

Assignment-9.3

Ht.No: 2303A51584

Batch: 05

Task 1: Basic Docstring Generation

Scenario

You are developing a utility function that processes numerical lists and must be properly

documented for future maintenance.

Requirements

- Write a Python function to return the sum of even numbers and sum of odd numbers in a

given list

- Manually add a Google Style docstring to the function
- Use an AI-assisted tool (Copilot / Cursor AI) to generate a function-level docstring
- Compare the AI-generated docstring with the manually written docstring
- Analyze clarity, correctness, and completeness

] Os



```
def sum_even_odd(numbers):  
    """  
    Returns the sum of even and odd numbers from a list.  
  
    Args:  
        nums (list): List of integers.  
  
    Returns:  
        tuple: (sum_of_even, sum_of_odd)  
    """  
    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
```



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

    This function iterates through the given list of integers and
    computes two separate sums:
    - One for even numbers
    - One for odd numbers

    Args:
    | | numbers (list of int): A list containing integer values.

    Returns:
    | | tuple: A tuple containing two integers:
    | | | | - Sum of even numbers
    | | | | - Sum of odd numbers

    Example:
    | | >>> sum_even_odd([1, 2, 3, 4, 5, 6])
    | | (12, 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
```

Comparison:

Manual docstring:

- A manual docstring is written by the programmer in their own words.
- Manual docstrings may be short or long depending on the person.
- Writing a manual docstring takes more time.
- Manual docstrings show the programmer's understanding of the code.
- Manual docstrings are commonly used in exams and learning.

AI-generated docstring:

- An AI-generated docstring is written automatically by an AI tool.
- AI-generated docstrings are usually well-structured and detailed.
- AI-generated docstrings save time and effort.
- AI-generated docstrings depend on the quality of the prompt given.
- AI-generated docstrings are useful in real projects for consistency.

Task 2: Automatic Inline Comments

Scenario

You are developing a student management module that must be easy to understand for new

developers.

Requirements

- Write a Python program for an `sru_student` class with the following:
 - Attributes: `name`, `roll_no`, `hostel_status`
 - Methods: `fee_update()` and `display_details()`

- Manually write inline comments for each line or logical block
- Use an AI-assisted tool to automatically add inline comments
- Compare manual comments with AI-generated comments
- Identify missing, redundant, or incorrect AI comments

[4]

✓ Os



Creating a class named sru_student

class sru_student:

Constructor to initialize student details

def __init__(self, name, roll_no, hostel_status):

self.name = name # Store student name

self.roll_no = roll_no # Store roll number

self.hostel_status = hostel_status # Store hostel status (Yes/No)

self.fee = 0 # Initialize fee as 0

Method to update fee based on hostel status

def fee_update(self):

if self.hostel_status == "Yes": # If student stays in hostel

self.fee = 80000 # Set higher fee

else: # If student is day scholar

self.fee = 50000 # Set lower fee

Method to display student details

def display_details(self):

print("Name:", self.name) # Print name

print("Roll No:", self.roll_no) # Print roll number

print("Hostel Status:", self.hostel_status) # Print hostel status

print("Fee:", self.fee) # Print fee

Creating an object of the class

student1 = sru_student("Preethi", "22CS001", "Yes")

Calling fee update method

student1.fee_update()

Displaying student information

student1.display_details()

```

# Constructor to initialize student details
def __init__(self, name, roll_no, hostel_status):
    self.name = name          # Store student name
    self.roll_no = roll_no    # Store roll number
    self.hostel_status = hostel_status # Store hostel status (Yes/No)
    self.fee = 0              # Initialize fee as 0

# Method to update fee based on hostel status
def fee_update(self):
    if self.hostel_status == "Yes": # If student stays in hostel
        self.fee = 80000 # Set higher fee
    else: # If student is day scholar
        self.fee = 50000 # Set lower fee

# Method to display student details
def display_details(self):
    print("Name:", self.name) # Print name
    print("Roll No:", self.roll_no) # Print roll number
    print("Hostel Status:", self.hostel_status) # Print hostel status
    print("Fee:", self.fee) # Print fee

# Creating an object of the class
student1 = sru_student("Preethi", "22CS001", "Yes")
# Instantiate a student object with given details

# Calling fee update method
student1.fee_update()

```

```
[5] / Os
# Creating a class named sru_student
class sru_student:

