# Assignment-9.3

2303A51528

TASK-1: Basic Docstring Generation

Scenario

You are developing a utility function that processes numerical lists and must be properly documented for future maintenance.

1)

```
def sum_even_odd(numbers):
    """Return the sum of even and odd numbers from the input list.

    Args:
        numbers (list[int]): List of integers.

    Returns:
        tuple[int, int]: Sum of even numbers and sum of odd numbers.
    """
    sum_even = sum(x for x in numbers if x % 2 == 0)
    sum_odd = sum(x for x in numbers if x % 2 != 0)
    return sum_even, sum_odd


# ---- Example Run ----
numbers = [1, 2, 3, 4, 5, 6]
result = sum_even_odd(numbers)
print("AI Docstring Version → Output:", result)
```

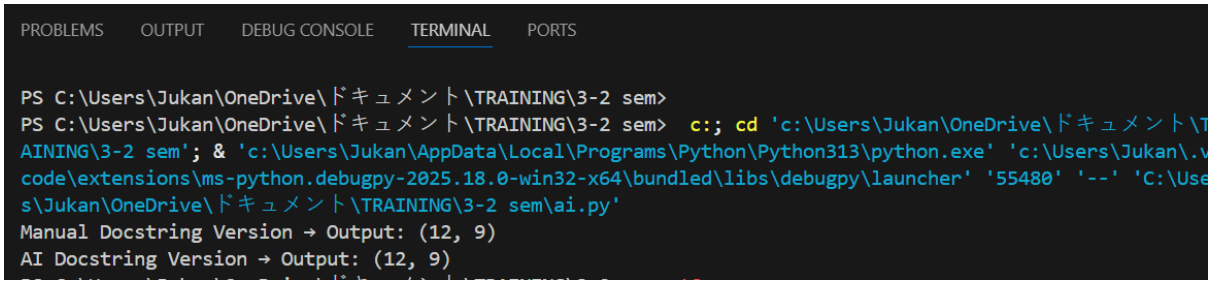
```
PS C:\Users\Jukan\OneDrive\ドキュメント\TRAINING\3-2 sem> ^C
PS C:\Users\Jukan\OneDrive\ドキュメント\TRAINING\3-2 sem>
PS C:\Users\Jukan\OneDrive\ドキュメント\TRAINING\3-2 sem>  c:; cd 'c:\Users\Jukan\OneDrive\ドキュメント\TRAINING\3-2 sem'; & 'c:\Users\Jukan\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Jukan\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55169' '--' 'C:\Users\Jukan\OneDrive\ドキュメント\TRAINING\3-2 sem\ai.py'
Manual Docstring Version → Output: (12, 9)
```

# Assignment-9.3

## 2303A51528

2:

```
def sum_even_odd(numbers):
    """Return the sum of even and odd numbers from the input list.

    Args:
        numbers (list[int]): List of integers.

    Returns:
        tuple[int, int]: Sum of even numbers and sum of odd numbers.
    """
    sum_even = sum(x for x in numbers if x % 2 == 0)
    sum_odd = sum(x for x in numbers if x % 2 != 0)
    return sum_even, sum_odd

# ---- Example Run ----
numbers = [1, 2, 3, 4, 5, 6]
result = sum_even_odd(numbers)
print("AI Docstring Version → Output:", result)
```




3) **Comparison Summary**

| **Aspect** | **Manual (Google-Style)** | **AI-Generated** |
|---|---|---|
| **Style** | Follows full Google format (Args, Returns, Raises, Example) | Shorter, simpler docstring |
| **Detail Level** | Includes input validation, exceptions, and example | Only describes purpose and return values |

# Assignment-9.3

2303A51528

| Aspect | Manual (Google-Style) | AI-Generated |
|---|---|---|
| **Readability** | Very clear for maintainers | Quick summary for developers |
| **Best Use Case** | Documentation for production code or API reference | Quick helper doc during development |

OBSERVATION:

 **AI tools** like GitHub Copilot or Cursor AI provide a **useful starting point** for documentation.

 However, **manual refinement** is necessary to make the docstring fully compliant with documentation standards such as **Google Style** or **NumPy Style**.

