ASSIGNMENT-9.3

Name:E.Ramya

Ht.no:2303A51282

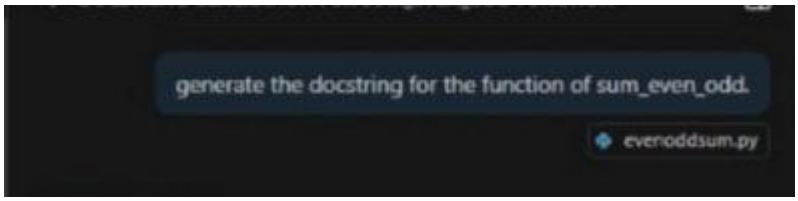
Batch:05

Task 1: Basic Docstring Generation.

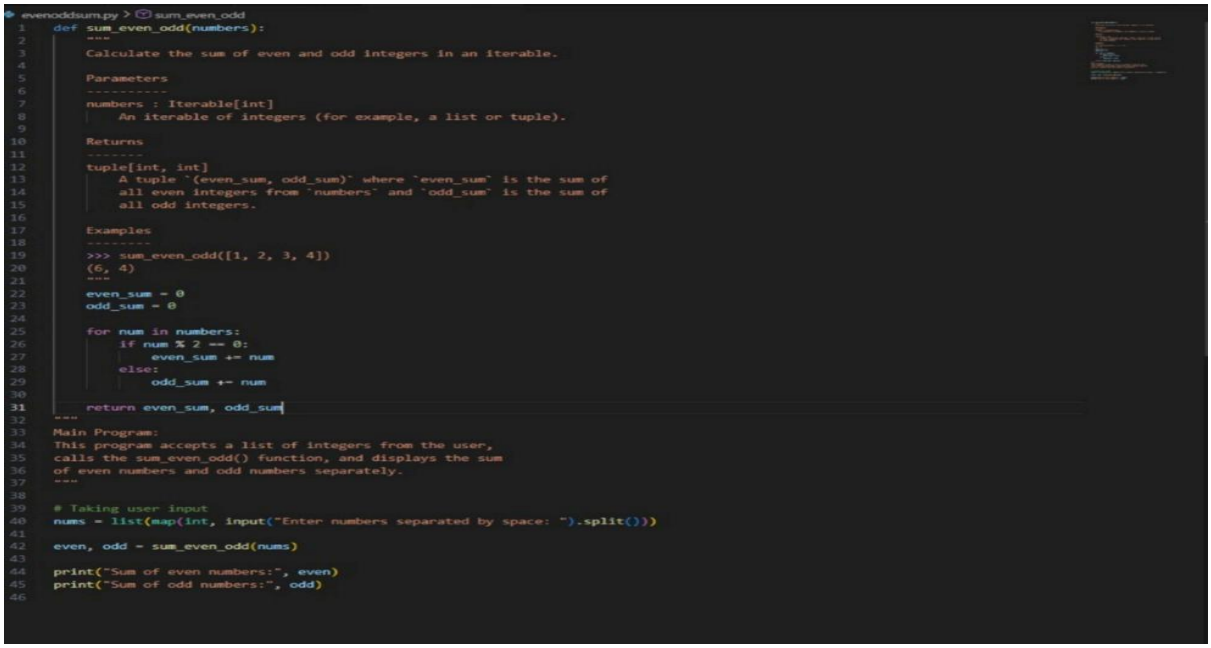
Scenario

You are developing a utility function that processes numerical lists and must be properly documented for future maintenance.

Prompt to generate function-level docstring



Code:



```
evenoddsum.py > sum_even_odd
def sum_even_odd(numbers):
    """
    Calculate the sum of even and odd integers in an iterable.

    Parameters
    ----------
    numbers : Iterable[int]
        An iterable of integers (for example, a list or tuple).

    Returns
    -------
    tuple[int, int]
        A tuple `(even_sum, odd_sum)` where `even_sum` is the sum of
        all even integers from `numbers` and `odd_sum` is the sum of
        all odd integers.

    Examples
    --------
    >>> sum_even_odd([1, 2, 3, 4])
    (6, 4)
    """
    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
"""
Main Program:
This program accepts a list of integers from the user,
calls the sum_even_odd() function, and displays the sum
of even numbers and odd numbers separately.
"""

# Taking user input
nums = list(map(int, input("Enter numbers separated by space: ").split()))

even, odd = sum_even_odd(nums)

print("Sum of even numbers:", even)
print("Sum of odd numbers:", odd)
```

Compare the AI-generated docstring with the manually written docstring.

