]SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName:B. Tech		**Assignment Type: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
rCourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week5-Wednesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:9.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	**Question**	***ExpectedTime to complete***
1	Lab 8: Documentation Generation: Automatic documentation and code comments **Lab Objectives:** • To understand the importance of documentation and code comments in software development. • To explore how AI-assisted coding tools can generate meaningful documentation and	Week4 - Wednesday

inline comments.

- To practice generating function-level and module-level docstrings automatically.
- To evaluate the quality, accuracy, and limitations of AI-generated documentation.
- To develop a small automated tool for documentation generation in Python..

Lab Outcomes (LOs):
After completing this lab, students will be able to:

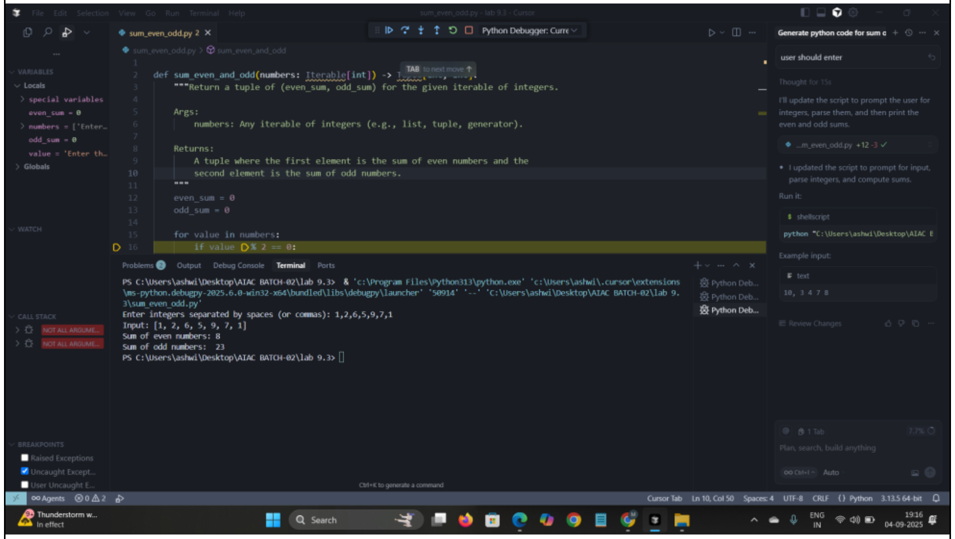
- Apply AI-assisted coding tools to generate docstrings and inline comments for Python code.
- Critically analyze AI-generated documentation for correctness, completeness, and readability.
- Create structured documentation (function-level, module-level) following standard formats.

- Design and implement a mini documentation generator tool to automate code commenting and docstring creation.

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



```python

def sum_even_and_odd(numbers: Iterable[int]) -> Tuple[int, int]:
    """Return a tuple of (even_sum, odd_sum) for the given iterable of integers.

    Args:
        numbers: Any iterable of integers (e.g., list, tuple, generator).

    Returns:
        A tuple where the first element is the sum of even numbers and the
        second element is the sum of odd numbers.
    """
    even_sum = 0
    odd_sum = 0

    for value in numbers:
        if value % 2 == 0:
```

```
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>  & 'c:\Program Files\Python313\python.exe' 'c:\Users\ashwi\.cursor\extensions
\ms-python.debugpy-2025.6.0-win32-x64\bundled\libs\debugpy\launcher' '50914' '--' 'C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.
3\sum_even_odd.py'
Enter integers separated by spaces (or commas): 1,2,6,5,9,7,1
Input: [1, 2, 6, 5, 9, 7, 1]
Sum of even numbers: 8
Sum of odd numbers:  23
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3> 
```

user should enter

Thought for 15s

I'll update the script to prompt the user for integers, parse them, and then print the even and odd sums.

