

Assignment-9.3

Task 1:

Write a python function to implement the sum of even and odd numbers to generate a docstring describing the function.

Code and Output:

(gemini)

give the without comment lines

```
[4] ✓ De
def sum_even_odd(numbers):
    sum_even = 0
    sum_odd = 0
    for number in numbers:
        if number % 2 == 0:
            sum_even += number
        else:
            sum_odd += number
    return sum_even, sum_odd

# 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
```

give the with in comments

```
1.1
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.
        (sum_even, sum_odd)
    """
    sum_even = 0
    sum_odd = 0
    for number in numbers:
        if number % 2 == 0:
            sum_even += number
        else:
            sum_odd += number
    return sum_even, sum_odd

# 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
```

Explanation:

K Explain the code

- ✦ The code defines a Python function called `sum_even_odd` that takes a list of numbers as input. It initializes two variables, `sum_even` and `sum_odd`, to zero. Then, it iterates through each number in the input list. If a number is even (divisible by 2 with no remainder), it's added to `sum_even`; otherwise (if it's odd), it's added to `sum_odd`. Finally, the function returns a tuple containing the calculated `sum_even` and `sum_odd`. The example usage demonstrates how to call the function with a sample list and print the results.



(Github copilot):

```
def sum_even_odd(numbers):  
    """  
    Calculates the sum of even and odd numbers in a list.  
    Args:  
        numbers (list of int): A list of integers.  
    Returns:  
        tuple: A tuple containing two values:  
            - The sum of even numbers.  
            - The sum of odd numbers.  
    Example:  
    >>> sum_even_odd([1, 2, 3, 4, 5])  
    (6, 9)  
    """  
    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  
  
# Example usage:  
nums = [1, 2, 3, 4, 5]  
even, odd = sum_even_odd(nums)  
print("Sum of even numbers:", even)  
print("Sum of odd numbers:", odd)
```

PS C:\Users\Vinod\OneDrive\Desktop\AI> & C:\Users\Vinod\AppData\Local\Programs\Python\Python113\python.exe "c:\Users\Vinod\OneDrive\Desktop\AI\AI(4).py"
Sum of even numbers: 6
Sum of odd numbers: 9
PS C:\Users\Vinod\OneDrive\Desktop\AI>

```
AI(4).py > sum_even_odd
1 def sum_even_odd(numbers):
2     even_sum = 0
3     odd_sum = 0
4     for num in numbers:
5         if num % 2 == 0:
6             even_sum += num
7         else:
8             odd_sum += num
9     return even_sum, odd_sum
10
11 nums = [1, 2, 3, 4, 5]
12 even, odd = sum_even_odd(nums)
13 print("Sum of even numbers:", even)
14 print("Sum of odd numbers:", odd)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Vinod\OneDrive\Desktop\AI> & C:\Users\Vinod\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\Vinod\OneDrive/D
esktop/AI/AI(4).py"
Sum of even numbers: 6
Sum of odd numbers: 9
PS C:\Users\Vinod\OneDrive\Desktop\AI>
```

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.

Code and Output:

Gemini:

```
[7]
✓ On # Define a class named 'sru_student'
class sru_student:
    # Initialize the class with name, roll number, and hostel status
    def __init__(self, name, roll_no, hostel_status):
        # Assign the provided name to the instance's name attribute
        self.name = name
        # Assign the provided roll number to the instance's roll_no attribute
        self.roll_no = roll_no
        # Assign the provided hostel status to the instance's hostel_status attribute
        self.hostel_status = hostel_status
        # Initialize the total_fee attribute to 0
        self._total_fee = 0

    # Define a method to update the student's fee
    def fee_update(self, amount):
        # Add the provided amount to the current total_fee
        self._total_fee += amount
        # Print a confirmation message with the updated fee
        print(f"Fee updated for {self.name}. Total fee: {self._total_fee}")

    # Define a method to display the student's details
    def display_details(self):
        # Print the student's name
        print(f"Name: {self.name}")
        # Print the student's roll number
        print(f"Roll No.: {self.roll_no}")
        # Print the student's hostel status
        print(f"Hostel Status: {self.hostel_status}")
        # Print the student's current total fee
        print(f"Total Fee: {self._total_fee}")
```

```

        # Print the student's current total fee
        print(f"Total Fee: {self._total_fee}")

# Example usage (optional, you can remove this if you only need the class definition)
# Create an instance of the sru_student class
student1 = sru_student("Alice", "12345", "Resident")
# Update the fee for student1
student1.fee_update(5000)
# Display the details of student1
student1.display_details()

