ASSIGNMENT 9

**TASK 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.

**GEMINI AI :**

**WITHOUT COMMENTS :**

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

# Example usage:

my\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even\_sum, odd\_sum = sum\_even\_odd(my\_list)

print(f"Sum of even numbers: {even\_sum}")

print(f"Sum of odd numbers: {odd\_sum}")

**WITH COMMENTS :**

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  # Initialize sum for even numbers

  odd\_sum = 0   # Initialize sum for odd numbers

  for number in numbers: # Iterate through the list

    if number % 2 == 0:  # Check if the number is even

      even\_sum += number # Add to even sum

    else: # If not even, it's odd

      odd\_sum += number  # Add to odd sum

  return even\_sum, odd\_sum # Return both sums

# Example usage:

my\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even\_sum, odd\_sum = sum\_even\_odd(my\_list)

print(f"Sum of even numbers: {even\_sum}")

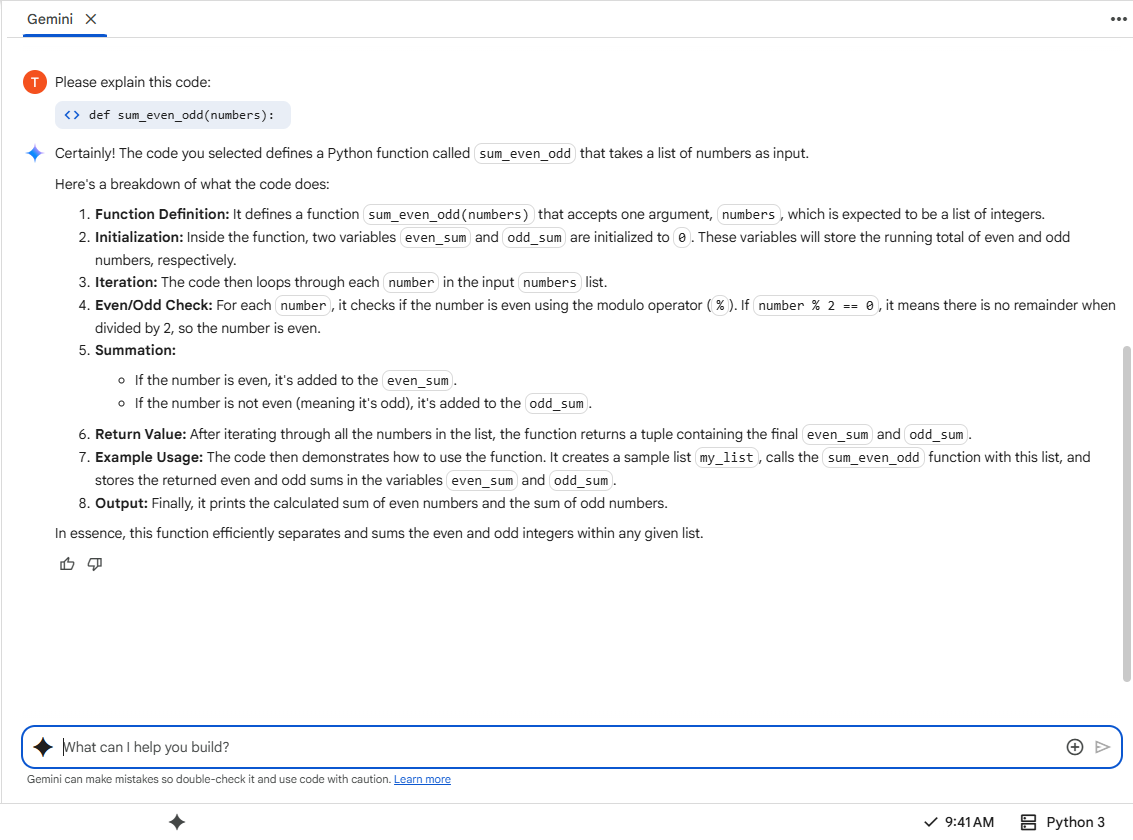
print(f"Sum of odd numbers: {odd\_sum}")