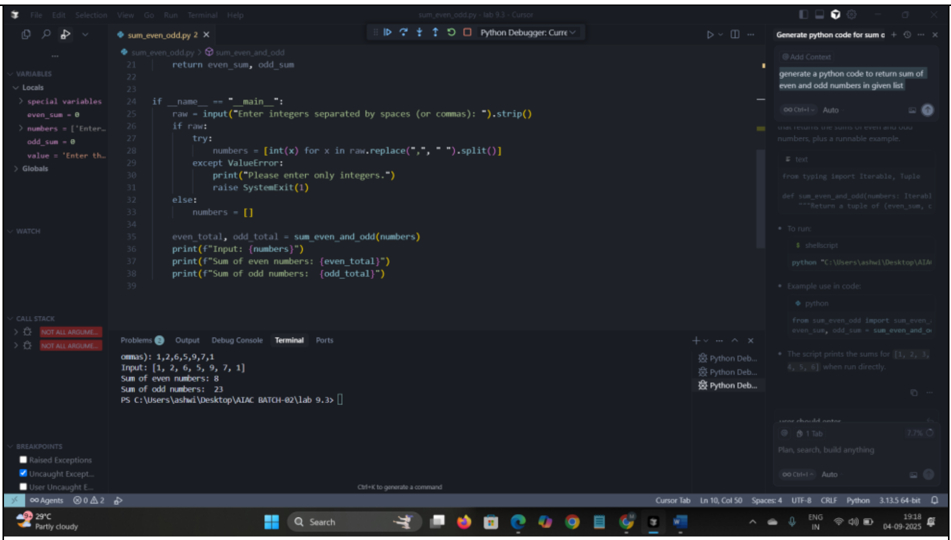
- I updated the script to prompt for input, parse integers, and compute sums.

Run it:

```shellscript
python "C:\Users\ashwi\Desktop\AIAC B
```

Example input:

```text
10, 3 4 7 8
```



```
    return even_sum, odd_sum


if __name__ == "__main__":
    raw = input("Enter integers separated by spaces (or commas): ").strip()
    if raw:
        try:
            numbers = [int(x) for x in raw.replace(",", " ").split()]
        except ValueError:
            print("Please enter only integers.")
            raise SystemExit(1)
    else:
        numbers = []

    even_total, odd_total = sum_even_and_odd(numbers)
    print(f"Input: {numbers}")
    print(f"Sum of even numbers: {even_total}")
    print(f"Sum of odd numbers:  {odd_total}")
```

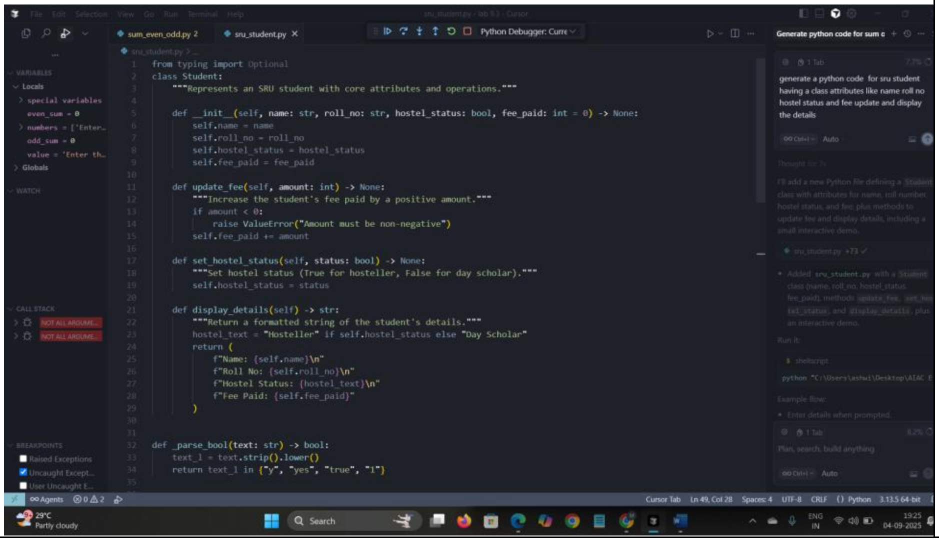
```
ommas): 1,2,6,5,9,7,1
Input: [1, 2, 6, 5, 9, 7, 1]
Sum of even numbers: 8
Sum of odd numbers:  23
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>
```