```

```

➡ Fee updated for Alice. Total fee: 5000
Name: Alice
Roll No.: 12345
Hostel Status: Resident
Total Fee: 5000

```

```

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._total_fee = 0

    def fee_update(self, amount):
        self._total_fee += amount
        print(f"Fee updated for {self.name}. Total fee: {self._total_fee}")

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll No.: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Total Fee: {self._total_fee}")

# Example usage (optional, you can remove this if you only need the class definition)
student1 = sru_student("Alice", "12345", "Resident")
student1.fee_update(5000)
student1.display_details()

```

```

➡ Fee updated for Alice. Total fee: 5000
Name: Alice
Roll No.: 12345
Hostel Status: Resident
Total Fee: 5000

```

🔍 explanation

➡ Certainly! Both of the visible cells contain the same Python code for a class called `sru_student`. Here's a breakdown of what the code does:

- `class sru_student`: This line defines a new class named `sru_student`. Think of a class as a blueprint for creating objects (in this case, student objects).
- `def __init__(self, name, roll_no, hostel_status)`: This is the constructor method. It's called when you create a new `sru_student` object.
 - `self`: Refers to the instance of the class being created.
 - `name`, `roll_no`, `hostel_status`: These are parameters that you pass when creating a student object.
 - Inside the method, `self.name = name`, `self.roll_no = roll_no`, and `self.hostel_status = hostel_status` assign the values you provide to attributes of the student object.
 - `self._total_fee = 0`: This initializes an attribute `_total_fee` to 0 for each new student object. The underscore at the beginning is a convention to indicate that this attribute is intended for internal use within the class.
- `def fee_update(self, amount)`: This is a method to update the student's fee.
 - `self`: Refers to the instance of the class.
 - `amount`: The amount to add to the fee.
 - `self._total_fee += amount`: This line adds the `amount` to the current `_total_fee` of the student object.
 - `print(f"Fee updated for {self.name}. Total fee: {self._total_fee}")`: This line prints a confirmation message showing the student's name and their updated total fee.
- `def display_details(self)`: This method is used to display the details of a student.
 - `self`: Refers to the instance of the class.
 - The `print` statements inside this method display the student's name, roll number, hostel status, and total fee.
- **Example usage (commented out in one cell, uncommented in the other):**
 - `student1 = sru_student("Alice", "12345", "Resident")`: This line creates a new `sru_student` object named `student1` with the provided details.
 - `student1.fee_update(5000)`: This calls the `fee_update` method on the `student1` object, adding 5000 to their total fee.
 - `student1.display_details()`: This calls the `display_details` method on the `student1` object to print their information.

In summary, this code defines a simple class to represent a student, allowing you to store their information, update their fees, and display their details.

