Assignment - 9.3

Name: D. Sravika Reddy Roll No: 2403a510d0

Batch No: 05 Course: AI Assisted Coding

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

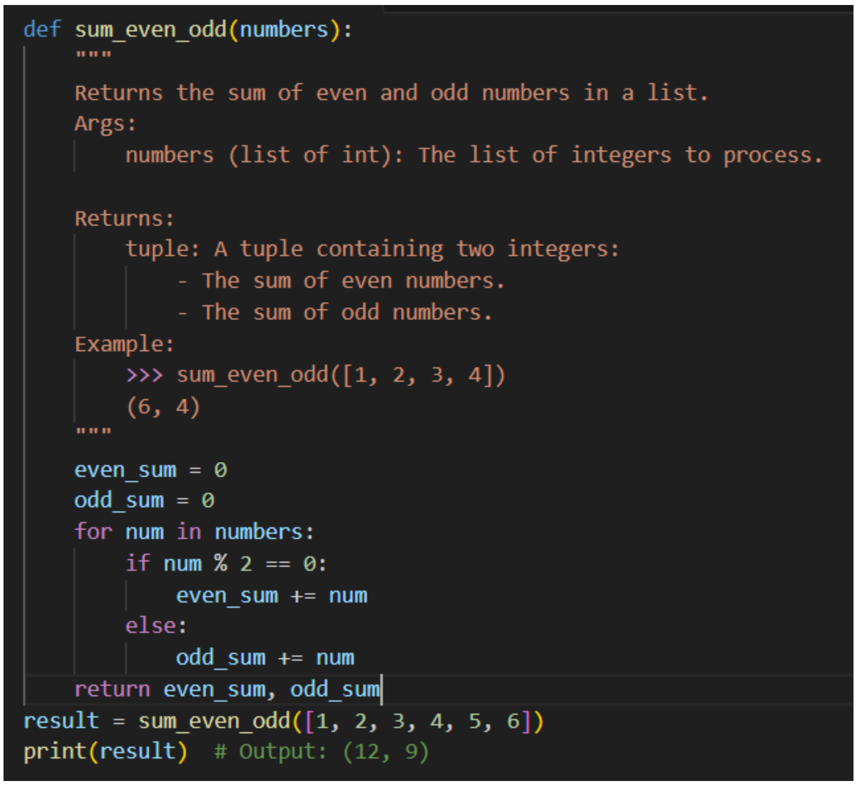
• Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing
the function.

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

Expected Outcome#1: Students understand how AI can produce function-level documentation

#Prompt: Python function to return sum of even and odd numbers

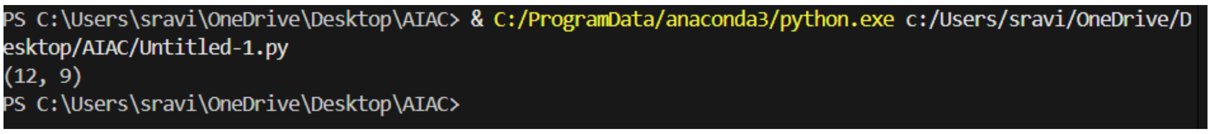
Code:



```python
def sum_even_odd(numbers):
    """
    Returns the sum of even and odd numbers in a list.
    Args:
        numbers (list of int): The list of integers to process.

    Returns:
        tuple: A tuple containing two integers:
            - The sum of even numbers.
            - The sum of odd numbers.
    Example:
        >>> sum_even_odd([1, 2, 3, 4])
        (6, 4)
    """
    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
result = sum_even_odd([1, 2, 3, 4, 5, 6])
print(result)  # Output: (12, 9)
```

Output:



```
PS C:\Users\sravi\OneDrive\Desktop\AIAC> & C:/ProgramData/anaconda3/python.exe c:/Users/sravi/OneDrive/D
esktop/AIAC/Untitled-1.py
(12, 9)
PS C:\Users\sravi\OneDrive\Desktop\AIAC>
```