generate a python code to return sum of even and odd numbers in given list

Task Description#2 Automatic Inline Comments

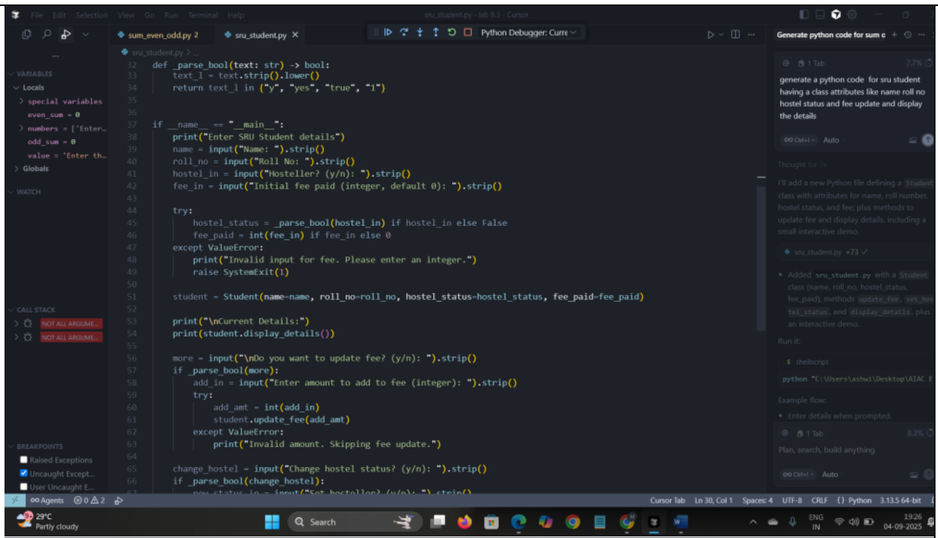
- Write python program for **sru_student** class with attributes like name, roll no., hostel_status and **fee_update** method and **display_details** method.
- Write comments manually for each line/code block
- Ask an AI tool to add inline comments explaining each line/step.
- Compare the AI-generated comments with your manually written one.

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



```
from typing import Optional
class Student:
    """Represents an SRU student with core attributes and operations."""

    def __init__(self, name: str, roll_no: str, hostel_status: bool, fee_paid: int = 0) -> None:
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = fee_paid

    def update_fee(self, amount: int) -> None:
        """Increase the student's fee paid by a positive amount."""
        if amount < 0:
            raise ValueError("Amount must be non-negative")
        self.fee_paid += amount

    def set_hostel_status(self, status: bool) -> None:
        """Set hostel status (True for hosteller, False for day scholar)."""
        self.hostel_status = status

    def display_details(self) -> str:
        """Return a formatted string of the student's details."""
        hostel_text = "Hosteller" if self.hostel_status else "Day Scholar"
        return (
            f"Name: {self.name}\n"
            f"Roll No: {self.roll_no}\n"
            f"Hostel Status: {hostel_text}\n"
            f"Fee Paid: {self.fee_paid}"
        )


def _parse_bool(text: str) -> bool:
    text_l = text.strip().lower()
    return text_l in {"y", "yes", "true", "1"}
```

generate a python code for sru student having a class attributes like name roll no hostel status and fee update and display the details



```python
def _parse_bool(text: str) -> bool:
    text_l = text.strip().lower()
    return text_l in {"y", "yes", "true", "1"}


if __name__ == "__main__":
    print("Enter SRU Student details")
    name = input("Name: ").strip()
    roll_no = input("Roll No: ").strip()
    hostel_in = input("Hosteller? (y/n): ").strip()
    fee_in = input("Initial fee paid (integer, default 0): ").strip()

    try:
        hostel_status = _parse_bool(hostel_in) if hostel_in else False
        fee_paid = int(fee_in) if fee_in else 0
    except ValueError:
        print("Invalid input for fee. Please enter an integer.")
        raise SystemExit(1)

    student = Student(name=name, roll_no=roll_no, hostel_status=hostel_status, fee_paid=fee_paid)

    print("\nCurrent Details:")
    print(student.display_details())

    more = input("\nDo you want to update fee? (y/n): ").strip()
    if _parse_bool(more):
        add_in = input("Enter amount to add to fee (integer): ").strip()
        try:
            add_amt = int(add_in)
            student.update_fee(add_amt)
        except ValueError:
            print("Invalid amount. Skipping fee update.")

    change_hostel = input("Change hostel status? (y/n): ").strip()
    if _parse_bool(change_hostel):
```

generate a python code for sru student having a class attributes like name roll no hostel status and fee update and display the details

Thought for 7s

I'll add a new Python file defining a `Student` class with attributes for name, roll number, hostel status, and fee; plus methods to update fee and display details, including a small interactive demo.