**OUTPUT:**

Sum of even numbers: 30

Sum of odd numbers: 25

**EXPLAINATION :**



**COPILOT:**

**WITHOUT COMMENTS :**

# Function to return sum of even and odd numbers in a given list

def sum\_even\_odd(numbers):

    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

**WITH COMMENTS :**

# Function to return sum of even and odd numbers in a given list

def sum\_even\_odd(numbers):

    even\_sum = 0  # Initialize sum for even numbers

    odd\_sum = 0   # Initialize sum for odd numbers

    for num in numbers:  # Iterate through each number in the list

        if num % 2 == 0:  # Check if the number is even

            even\_sum += num  # Add to even sum if even

        else:

            odd\_sum += num  # Add to odd sum if odd

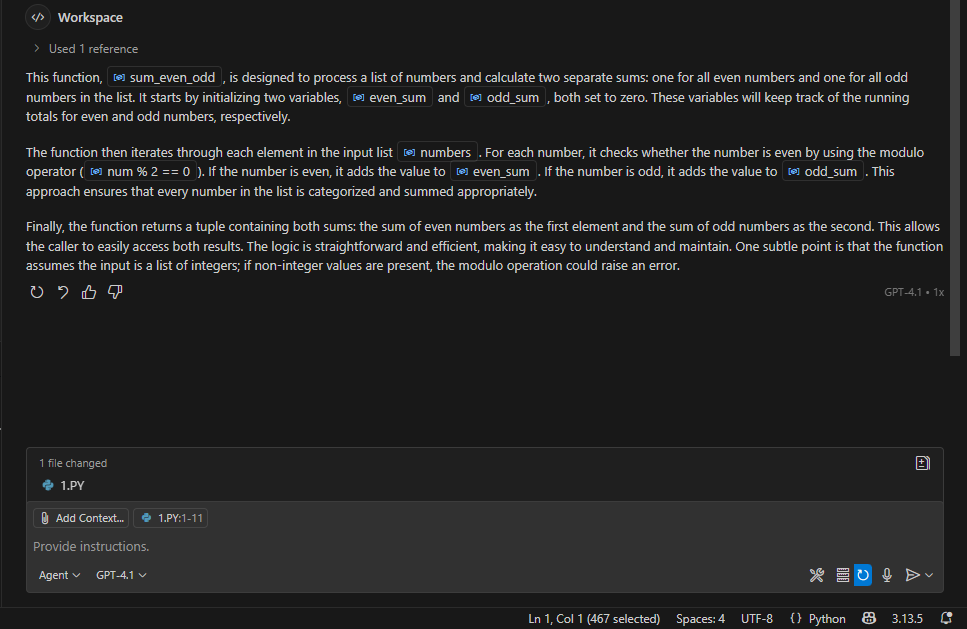
    return even\_sum, odd\_sum  # Return both sums as a tuple

**OUTPUT :**

[Running] python -u "c:\ASSIGNMENT 9\1.PY"

[Done] exited with code=0 in 0.084 seconds

**EXPLAINATION :**



**TASK 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

**GEMINI AI :**

**WITHOUT COMMENTS :**

class sru\_student:

  """

  Represents a student at SRU with attributes for name, roll number,

  hostel status, and methods for fee updates and displaying details.

  """

  def \_\_init\_\_(self, name, roll\_no, hostel\_status=False):

    """

    Initializes a new sru\_student object.

    Args:

      name (str): The name of the student.

      roll\_no (str): The roll number of the student.

      hostel\_status (bool, optional): The hostel status of the student. Defaults to False.

    """

    self.name = name

    self.roll\_no = roll\_no

    self.hostel\_status = hostel\_status

    self.fees\_paid = 0 # Initialize fees paid to 0

  def fee\_update(self, amount):

    """

    Updates the amount of fees paid by the student.

    Args:

      amount (float): The amount of fees to add.

    """

    if amount > 0:

      self.fees\_paid += amount

      print(f"{self.name}'s fees updated by {amount}. Total fees paid: {self.fees\_paid}")

    else:

      print("Fee amount must be positive.")

  def display\_details(self):

    """

    Displays the details of the student.

