LAB ASSESMENT 9.3

PROMPT:  Write python function to return sum of even and odd numbers in the given list

CODE AND OUTPUT : GEMINI AI:

def sum\_even\_odd(numbers):

  """

  Calculates the sum of even and odd numbers in a list.

  Args:

    numbers: A list of numbers.

  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}")

Sum of even numbers: 30

Sum of odd numbers: 25

CODE AND OUTPUT IN GITHUB CO-POILOT:

def sum\_even\_odd(nums):

    """

    Return a tuple (even\_sum, odd\_sum) for the given iterable of integers.

    Non-integer values and booleans are ignored.

    """

    even = sum(x for x in nums if isinstance(x, int) and not isinstance(x, bool) and x % 2 == 0)

    odd = sum(x for x in nums if isinstance(x, int) and not isinstance(x, bool) and x % 2 != 0)

    return even, odd

# Example:

# ev, od = sum\_even\_odd([1,2,3,4,5,6])

# ev == 12, od == 9

# Replaced duplicate definitions with one function and an example run:

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

    sample = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, "x", 3.5, True, False]

    ev\_sum, od\_sum = sum\_even\_odd(sample)

    print("Input:", sample)

    print("Even sum:", ev\_sum)

    print("Odd sum:", od\_sum)

Input: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 'x', 3.5, True, False]

Even sum: 30

Odd sum: 25

PROMPT: Write python program for sru\_student class with attributes like name, roll no., hostel\_status and fee\_update method and display\_details method.

CODE AND OUTPUT IN GEMIN AI:

class sru\_student:

  """

  Represents a student at SRU.

  Attributes:

    name (str): The name of the student.

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

    hostel\_status (bool): True if the student is in a hostel, False otherwise.

    fee\_paid (float): The amount of fees paid by the 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 = 0.0  # Initialize fee\_paid to 0

  def fee\_update(self, amount):

    """

    Updates the fee paid by the student.

    Args:

      amount (float): The amount to add to the fee paid.

    """

    if amount > 0:

      self.fee\_paid += amount

      print(f"Fee updated for {self.name}. Total fee paid: {self.fee\_paid}")

    else:

      print("Invalid amount. Please enter a positive value for fee update.")

  def display\_details(self):

    """

    Displays the details of the student.

    """

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

    print(f"Name: {self.name}")

    print(f"Roll Number: {self.roll\_no}")

    print(f"Hostel Status: {'In Hostel' if self.hostel\_status else 'Not in Hostel'}")

    print(f"Fee Paid: {self.fee\_paid}")

    print("-----------------------")

# Example usage:

student1 = sru\_student("Alice", "SRU123", True)

student2 = sru\_student("Bob", "SRU456", False)

student1.display\_details()

student2.display\_details()

student1.fee\_update(5000.0)

student2.fee\_update(3000.0)

student1.display\_details()

student2.display\_details()

--- Student Details ---

Name: Alice

Roll Number: SRU123

Hostel Status: In Hostel

Fee Paid: 0.0

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

--- Student Details ---

Name: Bob

Roll Number: SRU456

Hostel Status: Not in Hostel

Fee Paid: 0.0

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

Fee updated for Alice. Total fee paid: 5000.0

Fee updated for Bob. Total fee paid: 3000.0

--- Student Details ---

Name: Alice

Roll Number: SRU123

Hostel Status: In Hostel

Fee Paid: 5000.0

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

--- Student Details ---

Name: Bob

Roll Number: SRU456

Hostel Status: Not in Hostel

Fee Paid: 3000.0

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

CODE AND OUTPUT BY CO-POILOT:

class SRUStudent:

        """

        Simple student representation with name, roll\_no, hostel\_status and fees\_due.

        Methods:

          - fee\_update(payment): apply a payment (reduces fees\_due, cannot go below 0)

          - display\_details(): return a string with student details

        """

        def \_\_init\_\_(self, name, roll\_no, hostel\_status=False, fees\_due=0.0):

            self.name = str(name)

            self.roll\_no = str(roll\_no)

            self.hostel\_status = bool(hostel\_status)

            self.fees\_due = float(fees\_due)

        def fee\_update(self, payment):

            """Apply a payment (positive number). Returns updated fees\_due."""

            if not isinstance(payment, (int, float)):

                raise TypeError("payment must be a number")

            # decrease fees\_due by payment, but never below 0

            self.fees\_due = max(0.0, self.fees\_due - float(payment))

            return self.fees\_due

        def display\_details(self):

            """Return formatted student details."""

            hs = "Yes" if self.hostel\_status else "No"

            return (

                f"Name: {self.name}\n"

                f"Roll No: {self.roll\_no}\n"

                f"Hostel: {hs}\n"

                f"Fees due: ₹{self.fees\_due:.2f}"

            )

    # Demo usage

    student = SRUStudent(name="Priya Kumar", roll\_no="SRU2025-045", hostel\_status=True, fees\_due=7500.0)

    print("\nBefore payment:")

    print(student.display\_details())

    # Student pays some fees

    student.fee\_update(2500)  # pay 2500

    print("\nAfter payment of ₹2500:")

    print(student.display\_details())

    # Another payment that would clear the dues

    student.fee\_update(6000)

    print("\nAfter additional payment (₹6000):")

    print(student.display\_details())

# ...existing code...

Before payment:

Name: Priya Kumar

Roll No: SRU2025-045

Hostel: Yes

Fees due: ₹7500.00

After payment of ₹2500:

Name: Priya Kumar

Roll No: SRU2025-045

Hostel: Yes

Fees due: ₹5000.00

After additional payment (₹6000):

Name: Priya Kumar

Roll No: SRU2025-045

Hostel: Yes

Fees due: ₹0.00

PROMPT: Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).

CODE AND OUTPUT BY GEMINI AI:

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."""

  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)}")

print(f"{num1} / 0 = {divide(num1, 0)}") # Example of division by zero

10 + 5 = 15

10 - 5 = 5

10 \* 5 = 50

10 / 5 = 2.0

10 / 0 = Error: Division by zero

CODE AND OUTPUT BY COPOILOT:

def add(a, b):

    """Return a + b."""

    return a + b

def subtract(a, b):

    """Return a - b."""

    return a - b

def multiply(a, b):

    """Return a \* b."""

    return a \* b

def divide(a, b):

    """Return a / b. Raises ZeroDivisionError on b == 0."""

    if b == 0:

        raise ZeroDivisionError("division by zero")

    return a / b

def calculator\_demo():

    """Simple demo showing the four operations."""

    a, b = 12, 4

    print("\nCalculator demo:")

    print(f"{a} + {b} = {add(a, b)}")

    print(f"{a} - {b} = {subtract(a, b)}")

    print(f"{a} \* {b} = {multiply(a, b)}")

    print(f"{a} / {b} = {divide(a, b)}")

# If running as script, call demo (keeps existing \_\_main\_\_ behavior intact).

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

    calculator\_demo()

Calculator demo:

12 + 4 = 16

12 - 4 = 8

12 \* 4 = 48

12 / 4 = 3.0