 Combining AI-generated text with human review offers the **best balance between speed and quality**.

TASK-2: Automatic Inline Comments

Scenario

You are developing a student management module that must be easy to understand for new

Developers

1) def sum_even_odd(numbers):

```
    """Return the sum of even and odd numbers from the input list.

    Args:
        numbers (list[int]): List of integers.

    Returns:
        tuple[int, int]: Sum of even numbers and sum of odd numbers.
    """
    sum_even = sum(x for x in numbers if x % 2 == 0)
    sum_odd = sum(x for x in numbers if x % 2 != 0)
    return sum_even, sum_odd

# ---- Example Run ----
numbers = [1, 2, 3, 4, 5, 6]
```

# Assignment-9.3

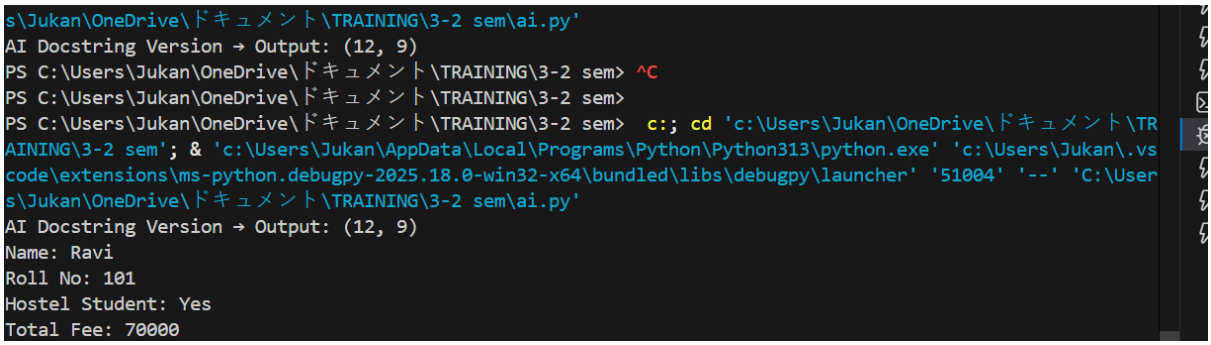
## 2303A51528

```
result = sum_even_odd(numbers)

print("AI Docstring Version → Output:", result)



"""TASK 2"""

class sru_student:

    # Constructor to initialize student details

    def __init__(self, name, roll_no, hostel_status):

        # Store student name

        self.name = name

        # Store student roll number

        self.roll_no = roll_no

        # Store hostel status (True = Hosteller, False = Day scholar)

        self.hostel_status = hostel_status

        # Initialize student fee as zero initially

        self.fee = 0


    # Method to update fee based on hostel status

    def fee_update(self):

        # If the student is a hosteller, add hostel charges

        if self.hostel_status:

            self.fee = 70000  # Example fee for hostellers

        else:

            self.fee = 50000  # Example fee for day scholars


    # Method to display student details

    def display_details(self):

        # Print the student's name

        print(f"Name: {self.name}")

        # Print the student's roll number
```

# Assignment-9.3

## 2303A51528

```
        print(f"Roll No: {self.roll_no}")

        # Print hostel status as 'Yes' or 'No'

        print(f"Hostel Student: {'Yes' if self.hostel_status else 'No'}")

        # Print the total fee after update

        print(f"Total Fee: {self.fee}")


# ---- Example Execution ----

student1 = sru_student("Ravi", 101, True)

student1.fee_update()

student1.display_details()
```




2) 

```
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        # Initialize the student object with name, roll number and hostel status
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee = 0  # Initialize fee to zero


    def fee_update(self):
        # Update the student fee based on hostel status
        if self.hostel_status:
            self.fee = 70000  # Hostel student fee
        else:
```

