

LAB ASSIGNMENT-9

TASK-1:

Prompt:

Generate a Google-Style Docstring describing the python function to return sum of even and odd numbers in the given list.

Code and Output:

GitHub Copilot:

```
lab 9.py •
lab 9.py > sum_even_odd
Search (Ctrl+Shift+F) even_odd(numbers):
2
3     Returns the sum of even and odd numbers in a list.
4     This function takes a list of integers and calculates two separate sums:
5     one for all even numbers and one for all odd numbers.
6
7     Args:
8     |     numbers (list[int]): A list of integers.
9
10    Returns:
11    |     tuple[int, int]: A tuple containing two integers:
12    |         - The first element is the sum of even numbers.
13    |         - The second element is the sum of odd numbers.
14
15    Raises:
16    |     TypeError: If any element in the list is not an integer.
17
18    Examples:
19    |     >>> sum_even_odd([1, 2, 3, 4, 5])
20    |     (6, 9)
21
22    |     >>> sum_even_odd([])
23    |     (0, 0)
24    |     """
25    even_sum = 0
26    odd_sum = 0
27
28    for num in numbers:
29        if not isinstance(num, int):
30            raise TypeError("All elements in the list must be integers.")
31        if num % 2 == 0:
32            even_sum += num
33        else:
34            odd_sum += num
35
36    return even_sum, odd_sum
37 print(sum_even_odd([1, 2, 3, 4, 5])) # Output: (6, 9)
```

Gemini:

```
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}")
```

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

TASK-2:

Prompt:

Generate 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/step.

Code and Output:

GitHub Copilot:

```
lab 9.py X
lab 9.py > ...
1  # Define the sru_student class to model a student
2  class sru_student:
3      # Constructor method to initialize student attributes
4      def __init__(self, name, roll_no, hostel_status):
5          # Assign the student's name to the instance variable
6          self.name = name
7          # Assign the student's roll number to the instance variable
8          self.roll_no = roll_no
9          # Assign the student's hostel status (True/False) to the instance variable
10         self.hostel_status = hostel_status
11         # Initialize fee_paid attribute to False by default (fee not paid)
12         self.fee_paid = False
13
14     # Method to update the fee payment status of the student
15     def fee_update(self, status):
16         # Update the fee_paid attribute with the provided status (True/False)
17         self.fee_paid = status
18
19     # Method to display all details of the student
20     def display_details(self):
21         # Print the student's name
22         print(f"Name: {self.name}")
23         # Print the student's roll number
24         print(f"Roll No.: {self.roll_no}")
25         # Print the student's hostel status in a readable form
26         print(f"Hostel Status: {'Hosteller' if self.hostel_status else 'Non-Hosteller'}")
27         # Print whether the fee has been paid or not
28         print(f"Fee Paid: {'Yes' if self.fee_paid else 'No'}")
29
30
31     # Prompt the user to enter the student's name and store it in a variable
32     name = input("Enter student name: ")
33
34     # Loop to get a valid roll number from the user
35     while True:
36         try:
37             # Prompt the user for roll number and convert it to integer
```

```

37         # Prompt the user for roll number and convert it to integer
38         roll_no = int(input("Enter roll number: "))
39         # Exit the loop if conversion was successful
40         break
41     except ValueError:
42         # If input is not a valid integer, show error message and repeat
43         print("Invalid input! Please enter a numeric roll number.")
44
45     # Loop to get valid hostel status input ('y' or 'n') from the user
46     while True:
47         # Prompt the user if the student is a hosteller (yes/no)
48         hostel_input = input("Is the student a hosteller? (y/n): ").strip().lower()
49         # Check if the input is valid
50         if hostel_input in ('y', 'n'):
51             # Set hostel_status True if 'y', otherwise False
52             hostel_status = True if hostel_input == 'y' else False
53             # Exit the loop on valid input
54             break
55         else:
56             # Ask the user again if input was invalid
57             print("Please enter 'y' for yes or 'n' for no.")
58
59     # Loop to get valid fee payment status input ('y' or 'n') from the user
60     while True:
61         # Prompt the user if the fee has been paid
62         fee_input = input("Has the student paid the fee? (y/n): ").strip().lower()
63         # Check if the input is valid
64         if fee_input in ('y', 'n'):
65             # Set fee_paid True if 'y', otherwise False
66             fee_paid = True if fee_input == 'y' else False
67             # Exit the loop on valid input
68             break
69         else:
70             # Ask the user again if input was invalid
71             print("Please enter 'y' for yes or 'n' for no.")
72
73     # Create an sru student object with the user-provided name, roll number, and hostel status
74     student = sru_student(name, roll_no, hostel_status)
75
76     # Update the student's fee payment status using the fee_update method
77     student.fee_update(fee_paid)
78
79     # Print a header before displaying student details
80     print("\nStudent Details:")
81
82     # Call the method to display all details of the student
83     student.display_details()

