hostel_status):
        self.name = name  # Student's name
        self.roll_no = roll_no  # Student's roll number
        self.hostel_status = hostel_status  # Whether the student stays in hostel
        self.fee_paid = False  # Initial fee status set to unpaid

    # Method to update fee status
    def fee_update(self, status):
        self.fee_paid = status  # Update fee status based on input

    # Method to display student details
    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
        print(f"Fee Paid: {'Yes' if self.fee_paid else 'No'}")  # Print fee payment status

```

• Write comments manually for each line/code block

```
class sru_student:
    # Define a class to represent a student at SRU
    def __init__(self, name, roll_no, hostel_status):
        self.name = name  # Assign the student's name
        self.roll_no = roll_no  # Assign the student's roll number
        self.hostel_status = hostel_status  # Assign hostel status (Yes/No)
        self.fee_paid = False  # Initialize fee status to unpaid

    def fee_update(self, status):
        self.fee_paid = status  # Set the fee status to the given value

    def display_details(self):
        print(f"Name: {self.name}")  # Display the student's name
        print(f"Roll No: {self.roll_no}")  # Display the roll number
        print(f"Hostel Status: {self.hostel_status}")  # Display hostel status
        print(f"Fee Paid: {'Yes' if self.fee_paid else 'No'}")  # Display fee payment status
```

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

Line/Block	Manual Comment	AI-Generated Comment
`class sru_student:`	No comment	"Define a class to represent a student..."
`__init__` method	"Constructor to initialize..."	"Initialize student attributes" (implied)
`self.name = name`	"Student's name"	"Assign the student's name"
`self.fee_paid = False`	"Initial fee status set to unpaid"	"Initialize fee status to unpaid"
`fee_update` method	"Method to update fee status"	"Set the fee status to the given value"
`display_details` method	"Method to display student details"	"Display the student's..." (per line)

Task Description#3:

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

```
# ========================
# Manual Module Docstring
# ========================
"""
This module provides basic arithmetic operations: addition, subtraction,
multiplication, and division. Each function takes two numeric inputs and
returns the result of the operation.

Functions:
    add(a, b)
    subtract(a, b)
    multiply(a, b)
    divide(a, b)
"""

def add(a, b):
    """
    Add two numbers.

    Parameters
    ----------
    a : float or int
        The first number.
```

```
    float or int
        The result of a - b.
    """
    return a - b

def multiply(a, b):
    """
    Multiply two numbers.

    Parameters
    ----------
    a : float or int
        The first number.
    b : float or int
        The second number.

    Returns
    -------
    float or int
        The product of a and b.
    """
    return a * b
```

```
        The first number.
    b : float or int
        The second number.

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

def subtract(a, b):
    """
    Subtract one number from another.

    Parameters
    ----------
    a : float or int
        The number to subtract from.
    b : float or int
        The number to subtract.

    Returns
```

```
70  def divide(a, b):
71      """
72      Divide one number by another.
73
74      Parameters
75      ----------
76      a : float or int
77          The numerator.
78      b : float or int
79          The denominator.
80
81      Returns
82      -------
83      float
84          The result of a / b.
85
86      Raises
87      ------
88      ValueError
89          If b is zero.
90      """
91      if b == 0:
92          raise ValueError("Cannot divide by zero.")
```

```
91      if b == 0:
92          raise ValueError("Cannot divide by zero.")
93      return a / b
```

• Incorporate manual docstring in code with

Numpy

```
"""
A simple calculator module that provides functions to perform basic arithmetic operations:
addition, subtraction, multiplication, and division.
"""

def add(a, b):
    """Returns the sum of two numbers."""
    return a + b

def subtract(a, b):
    """Returns the difference between two numbers."""
    return a - b

def multiply(a, b):
    """Returns the product of two numbers."""
    return a * b

def divide(a, b):
    """Returns the result of dividing a by b. Raises an error if b is zero."""
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b
```

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

```
"""
A basic calculator module that provides functions for performing arithmetic operations.

This module includes functions to add, subtract, multiply, and divide two numeric values.
Each function accepts two inputs and returns the result of the corresponding operation.

Functions
---------
add(a, b)
    Returns the sum of two numbers.

subtract(a, b)
    Returns the difference between two numbers.

multiply(a, b)
    Returns the product of two numbers.

divide(a, b)
    Returns the quotient of two numbers, with error handling for division by zero.
```

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

Feature	Manual Docstring (NumPy Style)	AI-Generated Docstring
Format	Structured with headers (`Parameters`, `Returns`)	Simple one-liners
Detail Level	Explicit types, error handling, and descriptions	Concise and readable
Consistency	Uniform across all functions	Varies slightly in phrasing
Style Compliance	Fully adheres to NumPy Style	More casual, not style-specific
Module Docstring	Lists functions and purpose	General overview only