# AI ASSISTED CODING

# LAB ASSIGNMENT – 9.3

# NAME : N.KEERTHI REDDY

# HALL NO. :2403A52376

# BATCH : AI 14

# TASK -1

## .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., Gemini, Copilot, Cursor AI) to generate a docstring describing the function.

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

# CODE:

# 

# OUTPUT:

# 

# EXPLANATION :

# 

# TASK – 2

# 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.

# CODE :

# 

# OUTPUT :

# 

# EXPLANATION :

# This code defines a Python class named sru\_student. A class is a blueprint for creating objects (instances). In this case, each object created from this class will represent a student from SRU.

# Here's a breakdown of the code:

# class sru\_student:: This line defines the class named sru\_student.

# def \_init\_(self, name, roll\_no, hostel\_status):: This is the constructor method. It's automatically called when you create a new object of the sru\_student class.

# self: Refers to the instance of the class being created.

# name, roll\_no, hostel\_status: These are parameters that you pass when creating a new sru\_student object.

# Inside the method, self.name = name, self.roll\_no = roll\_no, and self.hostel\_status = hostel\_status assign the values passed during object creation to the attributes of the object.

# self.fee\_paid = 0: This initializes an attribute fee\_paid to 0 for every new student object.

# def fee\_update(self, amount):: This method is used to update the fee paid by a student.

# self: Refers to the instance of the class on which the method is called.

# amount: The amount of fee being paid.

# self.fee\_paid += amount: This line adds the amount to the current self.fee\_paid.

# print(f"Fee updated for {self.name}. Total fee paid: {self.fee\_paid}"): This line prints a confirmation message showing the student's name and their total fee paid.

# def display\_details(self):: This method is used to display the details of a student.

# self: Refers to the instance of the class on which the method is called.

# The print statements inside this method access the object's attributes (self.name, self.roll\_no, self.hostel\_status, self.fee\_paid) and print them in a formatted way.

# In essence, this class provides a structure to store and manage information about SRU students, allowing you to create individual student objects and perform actions like updating their fees and displaying their details.

# 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.

# CODE:

### def add(a, b):

### """

### Add two numbers.

### """

### return a + b

### def subtract(a, b):

### """

### Subtract two numbers.

### """

### return a - b

### def multiply(a, b):

### """

### Multiply o numbers.

### """

### return a \* b

### def divide(a, b):

### """

### Divide two numbers.

### Raises:

### ValueError: If the divisor is zero.

### """

### if b == 0:

### raise ValueError("Division by zero is not allowed.")

### return a / b

# OUTPUT :

A screenshot of a computer

AI-generated content may be incorrect.

# EXPLANATION :

### Module-level docstring: The first docstring, enclosed in triple quotes at the beginning of the cell, is a module-level docstring. It provides a brief overview of what the module (or script in this case) does and lists the functions it contains. def add(x, y):: This defines a function named add that takes two arguments, x and y. Docstring (NumPy Style): Below the function definition is its docstring. It explains the function's purpose, lists its Parameters with their types and descriptions, and describes the Returns value. def subtract(x, y):: This defines a function named subtract that takes two arguments, x and y, and returns their diAerence. It also has a NumPy style docstring explaining its purpose, parameters (with specific terms like "minuend" and "subtrahend"), and return value. def multiply(x, y):: This defines a function named multiply that takes two arguments, x and y, and returns their product. It includes a NumPy style docstring detailing its parameters and return value. def divide(x, y):: This defines a function named divide that takes two arguments, x and y, and returns their division. Docstring (NumPy Style): Its docstring explains the purpose, parameters, and return value. Raises section: This docstring also includes a Raises section to document the ZeroDivisionError that will occur if the divisor (y) is zero. Error Handling: The if y == 0: block explicitly checks for division by zero and raises a ZeroDivisionError with a descriptive message. Example Usage: The lines after the function definitions demonstrate how to call these functions with example numbers (num1 and num2) and print the results using f-strings for formatted output. Commented-out Example: The commented-out try...except block shows how you could handle the ZeroDivisionError when attempting to divide by zero. In essence, this code provides reusable functions for basic arithmetic with comprehensive documentation in the NumPy style, which is commonly used in scientific computing and data analysis libraries.