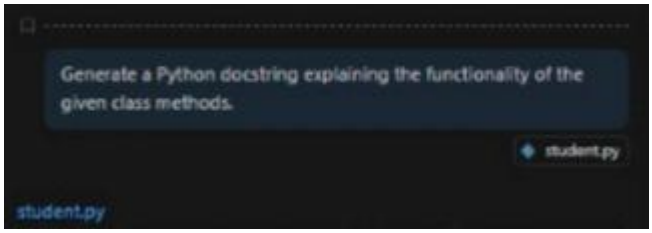
- A manual docstring is written by the programmer in their own words.
- An AI-generated docstring is written automatically by an AI tool.
- Manual docstrings may be short or long depending on the person.
- AI-generated docstrings are usually well-structured and detailed.
- Writing a manual docstring takes more time.
- AI-generated docstrings save time and effort.
- Manual docstrings show the programmer's understanding of the code.
- AI-generated docstrings depend on the quality of the prompt given.
- Manual docstrings are commonly used in exams and learning.
- AI-generated docstrings are useful in real projects for consistency.

Task 2: Automatic Inline Comments

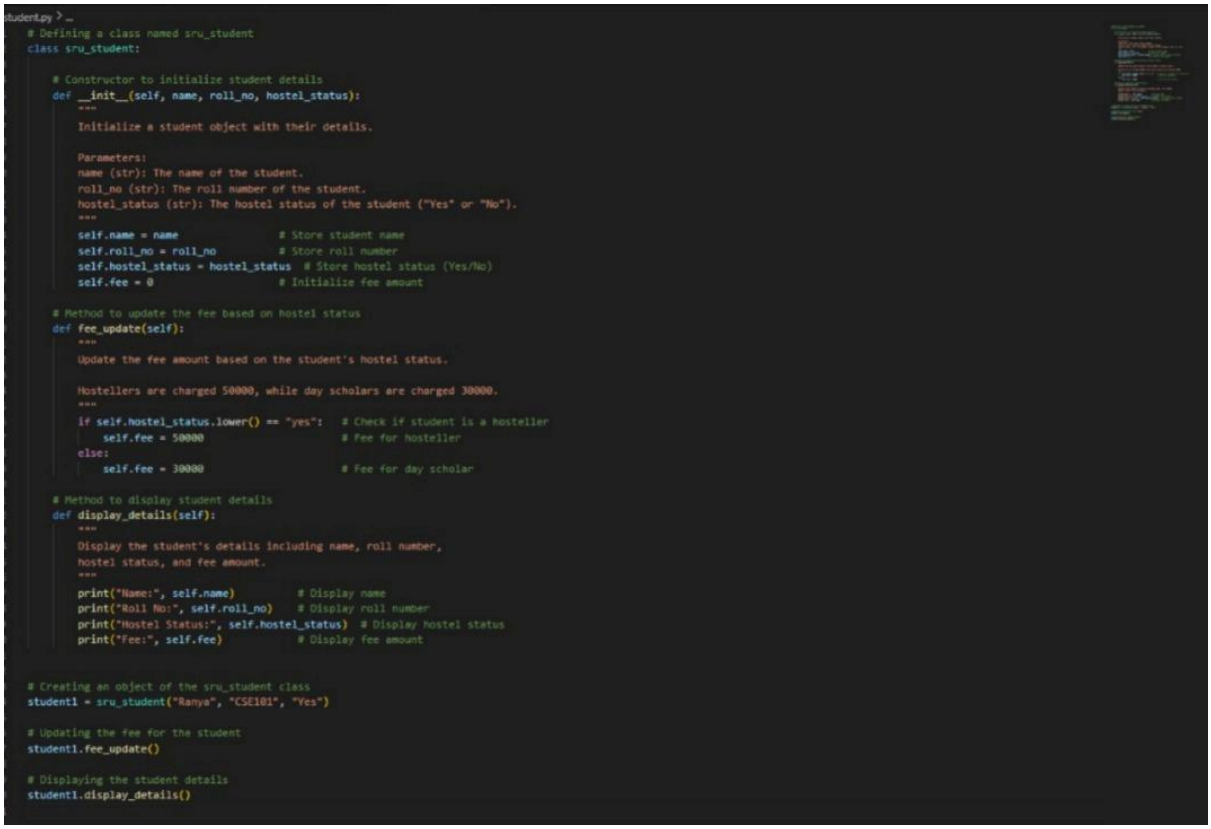
Scenario

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

Prompt to generate function-level docstring:



Code:



```
student.py > ...
# Defining a class named sru_student
class sru_student:

    # Constructor to initialize student details
    def __init__(self, name, roll_no, hostel_status):
        """
        Initialize a student object with their details.

        Parameters:
        name (str): The name of the student.
        roll_no (str): The roll number of the student.
        hostel_status (str): The hostel status of the student ("Yes" or "No").
        """
        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 amount

    # Method to update the fee based on hostel status
    def fee_update(self):
        """
        Update the fee amount based on the student's hostel status.

        Hostellers are charged 50000, while day scholars are charged 30000.
        """
        if self.hostel_status.lower() == "yes":   # Check if student is a hosteller
            self.fee = 50000                      # Fee for hosteller
        else:
            self.fee = 30000                      # Fee for day scholar

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


# Creating an object of the sru_student class
student1 = sru_student("Ranya", "CSE101", "Yes")

# Updating the fee for the student
student1.fee_update()

# Displaying the student details
student1.display_details()
```

Compare the AI-generated docstring with the manually written docstring.