sru_student.py +73 ✓

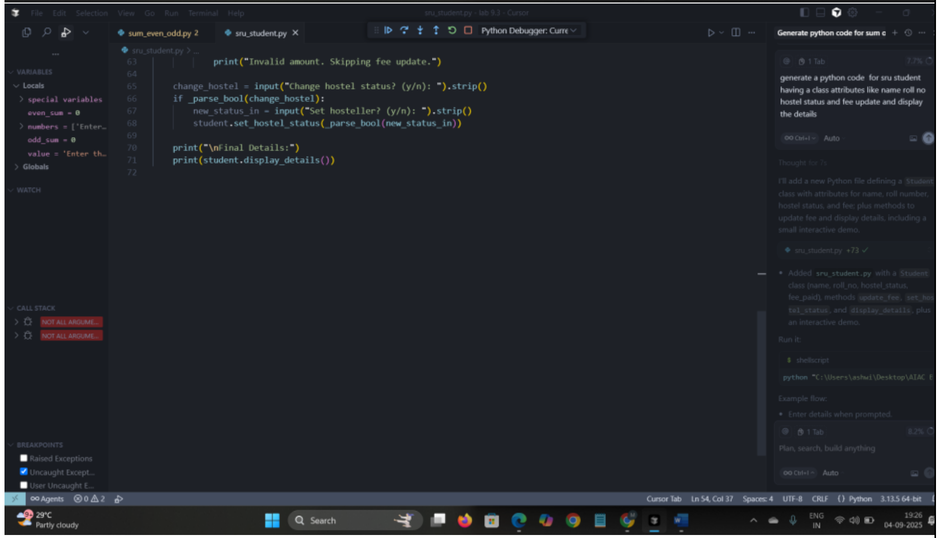
- Added `sru_student.py` with a `Student` class (name, roll_no, hostel_status, fee_paid), methods `update_fee`, `set_hostel_status`, and `display_details`, plus an interactive demo.

Run it:

```shellscript
python "C:\Users\ashwi\Desktop\AIAC
```

Example flow:

- Enter details when prompted.

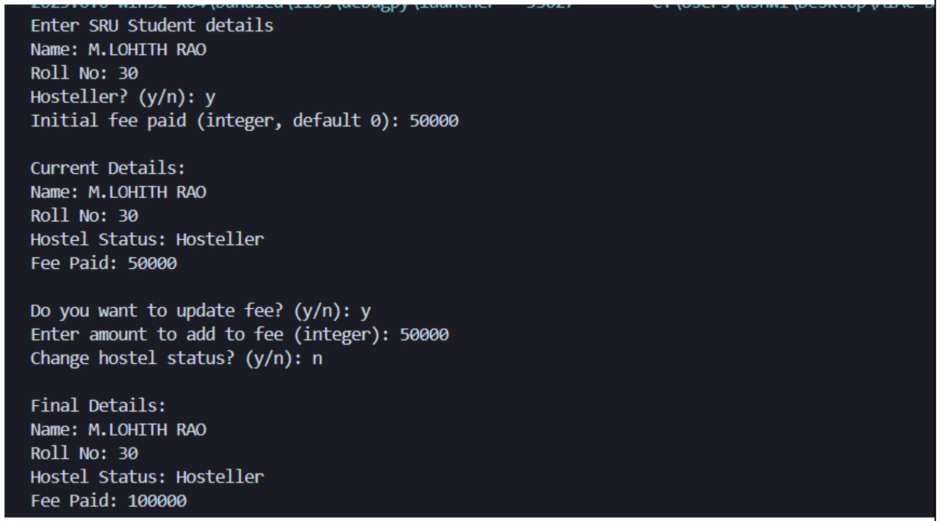


```python
            print("Invalid amount. Skipping fee update.")

    change_hostel = input("Change hostel status? (y/n): ").strip()
    if _parse_bool(change_hostel):
        new_status_in = input("Set hosteller? (y/n): ").strip()
        student.set_hostel_status(_parse_bool(new_status_in))

    print("\nFinal Details:")
    print(student.display_details())
```

generate a python code for sru student having a class attributes like name roll no hostel status and fee update and display the details

I'll add a new Python file defining a `Student` class with attributes for name, roll number, hostel status, and fee; plus methods to update fee and display details, including a small interactive demo.

- Added `sru_student.py` with a `Student` class (name, roll_no, hostel_status, fee_paid), methods `update_fee`, `set_hostel_status`, and `display_details`, plus an interactive demo.