    # Constructor to initialize student details
    def __init__(self, name, roll_no, hostel_status):
        self.name = name          # Store student name
        self.roll_no = roll_no    # Store roll number
        self.hostel_status = hostel_status # Store hostel status (Yes/No)
        self.fee = 0              # Initialize fee as 0

    # Method to update fee based on hostel status
    def fee_update(self):
        if self.hostel_status == "Yes":    # If student stays in hostel
            self.fee = 80000               # Set higher fee for hostel residents
        else:                               # If student is day scholar
            self.fee = 50000               # Set lower fee for day scholars

    # Method to display student details
    def display_details(self):
        print("Name:", self.name)          # Print the student's name
        print("Roll No:", self.roll_no)    # Print the student's roll number
        print("Hostel Status:", self.hostel_status) # Print the student's hostel status
        print("Fee:", self.fee)            # Print the calculated fee

# Creating an object of the class
student1 = sru_student("Preethi", "22CS001", "Yes") # Instantiate a student object with given details

# Calling fee update method
student1.fee_update() # Calculate and update the fee based on hostel status

# Displaying student information
student1.display_details() # Print all the student's details

... Name: Preethi
Roll No: 22CS001
Hostel Status: Yes
Fee: 80000
```

Comparison:

Manual comments:

- In the manual version, each line and block in the sru_student program is commented by the programmer to explain what it does.
- Manual comments clearly show the student's understanding of class, constructor, and methods like fee_update() and display_details().
- Writing manual comments for the program takes more time, especially for constructors and methods.
- Manual comments may be simple and exam-oriented.
- Manual comments are preferred in academic submissions to prove learning.

AI-generated comments:

- AI comments explain the same logic but in a more standardized and polished way.
- AI-generated comments reduce effort and save time.
- In the AI-generated version, comments are added automatically based on code understanding.
- AI comments may be slightly more descriptive, sometimes more than required for exams.
- AI-generated comments are useful in real projects where clarity and consistency matter.

Task 3: Module-Level and Function-Level Documentation

Scenario


You are building a small calculator module that will be shared across multiple projects and

requires structured documentation.


Requirements

- Write a Python script containing 3–4 functions (e.g., add, subtract, multiply, divide)
- Manually write NumPy Style docstrings for each function
- Use AI assistance to generate:
 - A module-level docstring
 - Individual function-level docstrings

- Compare AI-generated docstrings with manually written ones
- Evaluate documentation structure, accuracy, and readability



```
def add(a, b):  
    """Add two numbers."""  
    return a + b  
  
def subtract(a, b):  
    """Subtract b from a."""  
    return a - b  
  
def multiply(a, b):  
    """Multiply two numbers."""  
    return a * b  
  
def divide(a, b):  
    """Divide a by b."""  
    return a / b
```





```
def add(a, b):  
    """  
    Adds two numbers.  
  
    Parameters  
    -----  
    a : number  
        The first number.  
    b : number  
        The second number.  
  
    Returns  
    -----  
    number  
        The sum of a and b.  
    """  
    return a + b  
  
def subtract(a, b):  
    """  
    Subtracts b from a.  
  
    Parameters  
    -----  
    a : number  
        The number to subtract from.  
    b : number  
        The number to subtract.  
  
    Returns  
    -----  
    number  
        The result of a minus b.  
    """  
    return a - b  
  
def multiply(a, b):  
    """
```

```
"""
Multiplies two numbers.

Parameters
-----
a : number
    The first number.
b : number
    The second number.

Returns
-----
number
    The product of a and b.
"""
return a * b

def divide(a, b):
    """
    Divides a by b.

    Parameters
    -----
    a : number
        The numerator.
    b : number
        The denominator.

    Returns
    -----
    number
        The result of a divided by b.

    Raises
    -----
    ZeroDivisionError
        If b is zero.
    """
    return a / b
```

Comparsion:

Manual docstrings:

- Manual docstrings are written by the programmer.
- Manual docstrings show the programmer's understanding of the code.
- Writing manual docstrings takes more time.
- Manual docstrings may differ in quality and style.
- Manual docstrings are preferred in learning and exams.

AI-generated docstrings:

- AI-generated docstrings are created automatically by AI tools.
- AI docstrings are usually clear and well-structured.
- AI-generated docstrings save time and effort.
- AI-generated docstrings are consistent and standard.
- AI-generated docstrings are useful in real-world projects.