    """

    print("\n--- Student Details ---")

    print(f"Name: {self.name}")

    print(f"Roll No: {self.roll\_no}")

    print(f"Hostel Status: {'In Hostel' if self.hostel\_status else 'Day Scholar'}")

    print(f"Fees Paid: {self.fees\_paid}")

    print("-----------------------")

# Example Usage:

student1 = sru\_student("Alice", "SRU101", True)

student2 = sru\_student("Bob", "SRU102")

student1.display\_details()

student2.display\_details()

student1.fee\_update(5000)

student2.fee\_update(3000)

student1.fee\_update(2000)

student1.display\_details()

student2.display\_details()

**WITH COMMENTS :**

class sru\_student:

  """

  Represents a student at SRU with attributes for name, roll number,

  hostel status, and methods for fee updates and displaying details.

  """

  def \_\_init\_\_(self, name, roll\_no, hostel\_status=False):

    """

    Initializes a new sru\_student object.

    Args:

      name (str): The name of the student.

      roll\_no (str): The roll number of the student.

      hostel\_status (bool, optional): The hostel status of the student. Defaults to False.

    """

    self.name = name  # Store the student's name

    self.roll\_no = roll\_no # Store the student's roll number

    self.hostel\_status = hostel\_status # Store the student's hostel status (True if in hostel, False otherwise)

    self.fees\_paid = 0 # Initialize fees paid to 0 for a new student

  def fee\_update(self, amount):

    """

    Updates the amount of fees paid by the student.

    Args:

      amount (float): The amount of fees to add.

    """

    if amount > 0: # Check if the fee amount is positive

      self.fees\_paid += amount # Add the positive amount to the total fees paid

      print(f"{self.name}'s fees updated by {amount}. Total fees paid: {self.fees\_paid}") # Print a confirmation message

    else:

      print("Fee amount must be positive.") # Print an error message if the amount is not positive

  def display\_details(self):

    """

    Displays the details of the student.

    """

    print("\n--- Student Details ---") # Print a header for the student details

    print(f"Name: {self.name}") # Print the student's name

    print(f"Roll No: {self.roll\_no}") # Print the student's roll number

    print(f"Hostel Status: {'In Hostel' if self.hostel\_status else 'Day Scholar'}") # Print the hostel status (In Hostel or Day Scholar)

    print(f"Fees Paid: {self.fees\_paid}") # Print the total fees paid by the student

    print("-----------------------") # Print a separator line

# Example Usage:

student1 = sru\_student("Alice", "SRU101", True) # Create the first student object (in hostel)

student2 = sru\_student("Bob", "SRU102") # Create the second student object (day scholar)