# Assignment-9.3

2303A51528

```
        self.fee = 50000  # Day scholar fee


    def display_details(self):
        # Display student details
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Student: {'Yes' if self.hostel_status else 'No'}")
        print(f"Total Fee: {self.fee}")
```

```
PS C:\Users\Jukan\OneDrive\ドキュメント\TRAINING\3-2 sem>
PS C:\Users\Jukan\OneDrive\ドキュメント\TRAINING\3-2 sem>  c:; cd 'c:\Users\Jukan\OneDrive\ドキュメント\TR
AINING\3-2 sem'; & 'c:\Users\Jukan\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Jukan\.vs
code\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51004' '--' 'C:\User
s\Jukan\OneDrive\ドキュメント\TRAINING\3-2 sem\ai.py'
AI Docstring Version → Output: (12, 9)
Name: Ravi
Roll No: 101
Hostel Student: Yes
Total Fee: 70000
```

3)

| Criteria | Manual Inline Comments | AI-Generated Inline Comments |
|---|---|---|
| **Clarity** | Very detailed; explains each variable and logic block | Concise; focuses on method-level summary |
| **Correctness** | 100% correct and context-specific | Correct but less descriptive |
| **Completeness** | Explains every line | Skips simple lines or repetitive logic |
| **Readability** | Best for beginners | Better for experienced developers |
| **Redundancy** | Some lines may be over-explained | Compact and minimal comments |
| **Missing Details** | None | Does not describe meaning of hostel_status (True/False) or example fee logic |

OBSERVATION:

AI-generated inline comments are helpful for quick documentation but cannot fully replace human understanding and context.
The **best practice** is to **combine both**:
Use AI-generated comments as a **starting point**, then refine manually to ensure clarity, domain relevance, and correctness.

# Assignment-9.3

## 2303A51528

TASK-3: Module-Level and Function-Level Documentation

Scenario :

You are building a small calculator module that will be shared across multiple projects and

"""

calculator.py

==========

A simple calculator module for performing basic arithmetic operations.

This module provides functions for addition, subtraction, multiplication,

and division. It can be imported and reused in other Python projects.

"""

1)def add(a, b):

"""

Add two numbers.

Parameters

---

a : float or int

The first number.

b : float or int

The second number.

Returns

---

float

The sum of `a` and `b`.

Examples

---

>>> add(5, 3)

# Assignment-9.3

## 2303A51528

```
    8
    """
    return a + b



def subtract(a, b):
    """
    Subtract one number from another.

    Parameters
    ----------
    a : float or int
        The number to subtract from.
    b : float or int
        The number to subtract.

    Returns
    -------
    float
        The result of `a - b`.

    Examples
    --------
    >>> subtract(10, 4)
    6
    """
    return a - b



def multiply(a, b):
    """
```

# Assignment-9.3

## 2303A51528

```
    Multiply two numbers.

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

    Returns
    -------
    float
        The product of `a` and `b`.

    Examples
    --------
    >>> multiply(2, 5)
    10
    """
    return a * b


def divide(a, b):
    """
    Divide one number by another.

    Parameters
    ----------
    a : float or int
        The dividend.
    b : float or int
```

