AI Assisted Coding

Lab Assignment 9.3

Name: K.Akshitha

Hall Ticket no: 2303A51330

Batch No: 20

Task -1:

Prompt: Generate a Google-style Python docstring for a function named sum_even_odd (numbers) that takes a list of integers as input and returns a tuple containing the sum of even numbers and the sum of odd numbers. Include description, arguments, return values, and example usage.

Assignment-9.3.py

Assignment-9.3.py > sru_student > __init__

```python
#Task-1:
def sum_even_odd(numbers):
    """Calculates the sum of even and odd numbers in a list.
    Args:
        numbers (list): A list of integers to process.
    Returns:
        tuple: A tuple containing (sum_of_evens, sum_of_odds).
    Example:
        >>> sum_even_odd([1, 2, 3, 4, 5, 6])
        (12, 9)"""

    sum_even = sum(num for num in numbers if num % 2 == 0)
    sum_odd = sum(num for num in numbers if num % 2 != 0)
    return sum_even, sum_odd
# Test the function
if __name__ == "__main__":
    test_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    even_sum, odd_sum = sum_even_odd(test_list)
    print(f"Sum of even numbers: {even_sum}")
    print(f"Sum of odd numbers: {odd_sum}")
```

OUTPUT:

```
python -u "/Users/akshithakashireddy/Desktop/AI /tempCodeRunnerFile.py"
(base) akshithakashireddy@Akshithas AI  % python -u "/Users/akshithakashireddy/Desktop/AI /tempCodeRunnerFile.py"
Sum of even numbers: 30
Sum of odd numbers: 25
(base) akshithakashireddy@Akshithas AI  %
```

Justification:

The prompt clearly specifies the function name, input type, output format, and required Google-style documentation. This helps the AI generate accurate, structured, and complete docstrings. A well-defined prompt reduces ambiguity and improves the quality of AI-generated documentation

Task 2:

Prompt: Add clear and meaningful inline comments to the following Python class sru_student explaining each attribute, constructor, and method functionality. Ensure comments improve readability for beginner developers.

Assignment-9.3.py

```python
# Task-2:
# Define a class to manage SRU student details
class sru_student:
    """A class to manage student information and fees."""
    # Constructor to initialize student details
    def __init__(self, name, roll_no, hostel_status):
        # Assign student name
        self.name = name
        # Assign roll number
        self.roll_no = roll_no
        # Store hostel status (Yes/No)
        self.hostel_status = hostel_status
        # Initialize fee amount to zero
        self.fee = 0
    # Method to update student fee
    def fee_update(self, amount):
        # Add given amount to existing fee
        self.fee += amount
    # Method to display student details
    def display_details(self):
        # Print student name
        print(f"Name: {self.name}")
        # Print roll number
        print(f"Roll Number: {self.roll_no}")
        # Print hostel status
        print(f"Hostel Status: {self.hostel_status}")
        # Print total fee
        print(f"Fee: {self.fee}")
# Test the class
if __name__ == "__main__":
    # Create student object
    student = sru_student("Alice", 101, "Yes")
    # Update fee amount
    student.fee_update(5000)
    # Display student details
    student.display_details()

```

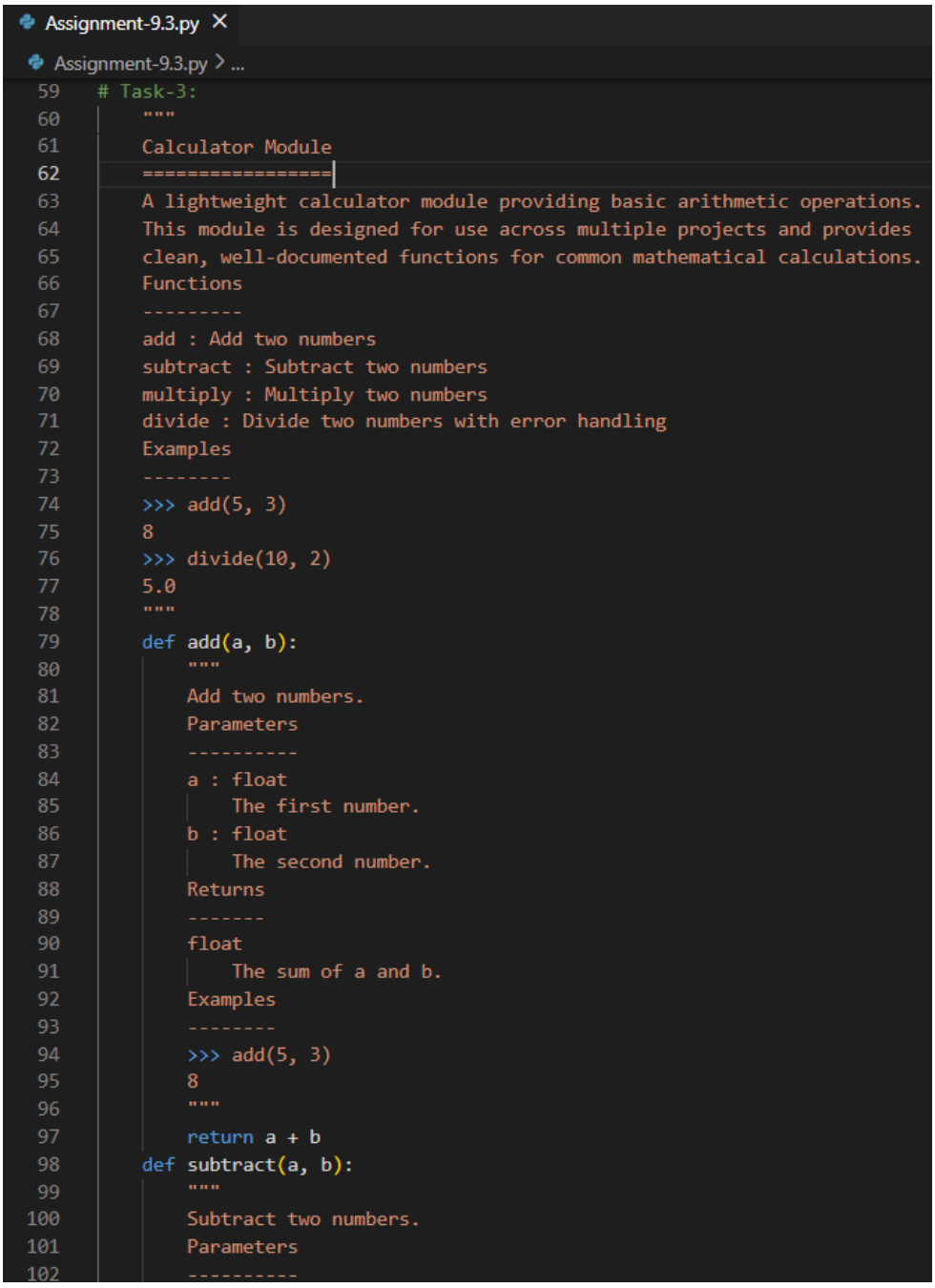
Output:

```
(base) akshithakashireddy@Akshithas AI  % python -u "/Users/akshithakashireddy/Desktop/AI /tempCodeRunnerFile.py"
Name: Alice
Roll Number: 101
Hostel Status: Yes
Fee: 5000
(base) akshithakashireddy@Akshithas AI  %
```

Justification: The prompt clearly asks the AI to generate inline comments explaining the class structure and methods. Specifying readability and beginner understanding helps produce simple and meaningful comments. This ensures the generated comments improve code clarity and maintainability.

Task-3:

Prompt: Generate a module-level docstring and NumPy-style function-level docstrings for a Python calculator module containing add, subtract, multiply, and divide functions. Include description, parameters, returns, and usage details.



```python
# Task-3:
    """
    Calculator Module
    ================
    A lightweight calculator module providing basic arithmetic operations.
    This module is designed for use across multiple projects and provides
    clean, well-documented functions for common mathematical calculations.
    Functions
    ---------
    add : Add two numbers
    subtract : Subtract two numbers
    multiply : Multiply two numbers
    divide : Divide two numbers with error handling
    Examples
    --------
    >>> add(5, 3)
    8
    >>> divide(10, 2)
    5.0
    """
    def add(a, b):
        """
        Add two numbers.
        Parameters
        ----------
        a : float
            The first number.
        b : float
            The second number.
        Returns
        -------
        float
            The sum of a and b.
        Examples
        --------
        >>> add(5, 3)
        8
        """
        return a + b
    def subtract(a, b):
        """
        Subtract two numbers.
        Parameters
        ----------
```

Assignment-9.3.py

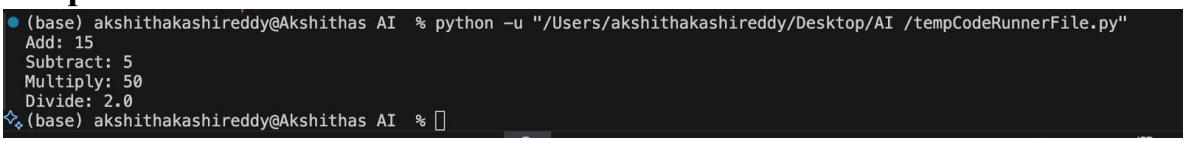
```
def subtract(a, b):
    a : float
        The minuend.
    b : float
        The subtrahend.
    Returns
    -------
    float
        The difference of a and b.
    Examples
    --------
    >>> subtract(10, 4)
    6
    """
    return a - b
def multiply(a, b):
    """
    Multiply two numbers.
    Parameters
    ----------
    a : float
        The first factor.
    b : float
        The second factor.
    Returns
    -------
    float
        The product of a and b.
    Examples
    --------
    >>> multiply(4, 5)
    20
    """
    return a * b
def divide(a, b):
    """
    Divide two numbers.
    Parameters
    ----------
    a : float
        The dividend.
    b : float
        The divisor.
    Returns
    -------
```

Assignment-9.3.py

```
def divide(a, b):
    float
        The quotient of a and b.
    Raises
    ------
    ValueError
        If b is zero.

    Examples
    --------
    >>> divide(10, 2)
    5.0
    """
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
# Test the functions
if __name__ == "__main__":
    print(f"Add: {add(10, 5)}")
    print(f"Subtract: {subtract(10, 5)}")
    print(f"Multiply: {multiply(10, 5)}")
    print(f"Divide: {divide(10, 5)}")
```

Output:



Justification:

The prompt clearly specifies the documentation style, module purpose, and required functions, helping the AI generate structured and accurate documentation. Mentioning NumPy style ensures standardized formatting. This improves readability and consistency across multiple functions.