student1.display\_details() # Display details for student1

student2.display\_details() # Display details for student2

student1.fee\_update(5000) # Update fees for student1

student2.fee\_update(3000) # Update fees for student2

student1.fee\_update(2000) # Update fees for student1 again

student1.display\_details() # Display updated details for student1

student2.display\_details() # Display updated details for student2

**output :**

--- Student Details ---

Name: Alice

Roll No: SRU101

Hostel Status: In Hostel

Fees Paid: 0

-----------------------

--- Student Details ---

Name: Bob

Roll No: SRU102

Hostel Status: Day Scholar

Fees Paid: 0

-----------------------

Alice's fees updated by 5000. Total fees paid: 5000

Bob's fees updated by 3000. Total fees paid: 3000

Alice's fees updated by 2000. Total fees paid: 7000

--- Student Details ---

Name: Alice

Roll No: SRU101

Hostel Status: In Hostel

Fees Paid: 7000

-----------------------

--- Student Details ---

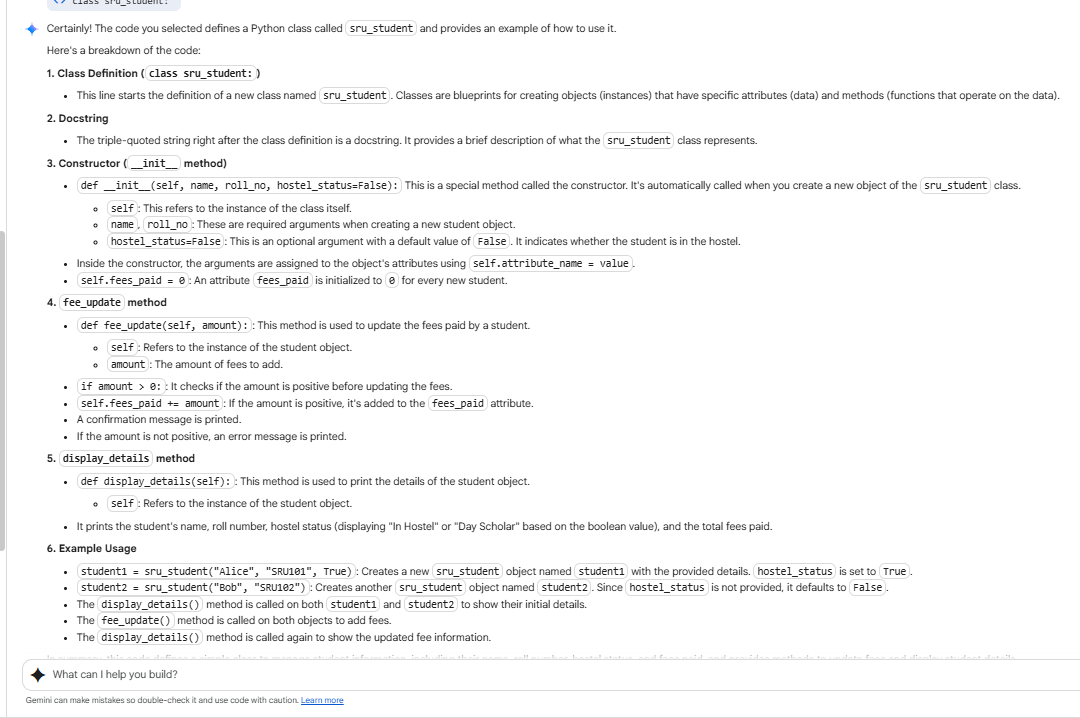
Name: Bob

Roll No: SRU102

Hostel Status: Day Scholar

Fees Paid: 3000

**EXPLAINATION :**



**COPILOT:**

**WITHOUT COMMENTS :**

# Class representing an SRU student

class SRUStudent:

    def \_\_init\_\_(self, name, roll\_no, hostel\_status, fee\_paid=False):

        self.name = name

        self.roll\_no = roll\_no

        self.hostel\_status = hostel\_status  # True for hostel, False for day scholar

        self.fee\_paid = fee\_paid

    def fee\_update(self, status):

        """Update the fee payment status."""

        self.fee\_paid = status

    def display\_details(self):

        """Display the student's details."""

        print(f"Name: {self.name}")

        print(f"Roll No.: {self.roll\_no}")

        print(f"Hostel Status: {'Hosteller' if self.hostel\_status else 'Day Scholar'}")

        print(f"Fee Paid: {'Yes' if self.fee\_paid else 'No'}")

**WITH COMMENTS :**

# Class representing an SRU student

class SRUStudent:

    def \_\_init\_\_(self, name, roll\_no, hostel\_status, fee\_paid=False):

        # Initialize the student's name

        self.name = name

        # Initialize the student's roll number

        self.roll\_no = roll\_no

        # Initialize hostel status: True for hostel, False for day scholar

        self.hostel\_status = hostel\_status  # True for hostel, False for day scholar

        # Initialize fee payment status (default is False)

        self.fee\_paid = fee\_paid

    def fee\_update(self, status):

        """Update the fee payment status."""

        # Update the fee\_paid attribute with the new status

        self.fee\_paid = status

    def display\_details(self):

        """Display the student's details."""

