|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | |
| **Program Name:** M.Tech. and MCA | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 |
| **Course Coordinator Name** | | | Venkataramana Veeramsetty | | | |
| **Course Code** | |  | **Course Title** | | AI Assisted Problem Solving Using Python | |
| **Year/Sem** | | I/I | **Regulation** | | R24 | |
| **AssignmentNumber:9.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | |
| **ASSIGNMENT – 9**  **Name: Rimsha Mujeeb Roll Number:2503B05138 ( M.Tech - CSE)** | | | | | | |
|  | **Q.No.** | **Question** | | | | |  |  |
|  | ***Q1.***  ***Q2.***  ***Q3.*** | **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.**  **CODE:**      **OUTPUT:**    **DOCUMENTATION:**   * The manually written Google-style docstring is far more detailed and structured compared to the AI-generated version. The manual docstring includes clear sections such as *Args*, *Returns*, *Raises*, and even an example, which makes it more professional and easier for beginners to understand. It explains the input type, outlines errors that may occur, and provides a sample usage to guide the user. * In contrast, the AI-generated docstring is shorter and more generic. It only mentions parameters and a basic return description without additional context or error handling. While the AI version is quick and functional, it lacks depth, clarity, and formatting standards. Overall, the manual docstring reflects stronger documentation practices and is more suitable for educational or professional use, whereas the AI-generated one offers only a minimal explanation.   **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**  **CODE:**      **OUTPUT:**    **DOCUMENTATION:**   * The manually written inline comments provide clear, step-by-step explanations of every part of the code, making it easy for a beginner to understand the logic and purpose of each line. They explain *why* each line exists, not just *what* it does, and they help the reader follow the flow of object creation, method usage, and user input. * The AI-generated comments tend to be very short and superficial. They only restate what the line does rather than giving deeper context, and they do not assist much in understanding the reasoning behind the code. While AI comments are neat and concise, they lack detail and educational value. Therefore, the manual comments are significantly more effective for learning and clarity, whereas the AI-generated comments are suitable only for quick reference   **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:**              **DOCUMENTATION:**   * The manual NumPy-style docstrings follow a strict and professional documentation format. They contain well-defined sections such as *Parameters*, *Returns*, *Raises*, and *Examples*, making the script suitable for real modules and libraries. These docstrings provide the user with everything needed to understand function inputs, outputs, exceptions, and sample usage. * In contrast, the AI-generated docstrings are simple one-line descriptions of what each function does. They lack structure, do not follow the NumPy format, and omit critical information like parameter types, return types, and error conditions. While the AI-generated docstrings are quick to produce, they are not detailed enough for proper module. | | | | |  |  |