LAB ASSIGNMET

NAME:M.TANVI

ROLLNO:2403A510A4

COURSE:ASSISTED CODING

BATCH:01

QUESTION

Task Description#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.

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.
- 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.

Expected Output#2: Students critically analyze AI-generated code comments.

Task Description#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.

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

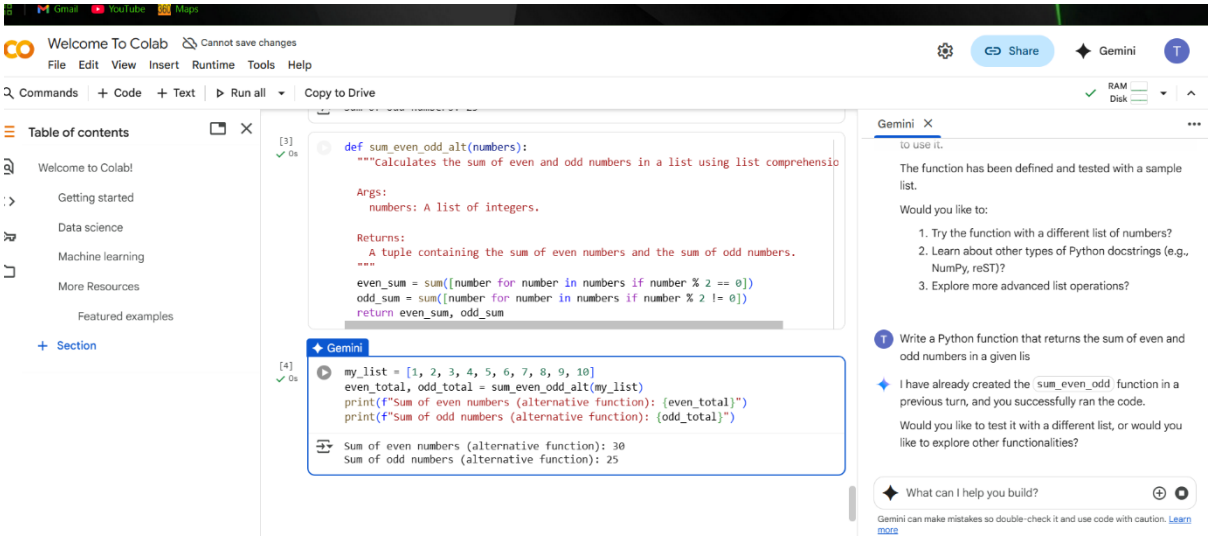
Push documentation whole workspace as .md file in GitHub Repository

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

TASK 1

AI PROMPT

Generate a Google-style docstring for a Python function that returns the sum of even and odd numbers from a given list.



MANNUAL

code

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


  Args:
    numbers: A list of integers.


  Returns:
    A tuple containing the sum of even numbers and the sum of odd numbers.
```

```
  """
  even_sum = 0
  odd_sum = 0
  for number in numbers:
    if number % 2 == 0:
      even_sum += number
    else:
      odd_sum += number
  return even_sum, odd_sum
```

OUTPUT

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_total, odd_total = sum_even_odd(my_list)
print(f"Sum of even numbers: {even_total}")
print(f"Sum of odd numbers: {odd_total}")
```

```
Sum of even numbers: 30
Sum of odd numbers: 25
```

EXPLAINATION :

The manually written docstring is more precise and follows the Google style guide, clearly specifying input types (list[int]) and return values (tuple[int, int]). It also explicitly describes the order of returned values (even sum, odd sum). In contrast, the AI-generated docstring is shorter and more generic, using basic terms like list and tuple without

specifying types or the order of the return values. While the AI version is quick and readable, the manual version is more detailed and professional.

WHICH ONE IS BETTER

COMAPARISION :

Mannual code is the best code compared to ai generator

It includes clear, detailed documentation following the Google style, which improves readability and maintainability.

It precisely defines argument types and return values.

It explicitly states the order of the returned tuple (even sum first, odd sum second).

TASK 2

AI PROMPT

Write a Python class named sru_student with the attributes name, roll_no, and hostel_status. Include a fee_update method and a display_details method. Manually add inline comments to explain each line or code block. Give me the correct output without errors

MANNUAL

CODE

```
class sru_student:
    # Constructor to initialize the student object
    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 provided hostel status to the student object
        self.fees_paid = 0  # Initialize fees paid to 0 for a new student

    # Method to update the fees paid by the student
    def fee_update(self, amount):
        self.fees_paid += amount  # Add the paid amount to the total fees paid

    # Method to display the details of the student
    def display_details(self):
        print(f"Student Name: {self.name}")  # Print the student's name
        print(f"Roll Number: {self.roll_no}")  # Print the student's roll number
        print(f"Hostel Status: {self.hostel_status}")  # Print the student's hostel status
        print(f"Fees Paid: {self.fees_paid}")  # Print the total fees paid by the student
```

OUTPUT

Example of how to create a student object and use its methods

student1 = sru_student("Alice", "SRU123", "Resident")

student1.fee_update(5000)

student1.display_details()

EXPLAINATION

COMAPARISION :

Mannual code is the best code compared to ai generator