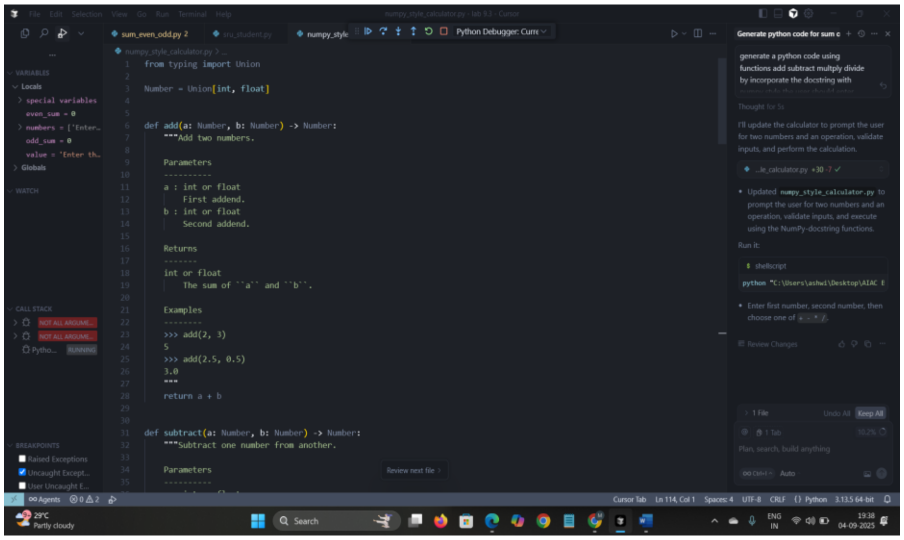


```
Enter SRU Student details
Name: M.LOHITH RAO
Roll No: 30
Hosteller? (y/n): y
Initial fee paid (integer, default 0): 50000

Current Details:
Name: M.LOHITH RAO
Roll No: 30
Hostel Status: Hosteller
Fee Paid: 50000

Do you want to update fee? (y/n): y
Enter amount to add to fee (integer): 50000
Change hostel status? (y/n): n

Final Details:
Name: M.LOHITH RAO
Roll No: 30
Hostel Status: Hosteller
Fee Paid: 100000
```

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



```python
from typing import Union

Number = Union[int, float]


def add(a: Number, b: Number) -> Number:
    """Add two numbers.

    Parameters
    ----------
    a : int or float
        First addend.
    b : int or float
        Second addend.

    Returns
    -------
    int or float
        The sum of ``a`` and ``b``.

    Examples
    --------
    >>> add(2, 3)
    5
    >>> add(2.5, 0.5)
    3.0
    """
    return a + b


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

    Parameters
    ----------
```

Generate python code for sum c

generate a python code using functions add subtract multply divide by incorporate the docstring with

Thought for 5s

I'll update the calculator to prompt the user for two numbers and an operation, validate inputs, and perform the calculation.

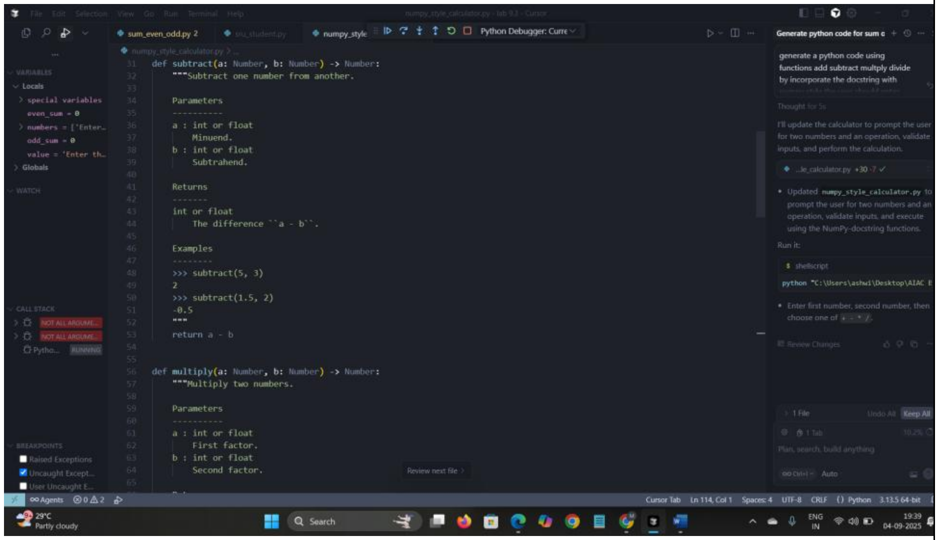
...le_calculator.py +30 -7 ✓

- Updated `numpy_style_calculator.py` to prompt the user for two numbers and an operation, validate inputs, and execute using the NumPy-docstring functions.