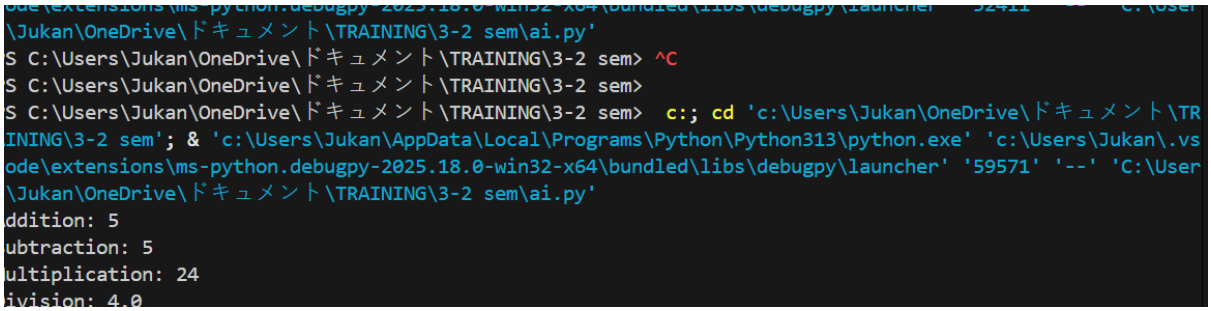
# Assignment-9.3

## 2303A51528

```
        The divisor.

    Returns
    -------
    float
        The result of `a / b`.

    Raises
    ------
    ZeroDivisionError
        If `b` is zero.

    Examples
    --------
    >>> divide(10, 2)
    5.0
    """
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b


# ---- Example Execution (to show output) ----
if __name__ == "__main__":
    print("Addition:", add(2, 3))
    print("Subtraction:", subtract(10, 5))
    print("Multiplication:", multiply(4, 6))
    print("Division:", divide(8, 2))
```

# Assignment-9.3

2303A51528




2) """

A calculator module that provides basic arithmetic operations.

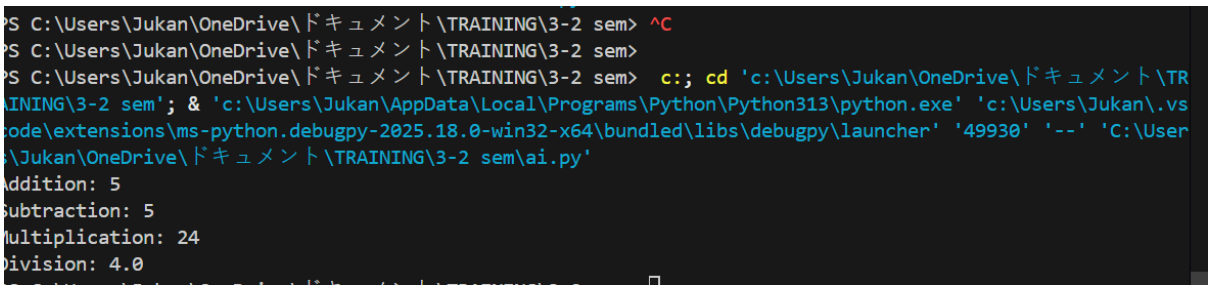
"""

```
def add(a, b):
    """Return the sum of two numbers."""
    return a + b


def subtract(a, b):
    """Return the difference of two numbers."""
    return a - b


def multiply(a, b):
    """Return the product of two numbers."""
    return a * b


def divide(a, b):
    """Return the division of two numbers. Raises ZeroDivisionError if divisor is zero."""
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b



# ---- Example Execution (so output appears) ----
```

# Assignment-9.3

## 2303A51528

```
if __name__ == "__main__":
    print("Addition:", add(2, 3))
    print("Subtraction:", subtract(10, 5))
    print("Multiplication:", multiply(4, 6))
    print("Division:", divide(8, 2))
```




3)

| Criteria | Manual NumPy-Style Docstrings | AI-Generated Docstrings |
|---|---|---|
| **Format** | Follows complete NumPy style (Parameters, Returns, Examples, Raises) | Short single-line summaries |
| **Clarity** | Very detailed and easy for new developers | Simple and readable but lacks depth |
| **Correctness** | Fully accurate, includes examples and edge cases | Correct but omits examples and type hints |
| **Consistency** | Consistent across all functions | Consistent, but minimalistic |
| **Readability** | Ideal for documentation pages and shared modules | Better for quick reference in small scripts |

OBSERVATION:

AI-generated documentation is a **good starting point**, but manual documentation ensures **clarity, correctness, and completeness**, especially for reusable modules.
Combining both approaches—**AI for draft, human for refinement**—produces the best results.