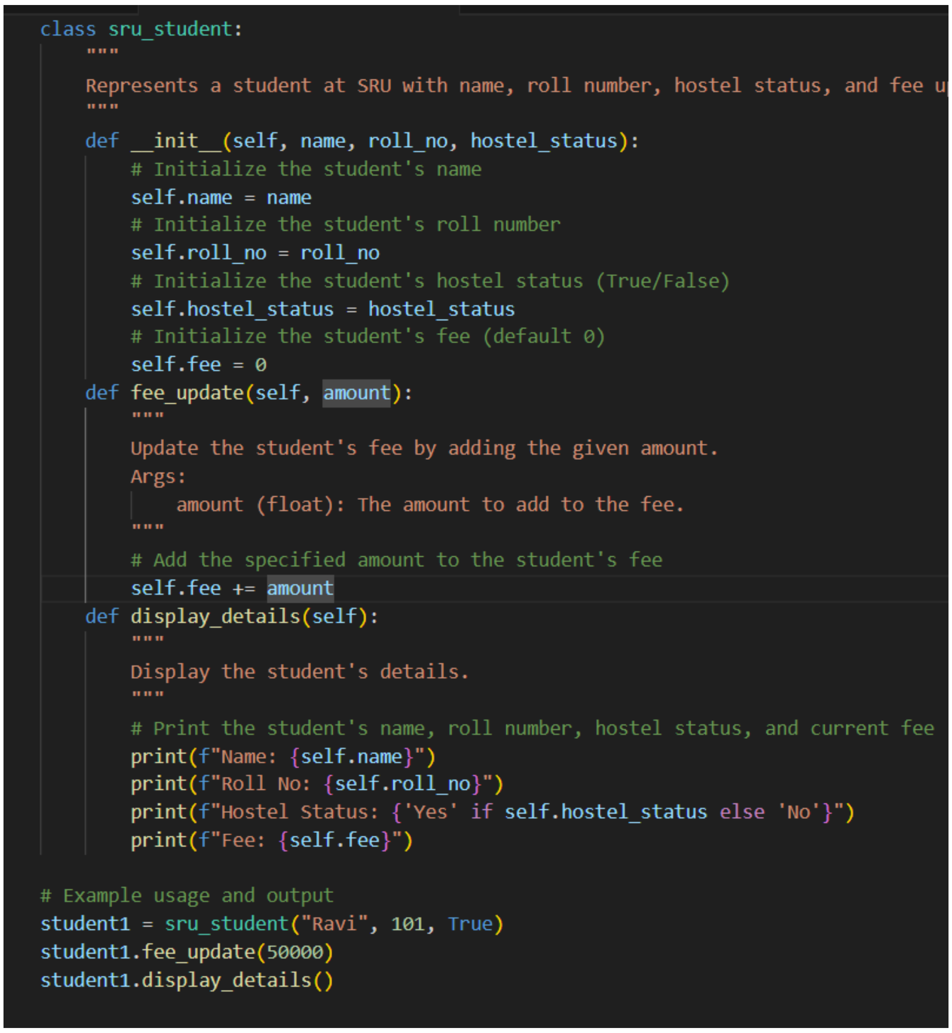
Explanation:

- The function iterates through the list, adding even numbers to even_sum and odd numbers to odd_sum.
- Returns a tuple: (sum of evens, sum of odds).
- The manual docstring uses Google Style and provides an example.
- The AI-generated docstring is concise and covers the essentials, but may lack detailed examples or formatting.
- **Comparison:** Manual docstrings can be more detailed and formatted, while AI-generated ones are quick and generally accurate for simple functions.

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.
Expected Output#2: Students critically analyze AI-generated code comments.

#Prompt: Python program for sru_student class with attributes like name, roll no, hostel_status and fee_update method and display_details method.

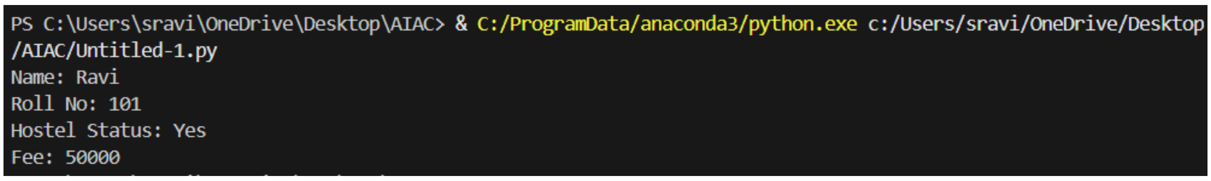
Code:



```python
class sru_student:
    """
    Represents a student at SRU with name, roll number, hostel status, and fee u
    """
    def __init__(self, name, roll_no, hostel_status):
        # Initialize the student's name
        self.name = name
        # Initialize the student's roll number
        self.roll_no = roll_no
        # Initialize the student's hostel status (True/False)
        self.hostel_status = hostel_status
        # Initialize the student's fee (default 0)
        self.fee = 0
    def fee_update(self, amount):
        """
        Update the student's fee by adding the given amount.
        Args:
            amount (float): The amount to add to the fee.
        """
        # Add the specified amount to the student's fee
        self.fee += amount
    def display_details(self):
        """
        Display the student's details.
        """
        # Print the student's name, roll number, hostel status, and current fee
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {'Yes' if self.hostel_status else 'No'}")
        print(f"Fee: {self.fee}")

# Example usage and output
student1 = sru_student("Ravi", 101, True)
student1.fee_update(50000)
student1.display_details()
```