Run it:

```shellscript
python "C:\Users\ashwi\Desktop\AIAC E
```

- Enter first number, second number, then choose one of `+ - * /`.



```python
def subtract(a: Number, b: Number) -> Number:
    """Subtract one number from another.

    Parameters
    ----------
    a : int or float
        Minuend.
    b : int or float
        Subtrahend.

    Returns
    -------
    int or float
        The difference ``a - b``.

    Examples
    --------
    >>> subtract(5, 3)
    2
    >>> subtract(1.5, 2)
    -0.5
    """
    return a - b


def multiply(a: Number, b: Number) -> Number:
    """Multiply two numbers.

    Parameters
    ----------
    a : int or float
        First factor.
    b : int or float
        Second factor.
```

Generate python code for sum c

generate a python code using functions add subtract multply divide by incorporate the docstring with

Thought for 5s

I'll update the calculator to prompt the user for two numbers and an operation, validate inputs, and perform the calculation.

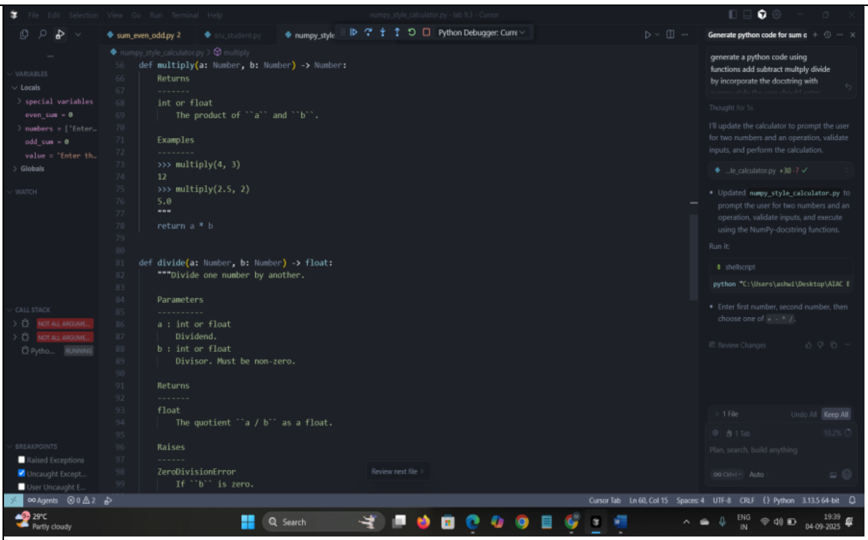
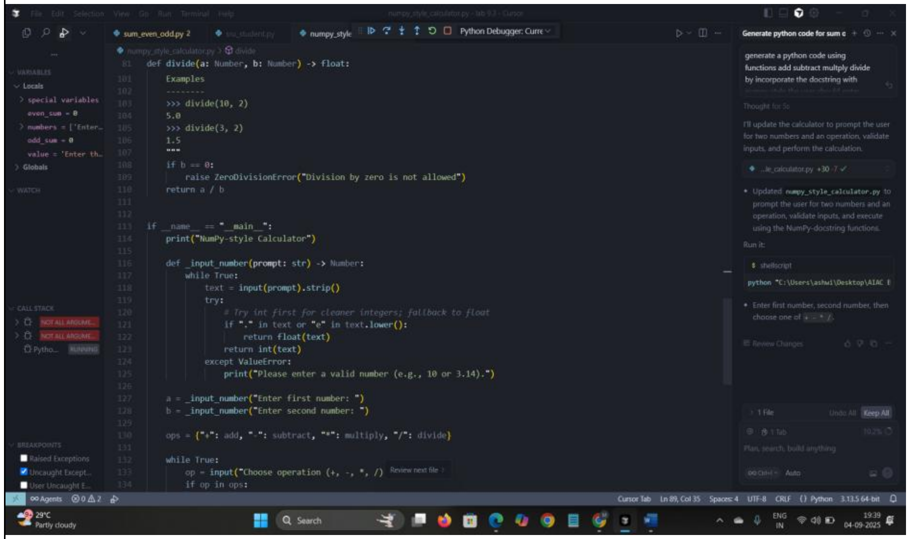
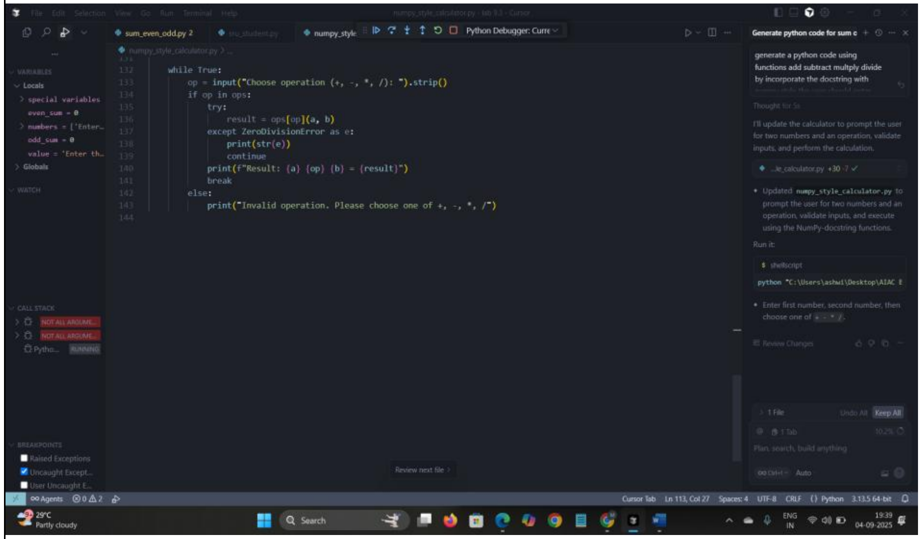
...le_calculator.py +30 -7 ✓