Github:

```

1 class sru_student:
2     # Initialize the student object with name, roll number, and hostel status
3     def __init__(self, name, roll_no, hostel_status):
4         self.name = name # Store the student's name
5         self.roll_no = roll_no # Store the student's roll number
6         self.hostel_status = hostel_status # Store hostel status (Yes/No)
7         self.fee_paid = False # Track if the fee is paid, default is False
8
9     # Method to update the fee status
10    def fee_update(self, status):
11        self.fee_paid = status # Update the fee_paid attribute
12
13    # Method to display student details
14    def display_details(self):
15        print("Name:", self.name) # Print the student's name
16        print("Roll No.:", self.roll_no) # Print the student's roll number
17        print("Hostel Status:", self.hostel_status) # Print hostel status
18        print("Fee Paid:", "Yes" if self.fee_paid else "No") # Print fee status
19
20    # Example usage:
21    student1 = sru_student("Rahul", "22CS101", "Yes") # Create a student object
22    student1.fee_update(True) # Update fee status to paid
23    student1.display_details() # Display all details
24
25    student2 = sru_student("Priya", "22CS102", "No") # Create another student object
26    student2.fee_update(False) # Fee not paid
27    student2.display_details() # Display all details
28

```

Name: Rahul

Roll No.: 22CS101

Hostel Status: Yes

Fee Paid: Yes

Name: Priya

Roll No.: 22CS102

Hostel Status: No

Fee Paid: No

PS C:\Users\Vinod\OneDrive\Desktop\AI>


```
        print(f"{num1} / {num2} = {result}")
    except ZeroDivisionError as e:
        print(f"Error: {e}")
    else:
        print("Invalid operation. Please choose add, subtract, multiply, or divide.")

calculate()

Enter operation (add, subtract, multiply, divide): divide
Enter first number: 12
Enter second number: 4
12.0 / 4.0 = 3.0
```

give the code with inline comments:

```
def calculate():
    # Get the operation choice from the user and convert to lowercase
    operation = input("Enter operation (add, subtract, multiply, divide): ").lower()
    try:
        # Get the two numbers from the user and convert them to floats
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))
    except ValueError:
        # Handle the case where the input is not a valid number
        print("Invalid input. Please enter numbers.")
        return # Exit the function if input is invalid

    # Perform the chosen operation and print the result
    if operation == 'add':
        result = add(num1, num2)
        print(f"{num1} + {num2} = {result}")
    elif operation == 'subtract':
        result = subtract(num1, num2)
        print(f"{num1} - {num2} = {result}")
    elif operation == 'multiply':
        result = multiply(num1, num2)
        print(f"{num1} * {num2} = {result}")
    elif operation == 'divide':
        try:
            # Handle potential ZeroDivisionError during division
            result = divide(num1, num2)
```

```

    print(f"{num1} * {num2} = {result}")
elif operation == 'divide':
    try:
        # Handle potential ZeroDivisionError during division
        result = divide(num1, num2)
        print(f"{num1} / {num2} = {result}")
    except ZeroDivisionError as e:
        print(f"Error: {e}")
    else:
        # Handle the case where the operation is not recognized
        print("Invalid operation. Please choose add, subtract, multiply, or divide.")

# Call the calculate function to start the calculator
calculate()


```

```

Enter operation (add, subtract, multiply, divide): add
Enter first number: 22
Enter second number: 25
22.0 + 25.0 = 47.0

```

Explanation:

 give the explanation

 The code defines a function called `calculate` that serves as a simple calculator interface.

- Get Operation:** It first prompts the user to enter the desired operation (add, subtract, multiply, or divide) and converts the input to lowercase for case-insensitive matching.
- Get Numbers:** It then prompts the user to enter two numbers. A `try-except` block is used to handle potential `ValueError` if the user enters non-numeric input.
- Perform Calculation:** Based on the entered operation, it calls the corresponding function (`add`, `subtract`, `multiply`, or `divide`) with the provided numbers.
- Handle Division by Zero:** For the 'divide' operation, another `try-except` block specifically catches `ZeroDivisionError` if the second number is zero, printing an error message.
- Print Result:** If the operation is valid and no errors occur, the result of the calculation is printed in a user-friendly format.
- Invalid Operation:** If the entered operation is not one of the four supported options, an "Invalid operation" message is displayed.
- Call Function:** Finally, the `calculate()` function is called to start the calculator program.

