| SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE | | DEPARTMENT OF COMPUTER SCIENCE ENGINEERING | |
|---|---|---|---|
| NAME:P.SRIDHAR<br>ENROLL NO.:2403A53013<br>BATCH NO.:24BTCAICYB01 | | Assignment Type: Lab | AcademicYear:2025-2026 |
| CourseCoordinatorName | | Venkataramana Veeramsetty | |
| Instructor(s)Name | | Dr. V. Venkataramana (Co-ordinator)<br>Dr. T. Sampath Kumar<br>Dr. Pramoda Patro<br>Dr. Brij Kishor Tiwari<br>Dr.J.Ravichander<br>Dr. Mohammand Ali Shaik<br>Dr. Anirodh Kumar<br>Mr. S.Naresh Kumar<br>Dr. RAJESH VELPULA<br>Mr. Kundhan Kumar<br>Ms. Ch.Rajitha<br>Mr. M Prakash<br>Mr. B.Raju<br>Intern 1 (Dharma teja)<br>Intern 2 (Sai Prasad)<br>Intern 3 (Sowmya)<br>NS_2 ( Mounika) | |
| CourseCode | 24CS002PC215 | CourseTitle | AI Assisted Coding |
| Year/Sem | II/I | Regulation | R24 |
| Date and Day of Assignment | Week4 - 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 9: Documentation Generation: Automatic documentation and code comments<br><br>**Lab Objectives:** | Week4 - Wednesday |

- To understand the importance of documentation and code comments in software development.
- To explore how AI-assisted coding tools can generate meaningful documentation and 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:

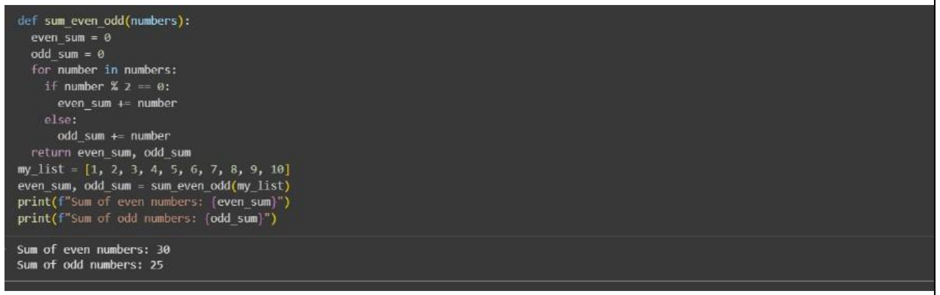
- Apply AI-assisted coding tools to generate docstrings and inline comments for Python code.
- Critically analyze AI-generated documentation for correctness, completeness, and readability.
- Create structured documentation (function-level, module-level) following standard formats.

- 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., Gemini, 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_odd(numbers):
  even_sum = 0
  odd_sum = 0
  for number in numbers:
    if number % 2 == 0:
      even_sum += number
    else:
      odd_sum += number
  return even_sum, odd_sum
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_even_odd(my_list)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

```
Sum of even numbers: 30
Sum of odd numbers: 25
```

**Manual Docstring (Google Style):**

- **More clear and detailed.**
- **Follows proper documentation format.**
- **Gives exact data types and return info.**
- **Best for professional or team projects.**

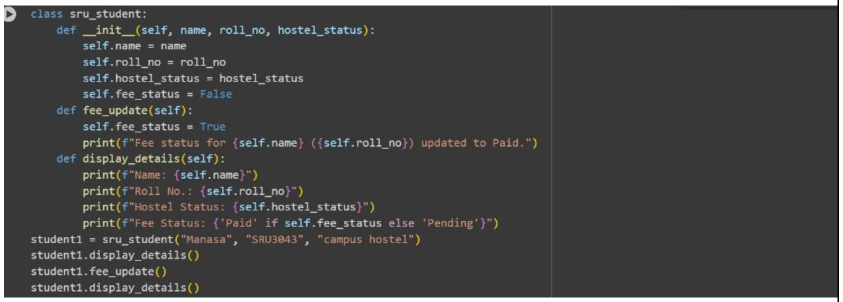
**AI-Generated Docstring:**

- **Short and basic.**
- **Doesn't follow a specific format.**
- **Less detail in parameters and return.**
- **Good for quick or personal use.**