```

```

PS C:\Users\mahit\OneDrive\Desktop\AIAC> & C:/Users/mahit/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/mahit/OneDrive/Desktop/AIAC/lab 9.py"
Enter student name: Vivek
Enter roll number: 97
Is the student a hosteller? (y/n): n
Has the student paid the fee? (y/n): y

Student Details:
Name: Vivek
Roll No.: 97
Hostel Status: Non-Hosteller
Name: Vivek
Roll No.: 97
Hostel Status: Non-Hosteller
Hostel Status: Non-Hosteller
Fee Paid: Yes

```

Gemini:

```
# Define a class named sru_student
class sru_student:
    # The __init__ method is the constructor for the class.
    # It is called when a new object of the class is created.
    def __init__(self, name, roll_no, hostel_status):
        # Initialize the name attribute with the provided name
        self.name = name
        # Initialize the roll_no attribute with the provided roll_no
        self.roll_no = roll_no
        # Initialize the hostel_status attribute with the provided hostel_status
        self.hostel_status = hostel_status
        # Initialize the fee_status attribute to False by default
        self.fee_status = False

    # Define a method to update the fee status
    def fee_update(self):
        # Set the fee_status attribute to True
        self.fee_status = True
        # Print a confirmation message
        print(f"Fee status for {self.name} (Roll No: {self.roll_no}) updated to Paid.")
```

```
# Define a method to display the student's details
def display_details(self):
    # Print the student's name
    print(f"Student Name: {self.name}")
    # Print the student's roll number
    print(f"Roll Number: {self.roll_no}")
    # Print the student's hostel status
    print(f"Hostel Status: {self.hostel_status}")
    # Print the student's fee status
    print(f"Fee Status: {'Paid' if self.fee_status else 'Pending'}")

# Example usage of the sru_student class:
# Create an instance of the sru_student class
student1 = sru_student("Alice", "SRU123", "Resident")
# Display the details of the student
student1.display_details()
# Update the fee status of the student
student1.fee_update()
# Display the details again to show the updated fee status
student1.display_details()
```

```
Student Name: Alice
Roll Number: SRU123
Hostel Status: Resident
Fee Status: Pending
Fee status for Alice (Roll No: SRU123) updated to Paid.
Student Name: Alice
Roll Number: SRU123
Hostel Status: Resident
Fee Status: Paid
Student Name: Bob
Roll Number: SRU456
Hostel Status: Day Scholar
Fee Status: Pending
```

TASK-3:

GitHub Copilot:

Prompt:

Generate a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide). Include module-level docstring + individual function docstrings.

Code and Output:

```
lab 9.py • multiply
1  """
2  calculator.py
3
4  A simple calculator module that provides basic arithmetic operations:
5  addition, subtraction, multiplication, and division.
6
7  This script is intended to demonstrate modular programming with
8  function definitions and proper docstring documentation.
9
10 Author: Your Name
11 Date: YYYY-MM-DD
12 """
13 def add(a, b):
14     """
15     Add two numbers.
16
17     Args:
18         a (float): The first number.
19         b (float): The second number.
20
21     Returns:
22         float: The sum of a and b.
23     Example:
24         >>> add(2, 3)
25         5
26     """
27     return a + b
28
29 def subtract(a, b):
30     """
31     Subtract one number from another.
32
33     Args:
34         a (float): The number to subtract from.
35         b (float): The number to subtract.
36
37     Returns:
38         float: The result of a - b.
39
40     Example:
41         >>> subtract(5, 2)
42         3
43     """
44     return a - b
45
46 def multiply(a, b):
47     """
48     Multiply two numbers.
```



```

49
50     Args:
51         a (float): The first number.
52         b (float): The second number.
53
54     Returns:
55         float: The product of a and b.
56
57     Example:
58         >>> multiply(4, 5)
59         20
60     """
61     return a * b
62
63
64 def divide(a, b):
65     """
66     Divide one number by another.
67
68     Args:
69         a (float): The numerator.
70         b (float): The denominator.
71
72     Returns:
73         float: The result of a divided by b.
74
75     Raises:
76         ZeroDivisionError: If b is zero.
77
78     Example:
79         >>> divide(10, 2)
80         5.0
81     """
82     if b == 0:
83         raise ZeroDivisionError("Cannot divide by zero.")
84     return a / b
85
86
87 # Optional: If you want to test the functions directly in this script
88 if __name__ == "__main__":
89     # Sample usage
90     print("Add: 10 + 5 =", add(10, 5))
91     print("Subtract: 10 - 5 =", subtract(10, 5))
92     print("Multiply: 10 * 5 =", multiply(10, 5))
93     print("Divide: 10 / 5 =", divide(10, 5))