- Updated `numpy_style_calculator.py` to prompt the user for two numbers and an operation, validate inputs, and execute using the NumPy-docstring functions.

Run it:

python "C:\Users\ashwi\Desktop\AIAC E

- Enter first number, second number, then choose one of `+ - * /`.

numpy_style_calculator.py > multiply

```python
def multiply(a: Number, b: Number) -> Number:
    Returns
    -------
    int or float
        The product of ``a`` and ``b``.

    Examples
    --------
    >>> multiply(4, 3)
    12
    >>> multiply(2.5, 2)
    5.0
    """
    return a * b


def divide(a: Number, b: Number) -> float:
    """Divide one number by another.

    Parameters
    ----------
    a : int or float
        Dividend.
    b : int or float
        Divisor. Must be non-zero.

    Returns
    -------
    float
        The quotient ``a / b`` as a float.

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

Generate python code for sum o

generate a python code using functions add subtract multply divide by incorporate the docstring with

Thought for 5s

I'll update the calculator to prompt the user for two numbers and an operation, validate inputs, and perform the calculation.

...le_calculator.py +30 -7 ✓

- Updated `numpy_style_calculator.py` to prompt the user for two numbers and an operation, validate inputs, and execute using the NumPy-docstring functions.

Run it:

```shellscript
python "C:\Users\ashwi\Desktop\AIAC E
```

- Enter first number, second number, then choose one of `+ - * /`.

Ln 60, Col 15   Spaces: 4   UTF-8   CRLF   Python   3.13.5 64-bit

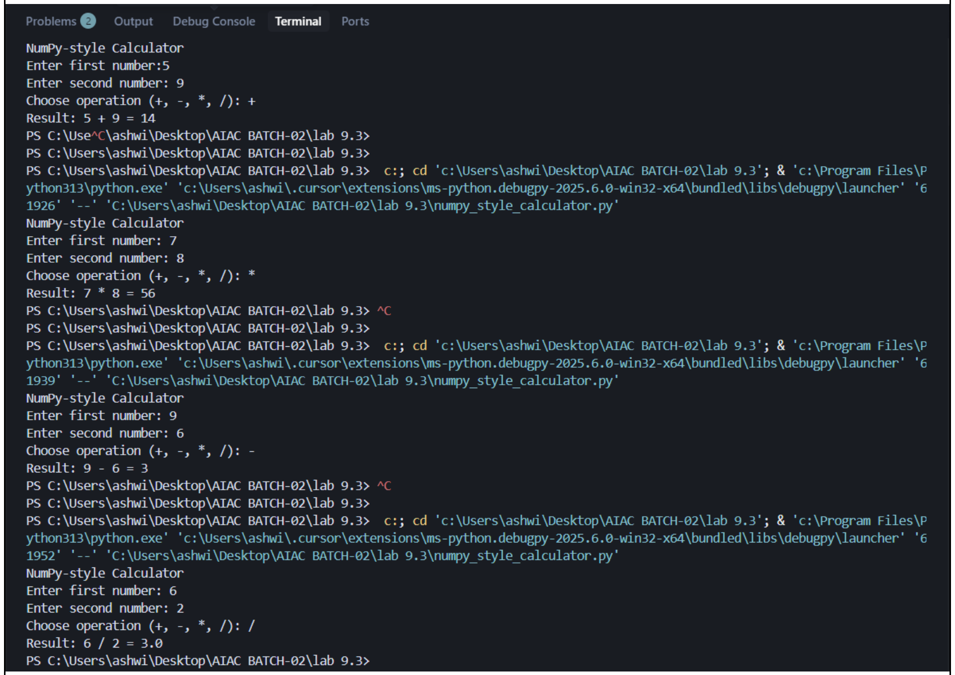
numpy_style_calculator.py > divide