Github:

give the code with inline comments

```

calculator.py
"""
A simple calculator module that provides basic arithmetic operations: addition, subtraction, multiplication, and division.
"""
from typing import Union

def add(a: Union[int, float], b: Union[int, float]) -> Union[int, float]:
    """Returns the sum of two numbers."""
    return a + b

def subtract(a: Union[int, float], b: Union[int, float]) -> Union[int, float]:
    """Returns the difference between two numbers."""
    return a - b

def multiply(a: Union[int, float], b: Union[int, float]) -> Union[int, float]:
    """Returns the product of two numbers."""
    return a * b

def divide(a: Union[int, float], b: Union[int, float]) -> Union[int, float]:
    """Returns the quotient of two numbers. Raises ZeroDivisionError if b is zero."""
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero")
    return a / b

def calculate():
    """Main function to start the calculator program. It prompts the user for an operation and two numbers, then performs the calculation and displays the result. It uses a try-except block to handle potential errors like non-numeric input or division by zero. If the operation is invalid, it displays an error message. Finally, it calls the calculate function to start the calculator program."""
    operation = input("Enter operation (add, subtract, multiply, divide): ").lower()
    num1 = input("Enter first number: ")
    num2 = input("Enter second number: ")

    try:
        result = None
        if operation == 'add':
            result = add(float(num1), float(num2))
        elif operation == 'subtract':
            result = subtract(float(num1), float(num2))
        elif operation == 'multiply':
            result = multiply(float(num1), float(num2))
        elif operation == 'divide':
            result = divide(float(num1), float(num2))
        else:
            print("Invalid operation. Please choose add, subtract, multiply, or divide.")
    except ValueError as e:
        print(f"Error: {e}")
    except ZeroDivisionError as e:
        print(f"Error: {e}")

    if result is not None:
        print(f"{num1} {operation} {num2} = {result}")

    # Call the calculate function to start the calculator
    calculate()

```



```

33     """
34     float or int
35     | | The sum of a and b.
36
37     Examples
38     """
39     >>> add(2, 3)
40     5
41     """
42     return a + b # Return the sum
43
44 def subtract(a, b):
45     """
46     Subtract one number from another.
47
48     Parameters
49     """
50     a : float or int
51     | | The number to subtract from.
52     b : float or int
53     | | The number to subtract.
54
55     Returns
56     """
57     float or int
58     | | The difference of a and b.
59
60     Examples
61     """
62     >>> subtract(5, 2)
63     3
64     """

```

```

◆ All4py > ...
44 def subtract(a, b):
45     """
46     return a - b # Return the difference.
47
48     Parameters
49     """
50     a : float or int
51     | | The first number.
52     b : float or int
53     | | The second number.
54
55     Returns
56     """
57     float or int
58     | | The product of a and b.
59
60     Examples
61     """
62     >>> multiply(2, 4)
63     8
64     """
65     return a * b # Return the product
66
67 def divide(a, b):
68     """
69     Divide one number by another.
70
71     Parameters
72     """

```

```
AI(4).py > --
10: def divide(a, b):
11:     """
12:     Parameters
13:     -----
14:     a : float or int
15:         The numerator.
16:     b : float or int
17:         The denominator.
18:
19:     Returns
20:     -----
21:     float
22:         The quotient of a and b.
23:
24:     Raises
25:     -----
26:     ValueError
27:         If b is zero.
28:
29:     Examples
30:     -----
31:     >>> divide(10, 2)
32:     5.0
33:     """
34:     if b == 0:
35:         raise ValueError("Cannot divide by zero.") # Raise error if denominator is zero
36:     return a / b # Return the quotient
37:
38: # Example usage
39: if __name__ == "__main__":
40:     x, y = 10, 5 # Example numbers
41:     print("Add:", add(x, y)) # Output: 15
42:     print("Subtract:", subtract(x, y)) # Output: 5
43:     print("Multiply:", multiply(x, y)) # Output: 50
44:     print("Divide:", divide(x, y)) # Output: 2.0
45:
46: PS C:\Users\Vinod\OneDrive\Desktop\AI> & C:\Users\Vinod\AppData\Local\Programs\Python\Python113\python.exe "C:\Users\Vinod\OneDrive\Desktop\AI\AI(4).py"
Add: 15
Subtract: 5
Multiply: 50
Divide: 2.0
PS C:\Users\Vinod\OneDrive\Desktop\AI>
```

