

AI ASSIGNMENT 9.3

NAME : Varshitha.K

ROLLNO: 2403A52397.

Batch : AI 14

Task Description#1 Basic Docstring Generation

- 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.

```
def sum_even_odd(numbers):  
    """  
    Calculate 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 integers:  
            - sum_even (int): Sum of all even numbers in the list.  
            - sum_odd (int): Sum of all odd numbers in the list.  
  
    Example:  
        >>> sum_even_odd([1, 2, 3, 4, 5])  
        (6, 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  
  
# Example usage  
numbers_list = [1, 2, 3, 4, 5]  
result = sum_even_odd(numbers_list)  
print("Sum of even numbers:", result[0])  
print("Sum of odd numbers:", result[1])
```

Sum of even numbers: 6
Sum of odd numbers: 9

```

def sum_even_odd(numbers):
    """Calculates the sum of even and odd numbers in a list.

    Args:
        numbers (list): A list of integers.

    Returns:
        tuple: A tuple containing two integers:
            - sum_even (int): The sum of all even numbers in the list.
            - sum_odd (int): The sum of all odd numbers in the list.

    Examples:
        >>> sum_even_odd([1, 2, 3, 4, 5])
        (6, 9)
        >>> sum_even_odd([])
        (0, 0)
        >>> sum_even_odd([-2, -1, 0, 1, 2])
        (0, -2)
    """
    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

# Example usage
numbers_list = [1, 2, 3, 4, 5]
result = sum_even_odd(numbers_list)
print("Sum of even numbers:", result[0])
print("Sum of odd numbers:", result[1])

```

Sum of even numbers: 6
 Sum of odd numbers: 9

Okay, I can explain the previous code, which is the `sum_even_odd` function in cell `N9GpnC_t_-2i` and its usage in cell `PdJBSjcaEVFs`.

Cell `N9GpnC_t_-2i`:

This cell defines a Python function called `sum_even_odd` that takes a list of numbers as input and calculates the sum of the even numbers and the sum of the odd numbers in that list.

- `def sum_even_odd(numbers):`: This line defines the function named `sum_even_odd` that accepts one argument, `numbers`, which is expected to be a list.
- `"""Docstring explaining the function..."""`: This is a docstring that explains what the function does, its arguments, what it returns, and provides examples of how to use it.
- `sum_even = sum(num for num in numbers if num % 2 == 0)`: This line calculates the sum of the even numbers. It uses a generator expression `(num for num in numbers if num % 2 == 0)` to iterate through the `numbers` list and select only the numbers where the remainder when divided by 2 is 0 (i.e., even numbers). The `sum()` function then calculates the sum of these selected even numbers.
- `sum_odd = sum(num for num in numbers if num % 2 != 0)`: This line calculates the sum of the odd numbers. Similar to the previous line, it uses a generator expression `(num for num in numbers if num % 2 != 0)` to iterate through the list and select only the numbers where the remainder when divided by 2 is not 0 (i.e., odd numbers). The `sum()` function then calculates the sum of these selected odd numbers.
- `return sum_even, sum_odd`: This line returns the calculated `sum_even` and `sum_odd` as a tuple.


Cell `PdJBSjcaEVFs`:

This cell demonstrates how to use the `sum_even_odd` function.

- `numbers_list = [1, 2, 3, 4, 5]`: This line creates a list of numbers named `numbers_list`.
- `result = sum_even_odd(numbers_list)`: This line calls the `sum_even_odd` function with `numbers_list` as the argument and stores the returned tuple (the sum of even and odd numbers) in the `result` variable.
- `print("Sum of even numbers:", result[0])`: This line prints the label "Sum of even numbers:" followed by the first element of the `result` tuple, which is the sum of the even numbers.
- `print("Sum of odd numbers:", result[1])`: This line prints the label "Sum of odd numbers:" followed by the second element of the `result` tuple, which is the sum of the odd numbers.