```python
def divide(a: Number, b: Number) -> float:
    Examples
    --------
    >>> divide(10, 2)
    5.0
    >>> divide(3, 2)
    1.5
    """
    if b == 0:
        raise ZeroDivisionError("Division by zero is not allowed")
    return a / b


if __name__ == "__main__":
    print("NumPy-style Calculator")

    def _input_number(prompt: str) -> Number:
        while True:
            text = input(prompt).strip()
            try:
                # Try int first for cleaner integers; fallback to float
                if "." in text or "e" in text.lower():
                    return float(text)
                return int(text)
            except ValueError:
                print("Please enter a valid number (e.g., 10 or 3.14).")

    a = _input_number("Enter first number: ")
    b = _input_number("Enter second number: ")

    ops = {"+": add, "-": subtract, "*": multiply, "/": divide}

    while True:
        op = input("Choose operation (+, -, *, /)
        if op in ops:
```

Ln 89, Col 35   Spaces: 4   UTF-8   CRLF   Python   3.13.5 64-bit

```python
    while True:
        op = input("Choose operation (+, -, *, /): ").strip()
        if op in ops:
            try:
                result = ops[op](a, b)
            except ZeroDivisionError as e:
                print(str(e))
                continue
            print(f"Result: {a} {op} {b} = {result}")
            break
        else:
            print("Invalid operation. Please choose one of +, -, *, /")
```

Ln 113, Col 27   Spaces: 4   UTF-8   CRLF   Python   3.13.5 64-bit



```
NumPy-style Calculator
Enter first number:5
Enter second number: 9
Choose operation (+, -, *, /): +
Result: 5 + 9 = 14
PS C:\Use^C\ashwi\Desktop\AIAC BATCH-02\lab 9.3>
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>  c:; cd 'c:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3'; & 'c:\Program Files\P
ython313\python.exe' 'c:\Users\ashwi\.cursor\extensions\ms-python.debugpy-2025.6.0-win32-x64\bundled\libs\debugpy\launcher' '6
1926' '--' 'C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3\numpy_style_calculator.py'
NumPy-style Calculator
Enter first number: 7
Enter second number: 8
Choose operation (+, -, *, /): *
Result: 7 * 8 = 56
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3> ^C
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>  c:; cd 'c:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3'; & 'c:\Program Files\P
ython313\python.exe' 'c:\Users\ashwi\.cursor\extensions\ms-python.debugpy-2025.6.0-win32-x64\bundled\libs\debugpy\launcher' '6
1939' '--' 'C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3\numpy_style_calculator.py'
NumPy-style Calculator
Enter first number: 9
Enter second number: 6
Choose operation (+, -, *, /): -
Result: 9 - 6 = 3
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3> ^C
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>  c:; cd 'c:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3'; & 'c:\Program Files\P
ython313\python.exe' 'c:\Users\ashwi\.cursor\extensions\ms-python.debugpy-2025.6.0-win32-x64\bundled\libs\debugpy\launcher' '6
1952' '--' 'C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3\numpy_style_calculator.py'
NumPy-style Calculator
Enter first number: 6
Enter second number: 2
Choose operation (+, -, *, /): /
Result: 6 / 2 = 3.0
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 9.3>
```

Push documentation whole workspace as .md file in GitHub Repository

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots