ASSIGNMENT:9.3_3

HTNO:2403A51284

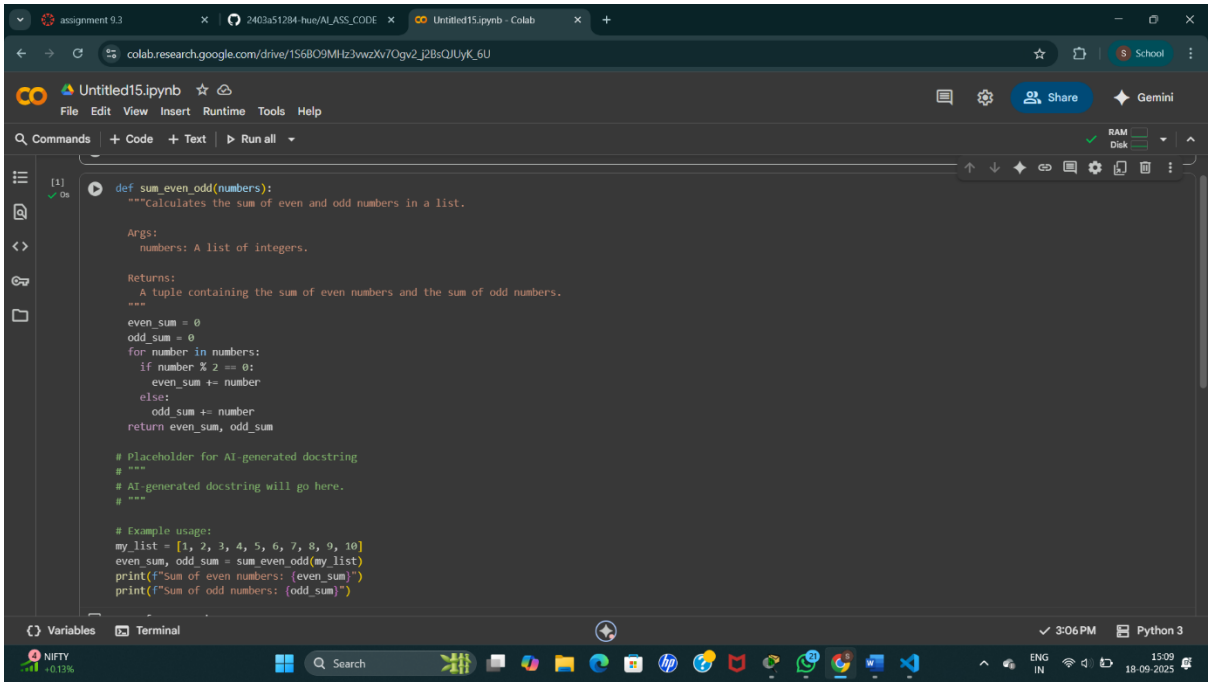
Task Description#1 Basic Docstring Generation

- 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

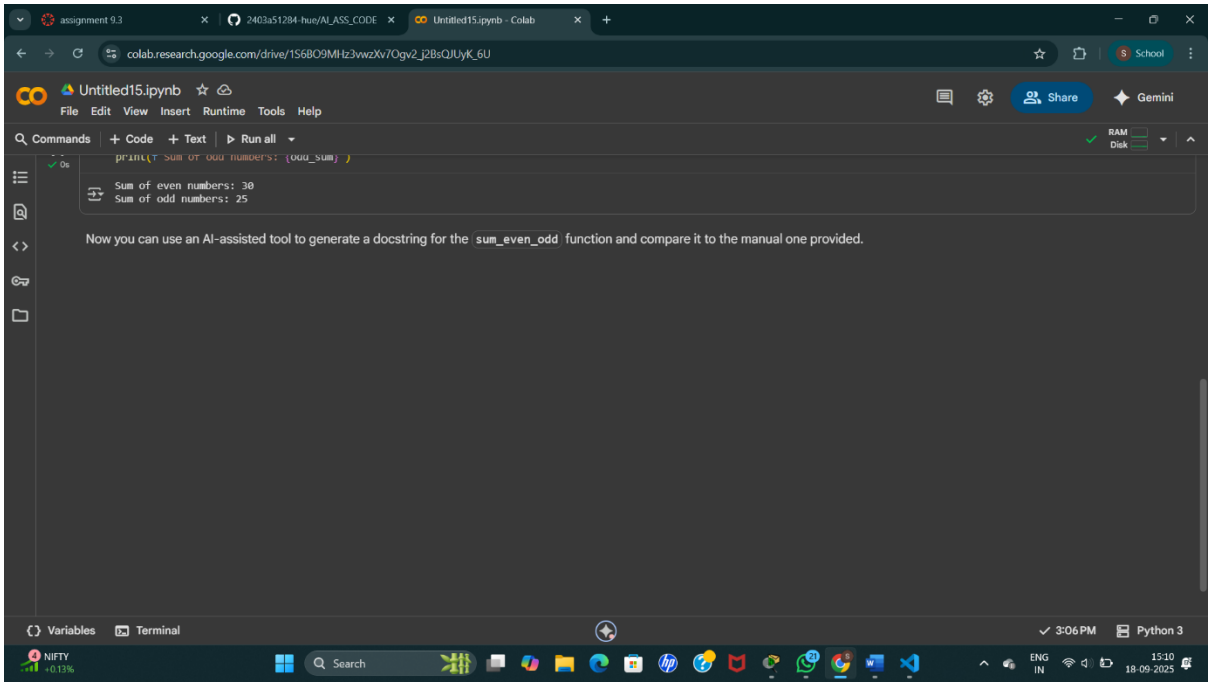
PROMPT:

Generate a Google-style docstring for the following Python function that returns the sum of even and odd numbers in a given list.
Compare the AI-generated docstring with a manually written Google-style docstring for the same function

CODE:



OUTPUT:



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.

PROMPT:

Add inline comments to the following Python program for the sru_student class, explaining each line or code block.

The class includes attributes like name, roll_no, hostel_status, a fee_update method, and a display_details method.

Compare the AI-generated comments with manually written comments for accuracy and clarity.

CODE:

```python
class sru_student:
    # Constructor to initialize the student object
    def __init__(self, name, roll_no, hostel_status):
        self.name = name  # Assign the name
        self.roll_no = roll_no  # Assign the roll number
        self.hostel_status = hostel_status  # Assign the hostel status
        self.fee_paid = False  # Initialize fee status to unpaid

    # Method to update fee status
    def fee_update(self):
        self.fee_paid = True  # Set fee status to paid
        print(f"Fee status for {self.name} updated to paid.") # Print confirmation

    # Method to display student details
    def display_details(self):
        print(f"Student Name: {self.name}") # Print student name
        print(f"Roll Number: {self.roll_no}") # Print roll number
        print(f"Hostel Status: {self.hostel_status}") # Print hostel status
        print(f"Fee Paid: {self.fee_paid}") # Print fee status

# Placeholder for AI-generated inline comments
# # AI-generated comments will go here.
# class sru_student:
#     def __init__(self, name, roll_no, hostel_status):
#         self.name = name
#         self.roll_no = roll_no
#         self.hostel_status = hostel_status
#         self.fee_paid = False

#     def fee_update(self):
```

```python
        print(f"Student Name: {self.name}") # Print student name
        print(f"Roll Number: {self.roll_no}") # Print roll number
        print(f"Hostel Status: {self.hostel_status}") # Print hostel status
        print(f"Fee Paid: {self.fee_paid}") # Print fee status

# Placeholder for AI-generated inline comments
# # AI-generated comments will go here.
# class sru_student:
#     def __init__(self, name, roll_no, hostel_status):
#         self.name = name
#         self.roll_no = roll_no
#         self.hostel_status = hostel_status
#         self.fee_paid = False

#     def fee_update(self):
#         self.fee_paid = True
#         print(f"Fee status for {self.name} updated to paid.")

#     def display_details(self):
#         print(f"Student Name: {self.name}")
#         print(f"Roll Number: {self.roll_no}")
#         print(f"Hostel Status: {self.hostel_status}")
#         print(f"Fee Paid: {self.fee_paid}")


# Example usage:
student1 = sru_student("Alice", "SRU123", "Resident")
student1.display_details()
student1.fee_update()
student1.display_details()
```

OUTPUT:

```
Student Name: Alice
Roll Number: SRU123
Hostel Status: Resident
Fee Paid: False
Fee status for Alice updated to paid.
Student Name: Alice
Roll Number: SRU123
Hostel Status: Resident
Fee Paid: True
```

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.

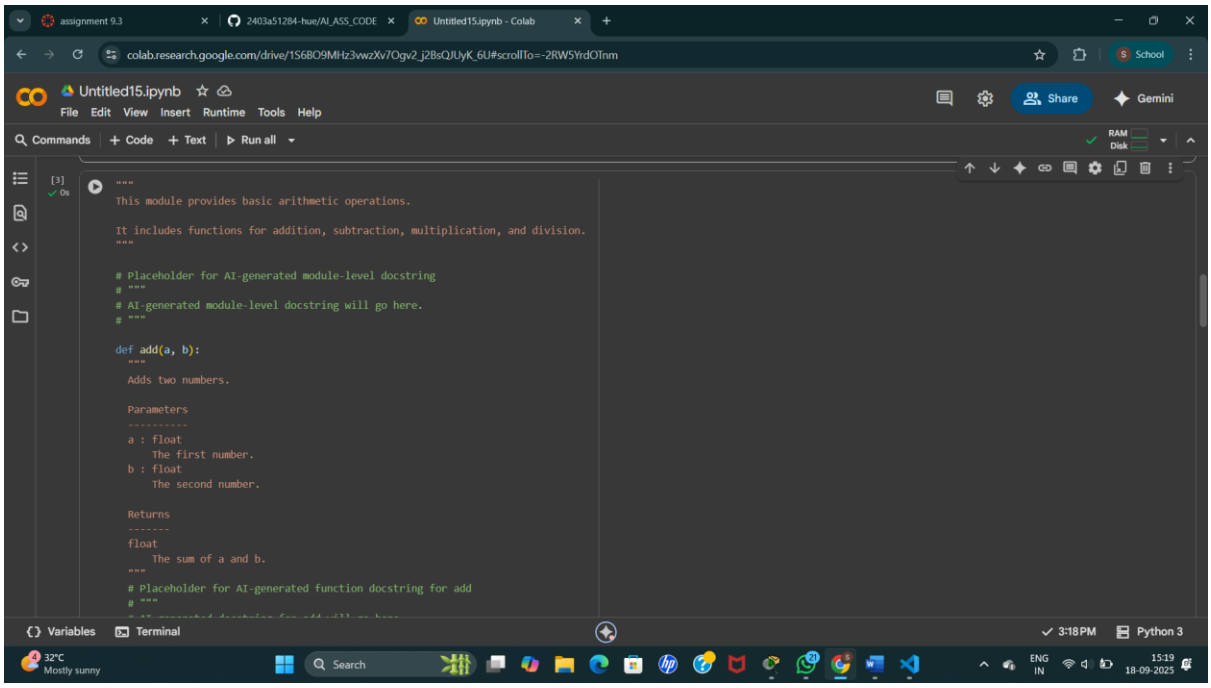
• Compare the AI-generated docstring with your manually written one.

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

PROMPT:

Generate a module-level docstring and individual function docstrings in NumPy style for the following Python script containing calculator functions (add, subtract, multiply, divide).
Compare the AI-generated docstrings with manually written NumPy-style docstrings for accuracy, structure, and completeness.

CODE:



```
    # Placeholder for AI-generated function docstring for add
    # """
    # AI-generated docstring for add will go here.
    # """
    return a + b

def subtract(a, b):
    """
    Subtracts the second number from the first.

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

    Returns
    -------
    float
        The difference between a and b.
    """
    # Placeholder for AI-generated function docstring for subtract
    # """
    # AI-generated docstring for subtract will go here.
    # """
    return a - b

def multiply(a, b):
    """
```

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

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

    Returns
    -------
    float
        The product of a and b.
    """
    # Placeholder for AI-generated function docstring for multiply
    # """
    # AI-generated docstring for multiply will go here.
    # """
    return a * b

def divide(a, b):
    """
    Divides the first number by the second.

    Parameters
    ----------
    a : float
        The dividend.
```

```python
def divide(a, b):
    """
    Divides the first number by the second.

    Parameters
    ----------
    a : float
        The dividend.
    b : float
        The divisor.

    Returns
    -------
    float
        The quotient of a and b.

    Raises
    ------
    ZeroDivisionError
        If the divisor is zero.
    """
    # Placeholder for AI-generated function docstring for divide
    # """
    # AI-generated docstring for divide will go here.
    # """
    if b == 0:
      raise ZeroDivisionError("Cannot divide by zero")
    return a / b

# Example Usage:
```

```python
    Returns
    -------
    float
        The quotient of a and b.

    Raises
    ------
    ZeroDivisionError
        If the divisor is zero.
    """
    # Placeholder for AI-generated function docstring for divide
    # """
    # AI-generated docstring for divide will go here.
    # """
    if b == 0:
      raise ZeroDivisionError("Cannot divide by zero")
    return a / b

# Example Usage:
num1 = 10
num2 = 5

print(f"{num1} + {num2} = {add(num1, num2)}")
print(f"{num1} - {num2} = {subtract(num1, num2)}")
print(f"{num1} * {num2} = {multiply(num1, num2)}")
print(f"{num1} / {num2} = {divide(num1, num2)}")

# Example with division by zero
# print(f"{num1} / 0 = {divide(num1, 0)}") # Uncomment to test ZeroDivisionError
```

OUTPUT:

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
```

Now you can use an AI-assisted tool to generate module-level and function docstrings for this code and compare them with the manual ones provided.

Now you can use an AI-assisted tool to add inline comments to the placeholder code block and compare them with the manually written comments.