Task Description#2 Automatic Inline Comments


- 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.

```
 class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status

    def fee_update(self, status):
        self.hostel_status = status

    def display_details(self):
        print("Name:", self.name)
        print("Roll No.:", self.roll_no)
        print("Hostel Status:", self.hostel_status)

student1 = sru_student("Varshitha", "2403A52397", "No hostel")
student1.fee_update("No hostel")
student1.display_details()
```

 Name: Varshitha
Roll No.: 2403A52397
Hostel Status: No hostel

This code defines a Python class called `sru_student` to represent a student with their details.

- `class sru_student:`: This line defines a new class named `sru_student`.
- `__init__(self, name, roll_no, hostel_status):`: This is the constructor method. It's called when you create a new `sru_student` object. It takes the student's `name`, `roll_no`, and `hostel_status` as arguments and stores them as attributes of the object using `self.name`, `self.roll_no`, and `self.hostel_status`.
- `fee_update(self, status):`: This method is used to update the `hostel_status` of a student. It takes a `status` argument and updates the `self.hostel_status` attribute with this new value.
- `display_details(self):`: This method prints the details of the student (name, roll number, and hostel status) to the console.
- `student1 = sru_student("Varshitha", "2403A52397", "No hostel")`: This line creates a new instance (object) of the `sru_student` class named `student1`. It initializes the object with the provided name, roll number, and hostel status.
- `student1.fee_update("No hostel")`: This line calls the `fee_update` method on the `student1` object to update the hostel status. In this specific case, it's being updated to "No hostel".
- `student1.display_details()`: This line calls the `display_details` method on the `student1` object to print the student's information.

Task Description#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.

```
import numpy as np

def add(a, b):
    """
    Adds two numbers.

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

    Returns
    -----
    float or int
        The sum of a and b.
    """
    return a + b

def subtract(a, b):
    """
    Subtracts the second number from the first.

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

    Returns
    -----
    float or int
        The difference between a and b.
    """
    return a - b
```

```

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

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

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

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

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

    Returns
    -----
    float or int or str
        The result of the division, or an error message if the denominator is zero.
    """
    if b == 0:
        return "Error: Division by zero"

```

```

numbers_list = [1, 2, 3, 4, 5]
result = sum_even_odd(numbers_list)
print("Sum of even numbers:", result[0])
print("Sum of odd numbers:", result[1])

```

```

Sum of even numbers: 6
Sum of odd numbers: 9

```

✦ Certainly. I can explain the code in cell 51882833, which contains the calculator functions.

This code defines four basic arithmetic functions: `add`, `subtract`, `multiply`, and `divide`.

- **Module-level docstring:** The triple-quoted string at the beginning of the cell is a module-level docstring. It provides a brief overview of what the module does.
- `import numpy as np`: This line imports the NumPy library, although it's not strictly necessary for these basic arithmetic functions. It might have been included for potential future use or as part of a larger script.
- `def add(a, b):`: This defines the `add` function, which takes two arguments, `a` and `b`.
 - The docstring within the `add` function explains its purpose, parameters (`a` and `b`), and what it returns (the sum of `a` and `b`).
 - `return a + b`: This line simply returns the sum of the two input numbers.
- `def subtract(a, b):`: This defines the `subtract` function, which takes two arguments, `a` and `b`.
 - The docstring explains its purpose, parameters, and return value (the difference between `a` and `b`).
 - `return a - b`: This line returns the result of subtracting `b` from `a`.
- `def multiply(a, b):`: This defines the `multiply` function, which takes two arguments, `a` and `b`.
 - The docstring explains its purpose, parameters, and return value (the product of `a` and `b`).
 - `return a * b`: This line returns the product of `a` and `b`.
- `def divide(a, b):`: This defines the `divide` function, which takes two arguments, `a` and `b`.
 - The docstring explains its purpose, parameters, and return value. It also mentions the special case of division by zero.
 - `if b == 0:`: This checks if the denominator `b` is zero.
 - `return "Error: Division by zero"`: If `b` is zero, it returns an error message string.
 - `return a / b`: If `b` is not zero, it returns the result of dividing `a` by `b`.