**Conclusion:**
**Manual docstring is more accurate and professional.**
**AI docstring is faster but less complete.**

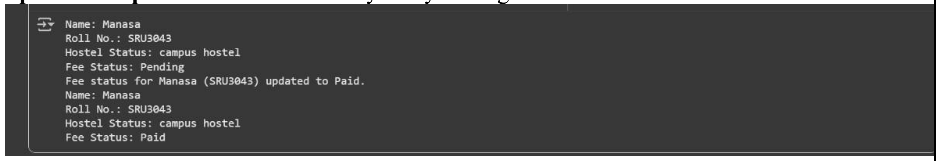
**Task Description#2 Automatic Inline Comments**

- 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.



```python
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_status = False
    def fee_update(self):
        self.fee_status = True
        print(f"Fee status for {self.name} ({self.roll_no}) updated to Paid.")
    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll No.: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Status: {'Paid' if self.fee_status else 'Pending'}")
student1 = sru_student("Manasa", "SRU3043", "campus hostel")
student1.display_details()
student1.fee_update()
student1.display_details()
```

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



```
Name: Manasa
Roll No.: SRU3043
Hostel Status: campus hostel
Fee Status: Pending
Fee status for Manasa (SRU3043) updated to Paid.
Name: Manasa
Roll No.: SRU3043
Hostel Status: campus hostel
Fee Status: Paid
```

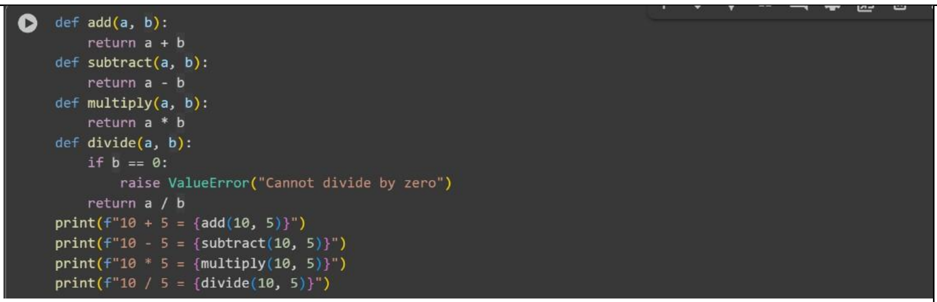
**Comparison:**

- **Manual Comments:**
  - **More descriptive and clear for each line.**
  - **Written with proper context and understanding.**
  - **Suitable for documentation or teaching.**
- **AI-Generated Comments:**
  - **Mostly correct, but some lines are too generic.**
  - **Faster to get, but might miss specific intent.**
  - **Good for quick help or personal use.**

**Conclusion:**
**Manual comments are more accurate and detailed.**
**AI comments are helpful, but not always perfect.**
**Manual is better for learning and real documentation.**

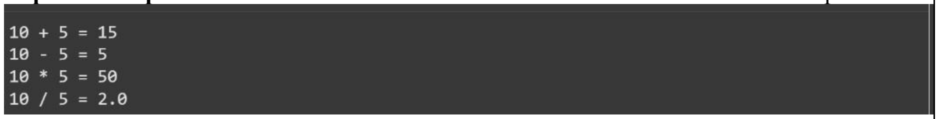
**Task Description#3**

- 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.



```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 ValueError("Cannot divide by zero")
    return a / b
print(f"10 + 5 = {add(10, 5)}")
print(f"10 - 5 = {subtract(10, 5)}")
print(f"10 * 5 = {multiply(10, 5)}")
print(f"10 / 5 = {divide(10, 5)}")
```

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



```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
```

**Comparison of Manual vs AI Docstrings:**

**Manual Docstrings (NumPy Style):**

- **Follows standard scientific documentation style.**
- **Clearly mentions parameters, return types, and exceptions.**
- **Best for professional, academic, or team coding projects.**

**AI-Generated Docstrings:**

- **Short and quick.**
- **Easy to read but lacks full detail.**
- **Good for small scripts or personal work.**

**Conclusion :**

**Manual docstrings are more detailed and follow NumPy standards.**
**AI-generated ones are simpler and faster but not as complete.**
**Manual is better for clarity and professional use.**

**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**