        # Print the student's name

        print(f"Name: {self.name}")

        # Print the student's roll number

        print(f"Roll No.: {self.roll\_no}")

        # Print hostel status as 'Hosteller' or 'Day Scholar'

        print(f"Hostel Status: {'Hosteller' if self.hostel\_status else 'Day Scholar'}")

        # Print whether the fee has been paid

        print(f"Fee Paid: {'Yes' if self.fee\_paid else 'No'}")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    # Create an SRUStudent object

    student = SRUStudent(name="Rahul", roll\_no=101, hostel\_status=True)

    # Display initial details

    print("Initial details:")

    student.display\_details()

    # Update fee status

    student.fee\_update(True)

    # Display updated details

    print("\nAfter fee update:")

    student.display\_details()

**OUTPUT :**

[Running] python -u "c:\ASSIGNMENT 9\1.PY"

Initial details:

Name: Rahul

Roll No.: 101

Hostel Status: Hosteller

Fee Paid: No

After fee update:

Name: Rahul

Roll No.: 101

Hostel Status: Hosteller

Fee Paid: Yes

[Done] exited with code=0 in 0.043 seconds

**Task 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

**GEMINI AI :**

**WITHOUT COMMENTS :**

def add(x, y):

  """Adds two numbers."""

  return x + y

def subtract(x, y):

  """Subtracts the second number from the first."""

  return x - y

def multiply(x, y):

  """Multiplies two numbers."""

  return x \* y

def divide(x, y):

  """Divides the first number by the second. Handles division by zero."""

  if y == 0:

    return "Error: Division by zero!"

  return x / y

# 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)}")

num3 = 10

num4 = 0

print(f"{num3} / {num4} = {divide(num3, num4)}") # Example of division by zero

**WITH COMMENTS :**

def subtract(x, y):

  """Subtracts the second number from the first."""

  return x - y # Return the difference between x and y

def multiply(x, y):

  """Multiplies two numbers."""

  return x \* y # Return the product of x and y

def divide(x, y):

  """Divides the first number by the second. Handles division by zero."""

  if y == 0: # Check if the denominator is zero

    return "Error: Division by zero!" # Return an error message if division by zero occurs

  return x / y # Return the result of the division

# Example Usage:

num1 = 10 # Assign the value 10 to num1

num2 = 5 # Assign the value 5 to num2

print(f"{num1} + {num2} = {add(num1, num2)}") # Call the add function and print the result

print(f"{num1} - {num2} = {subtract(num1, num2)}") # Call the subtract function and print the result

print(f"{num1} \* {num2} = {multiply(num1, num2)}") # Call the multiply function and print the result

print(f"{num1} / {num2} = {divide(num1, num2)}") # Call the divide function and print the result

num3 = 10 # Assign the value 10 to num3

num4 = 0 # Assign the value 0 to num4

print(f"{num3} / {num4} = {divide(num3, num4)}") # Call the divide function with num3 and num4 (demonstrating division by zero handling)

**OUTPUT:**

10 + 5 = 15

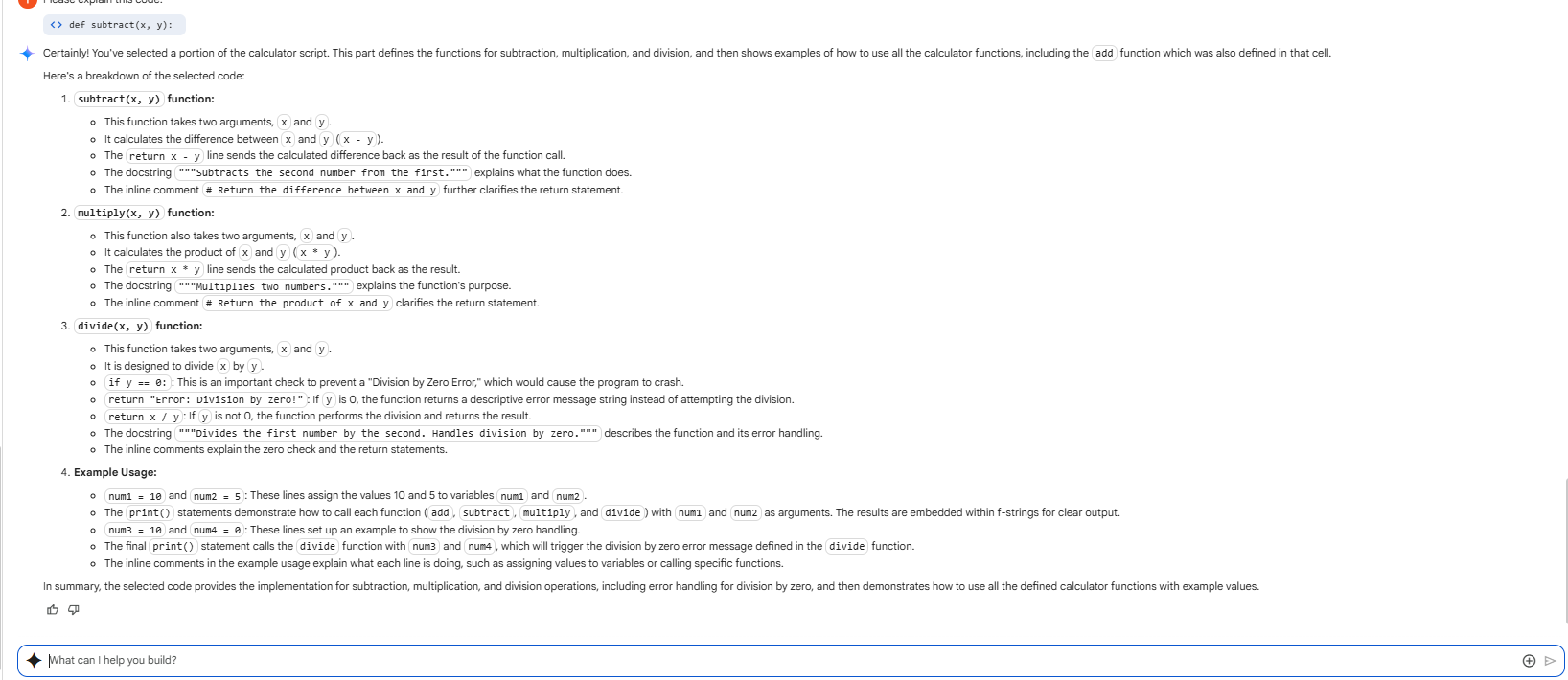
10 - 5 = 5

10 \* 5 = 50

10 / 5 = 2.0

10 / 0 = Error: Division by zero!

**EXPLAINATION :**



**COPILOT:**

**WITHOUT COMMENTS :**

# Simple Calculator Functions

def add(a, b):

    """Return the sum of a and b."""

    return a + b

def subtract(a, b):

    """Return the difference of a and b."""

    return a - b

def multiply(a, b):

    """Return the product of a and b."""

    return a \* b

def divide(a, b):

    """Return the quotient of a and b. Raises ValueError if b is zero."""

    if b == 0:

        raise ValueError("Cannot divide by zero.")

    return a / b

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    x = 10

    y = 5

    print(f"{x} + {y} = {add(x, y)}")

    print(f"{x} - {y} = {subtract(x, y)}")

    print(f"{x} \* {y} = {multiply(x, y)}")

    print(f"{x} / {y} = {divide(x, y)}")

**WITH COMMENTS :**

# Simple Calculator Functions

def add(a, b):

    """Return the sum of a and b."""

    # Add two numbers

    return a + b

def subtract(a, b):

    """Return the difference of a and b."""

    # Subtract second number from first

    return a - b

def multiply(a, b):

    """Return the product of a and b."""

    # Multiply two numbers

    return a \* b

def divide(a, b):

    """Return the quotient of a and b. Raises ValueError if b is zero."""

    # Check for division by zero

    if b == 0:

        raise ValueError("Cannot divide by zero.")

    # Divide first number by second

    return a / b

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    # Define two numbers

    x = 10

    y = 5

    # Print the result of addition

    print(f"{x} + {y} = {add(x, y)}")

    # Print the result of subtraction

    print(f"{x} - {y} = {subtract(x, y)}")

    # Print the result of multiplication

    print(f"{x} \* {y} = {multiply(x, y)}")

    # Print the result of division

    print(f"{x} / {y} = {divide(x, y)}")

**OUTPUT :**

[Running] python -u "c:\ASSIGNMENT 9\calculator.py"

10 + 5 = 15

10 - 5 = 5

10 \* 5 = 50

10 / 5 = 2.0

[Done] exited with code=0 in 0.07 seconds