```

```

PS C:\Users\mahit\OneDrive\Desktop\AIAC> & C:\Users\mahit\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/mahit/OneDrive/Desktop/AIAC/lab 9.py"
Add: 10 + 5 = 15
Subtract: 10 - 5 = 5
Multiply: 10 * 5 = 50
Divide: 10 / 5 = 2.0

```

Gemini:

Prompt:

Generate a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide) . Write docstring in code with NumPy Style

Code and Output:

```
def add(a, b):  
    """  
    Adds two numbers.  
  
    Parameters  
    -----  
    a : int or float  
        The first number.  
    b : int or float  
        The second number.  
  
    Returns  
    -----  
    int or float  
        The sum of a and b.  
  
    Examples  
    -----  
    >>> add(2, 3)  
    5  
    >>> add(1.5, 2.5)  
    4.0
```

```
    """  
    return a + b  
  
def subtract(a, b):  
    """  
    Subtracts the second number from the first.  
  
    Parameters  
    -----  
    a : int or float  
        The first number.  
    b : int or float  
        The second number.  
  
    Returns  
    -----  
    int or float  
        The difference between a and b.  
  
    Examples  
    -----  
    >>> subtract(5, 2)  
    3
```

```
▶ >>> subtract(4.0, 1.5)
2.5
"""
return a - b

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

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

    Returns
    -----
    int or float
        The product of a and b.

    Examples
```

```
▶ -----
>>> multiply(2, 4)
8
>>> multiply(2.5, 2)
5.0
"""
return a * b

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

    Parameters
    -----
    a : int or float
        The numerator.
    b : int or float
        The denominator.

    Returns
    -----
    int or float
        The result of the division.
```

```

▶ Raises
-----
ZeroDivisionError
    If the denominator is zero.

Examples
-----
>>> divide(10, 2)
5.0
>>> divide(5, 0)
Traceback (most recent call last):
...
ZeroDivisionError: division by zero
"""
if b == 0:
    raise ZeroDivisionError("division by zero")
return a / b

# Example usage:
print(f"Addition: {add(5, 3)}")
print(f"Subtraction: {subtract(10, 4)}")
print(f"Multiplication: {multiply(6, 7)}")

```

```

⇨ Addition: 8
Subtraction: 6
Multiplication: 42
Division: 4.0

```

Comparison: Google-style vs NumPy-style

Readability:

- *Google-style* is cleaner and easier for beginners to read.
- *NumPy-style* is more structured, preferred in scientific computing.

Parameters:

- *Google-style*: Uses Args: with simple name-type-description format.
- *NumPy-style*: Uses a Parameters section with aligned formatting.

Return values:

- *Google-style*: Returns: with brief description.
- *NumPy-style*: Returns block includes type and detailed explanation.

Exceptions:

- Both styles support Raises: or Raises section, but NumPy-style is more detailed.

Examples:

- *Google-style*: Examples: section uses inline >>> syntax.
- *NumPy-style*: Examples section is more formal, under its own heading.

Usage:

- *Google-style*: Common in web apps, general Python projects.
- *NumPy-style*: Standard in scientific libraries (NumPy, Pandas, etc.).

Tooling:

- Both are compatible with Sphinx; *NumPy-style* works best with napoleon extension for auto-doc generation.