give the code without comment lines:

```
AI(4).py > --
21: def add(a, b):
22:     """
23:     Add two numbers.
24:
25:     Parameters
26:     -----
27:     a : float or int
28:         The first number.
29:     b : float or int
30:         The second number.
31:
32:     Returns
33:     -----
34:     float or int
35:         The sum of a and b.
36:
37:     Examples
38:     -----
39:     >>> add(2, 3)
40:     5
41:     """
42:     return a + b
43:
44: def subtract(a, b):
45:     """
46:     Subtract one number from another.
47:
48:     Parameters
49:     -----
50:     a : float or int
51:         The first number.
52:     b : float or int
53:         The second number.
54:
55:     Returns
56:     -----
57:     float or int
58:         The difference of a and b.
59:
60:     Examples
61:     -----
62:     >>> subtract(5, 3)
63:     2
64:     """
65:     return a - b
66:
67: # Example usage
68: if __name__ == "__main__":
69:     x, y = 10, 5 # Example numbers
70:     print("Add:", add(x, y)) # Output: 15
71:     print("Subtract:", subtract(x, y)) # Output: 5
72:     print("Multiply:", multiply(x, y)) # Output: 50
73:     print("Divide:", divide(x, y)) # Output: 2.0
74:
75: PS C:\Users\Vinod\OneDrive\Desktop\AI>
```

```

46     Subtract one number from another.
47
48     Parameters
49     -----
50     a : float or int
51         The number to subtract from.
52     b : float or int
53         The number to subtract.
54
55     Returns
56     -----
57     float or int
58         The difference of a and b.
59
60     Examples
61     -----
62     >>> subtract(5, 2)
63     3
64     ***
65     return a - b
66
67 def multiply(a, b):
68     """
69     Multiply two numbers.
70
71     Parameters
72     -----
73     a : float or int
74         The first number.
75     b : float or int
76         The second number.
77
78     Returns
79     -----
80     float or int
81         The product of a and b.

```

```

82
83     Examples
84     -----
85     >>> multiply(2, 4)
86     8
87     ***
88     return a * b
89
90 def divide(a, b):
91     """
92     Divide one number by another.
93
94     Parameters
95     -----
96     a : float or int
97         The numerator.
98     b : float or int
99         The denominator.
100
101     Returns
102     -----
103     float
104         The quotient of a and b.
105
106     Raises
107     -----
108     ValueError
109         If b is zero.
110
111     Examples
112     -----
113     >>> divide(10, 2)
114     5.0
115     ***

```

```
116     if b == 0:
117         raise ValueError("Cannot divide by zero.")
118     return a / b
119
120 if __name__ == "__main__":
121     x, y = 10, 5
122     print("Add:", add(x, y))
123     print("Subtract:", subtract(x, y))
124     print("Multiply:", multiply(x,y))
125     print("Divide:", divide(x, y))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell + - [] [] ... [] [] [] []

PS C:\Users\Vinod\OneDrive\Desktop\AI> & C:\Users\Vinod\AppData\Local\Programs\Python\Python313\python.exe
"c:/Users/Vinod/OneDrive/Desktop/AI/AI(4).py"

Add: 15

Subtract: 5

Multiply: 50

Divide: 2.0

PS C:\Users\Vinod\OneDrive\Desktop\AI> |