Output:



```
PS C:\Users\sravi\OneDrive\Desktop\AIAC> & C:/ProgramData/anaconda3/python.exe c:/Users/sravi/OneDrive/Desktop
/AIAC/Untitled-1.py
Name: Ravi
Roll No: 101
Hostel Status: Yes
Fee: 50000
```

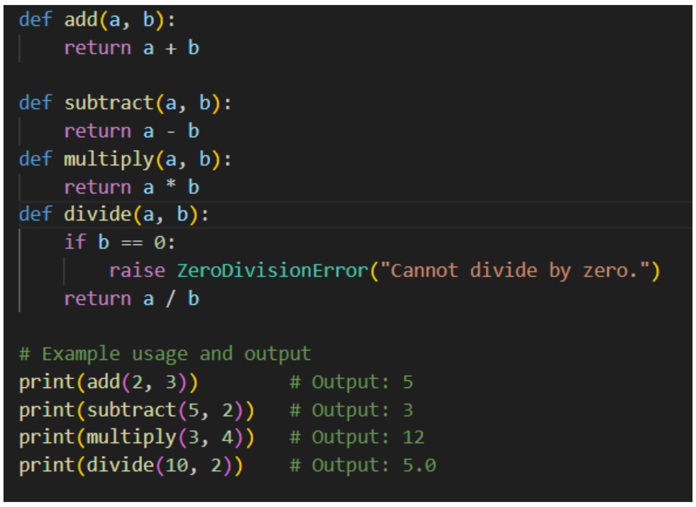
Explanation:

- The sru_student class models a student with attributes for name, roll number, hostel status, and fee.
- The fee_update method adds a specified amount to the student's fee.
- The display_details method prints all student details in a readable format.

- Manual comments are provided for each line/block.
- **To compare:**
  - AI-generated comments often describe each line or block, sometimes redundantly.
  - Manual comments can be more concise, focusing on logic or non-obvious steps.
- Both approaches help clarify code, but manual comments can be tailored for clarity and relevance.

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.
Expected Output#3: Students learn structured documentation for multi-function scripts

Code:

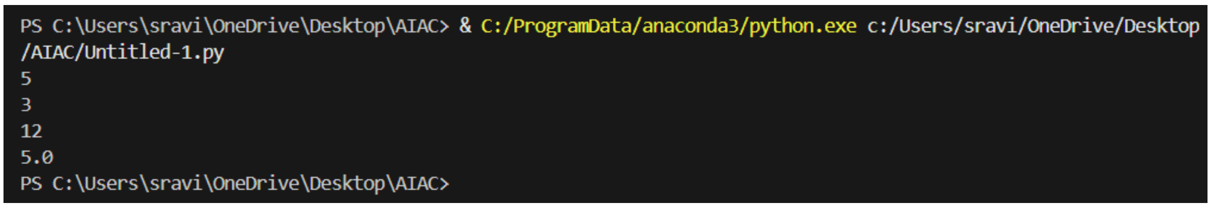


```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
def multiply(a, b):
    return a * b
def divide(a, b):
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b

# Example usage and output
print(add(2, 3))        # Output: 5
print(subtract(5, 2))   # Output: 3
print(multiply(3, 4))   # Output: 12
print(divide(10, 2))    # Output: 5.0
```

Output:



```
PS C:\Users\sravi\OneDrive\Desktop\AIAC> & C:/ProgramData/anaconda3/python.exe c:/Users/sravi/OneDrive/Desktop
/AIAC/Untitled-1.py
5
3
12
5.0
PS C:\Users\sravi\OneDrive\Desktop\AIAC>
```

Explanation:

- The script provides four arithmetic functions, each with a NumPy-style docstring.
- The module-level docstring (at the top) describes the module and gives usage examples.
- Each function's docstring details parameters, return values, and examples.
- Manual docstrings are detailed and structured; AI-generated docstrings are often concise and accurate, but may lack the full NumPy-style structure unless specifically prompted.
- This approach helps students learn best practices for documenting Python modules and functions.