- In the manual version, each line and block in the sru_student program is commented by the programmer to explain what it does.
- In the AI-generated version, comments are added automatically based on code understanding.
- Manual comments clearly show the student's understanding of class, constructor, and methods like fee_update() and display_details().

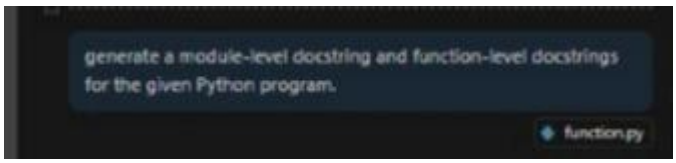
- AI comments explain the same logic but in a more standardized and polished way.
- Writing manual comments for the program takes more time, especially for constructors and methods.
- AI-generated comments reduce effort and save time.
- Manual comments may be simple and exam-oriented.
- AI comments may be slightly more descriptive, sometimes more than required for exams.
- Manual comments are preferred in academic submissions to prove learning.
- AI-generated comments are useful in real projects where clarity and consistency matter.

<u>Task 3:</u> Module-Level and Function-Level Documentation

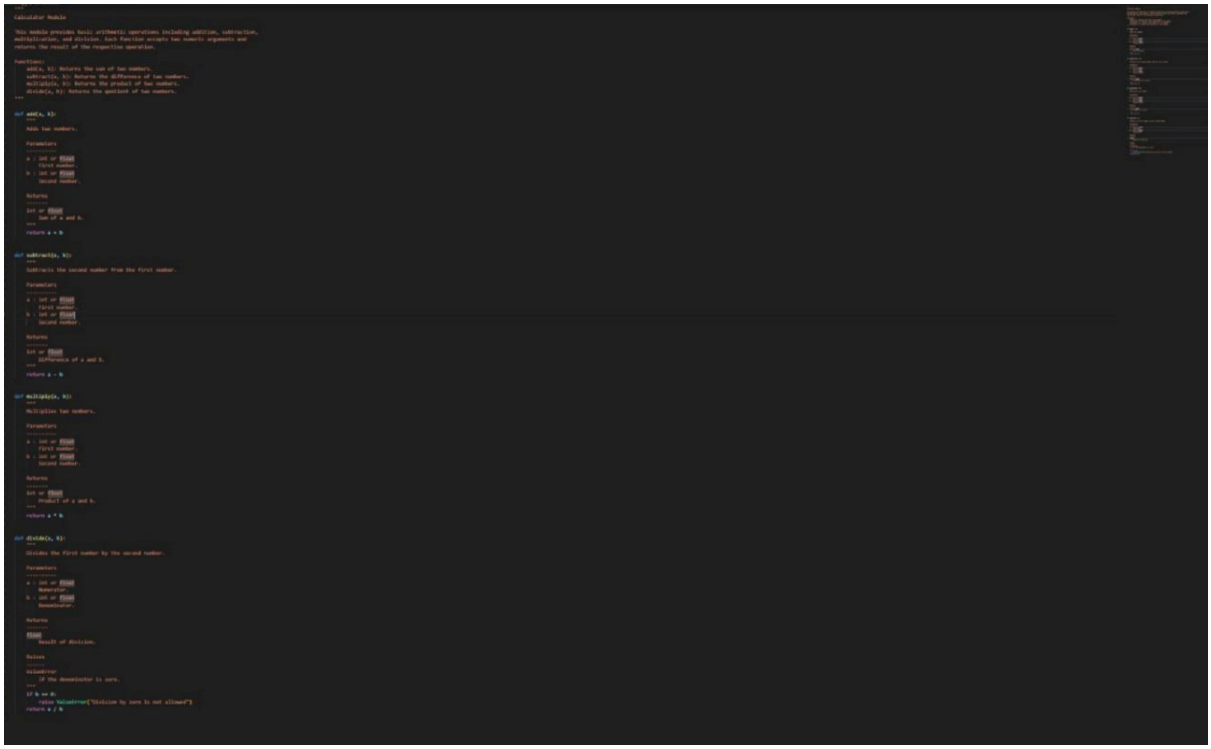
Scenario

You are building a small calculator module that will be shared across multiple projects and
requires structured documentation.

Prompt to generate function-level docstring:



Code:



Compare the AI-generated docstring with the manually written docstring.

- Manual docstrings are written by the programmer.
- AI-generated docstrings are created automatically by AI tools.
- Manual docstrings show the programmer's understanding of the code.
- AI docstrings are usually clear and well-structured.
- Writing manual docstrings takes more time.
- AI-generated docstrings save time and effort.
- Manual docstrings may differ in quality and style.
- AI-generated docstrings are consistent and standard.
- Manual docstrings are preferred in learning and exams.
- AI-generated docstrings are useful in real-world projects.