|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AssignmentNumber: 9.1**(Present assignment number)/**24**(Total number of assignments)  NAME:- P . VIKAS  ROLLNO:- 2403A510E6 | | | | |
|  | **Q.No.** | **Question** | ***Expected Time***  ***to complete*** |  |
|  | 1 | **Lab 9 – Documentation Generation: Automatic Documentation and Code Comments**  **Lab Objectives**   * To use AI-assisted coding tools for generating Python documentation and code comments. * To apply zero-shot, few-shot, and context-based prompt engineering for documentation creation. * To practice generating and refining docstrings, inline comments, and module-level documentation. * To compare outputs from different prompting styles for quality analysis.   **Task Description #1** (Documentation – Google-Style Docstrings for Python Functions)   * Task: Use AI to add Google-style docstrings to all functions in a given Python script. * Instructions:   + Prompt AI to generate docstrings without providing any input-output examples.   + Ensure each docstring includes:     - Function description     - Parameters with type hints     - Return values with type hints     - Example usage   + Review the generated docstrings for accuracy and formatting. * Expected Output #1:   + A Python script with all functions documented using correctly formatted Google-style docstrings.   **Prompt:**  I have a Python script containing several functions. Please add **Google-style docstrings** to **all functions** in the script.  The docstrings should include the following elements:   * A **brief description** of what the function does. * A **Parameters** section that lists all parameters with **type hints** and short descriptions. * A **Returns** section that lists the **return type** and a brief description. * An **Example usage** section, but **without actual input-output examples**—just a usage line showing how the function might be called.   Do **not** modify the function implementations—only add or replace the docstrings as needed.  CODE:-        **OUTPUT:-**    **OBSERVATIONS:-**    **Task Description #2** (Documentation – Inline Comments for Complex Logic)   * Task: Use AI to add meaningful inline comments to a Python program explaining only complex logic parts. * Instructions:   + Provide a Python script without comments to the AI.   + Instruct AI to skip obvious syntax explanations and focus only on tricky or non-intuitive code sections.   + Verify that comments improve code readability and maintainability. * Expected Output #2:   + Python code with concise, context-aware inline comments for complex logic blocks.     **CODE:-**          **OUTPUT:-**    **OBSERVATIONS:-**    **Task Description #3** (Documentation – Module-Level Documentation)   * Task: Use AI to create a module-level docstring summarizing the * purpose, dependencies, and main functions/classes of a Python file. * Instructions:   + Supply the entire Python file to AI.   + Instruct AI to write a single multi-line docstring at the top of the file.   + Ensure the docstring clearly describes functionality and usage without rewriting the entire code. * Expected Output #3:   + A complete, clear, and concise module-level docstring at the beginning of the file.   **PROMPT:-**  I am providing you with an entire Python file. Please add a \*\*module-level docstring\*\* at the very top of the file.  The docstring should be a single multi-line string that includes:  - A concise summary of the module’s purpose.  - Key dependencies or imports if applicable.  - Main functions or classes included in the module.  - Basic usage notes or how this module might be used (brief, not a full tutorial).  Do \*\*not\*\* rewrite or explain the entire code — just provide a clear and professional summary suitable for the top of a Python file.  Return the updated Python code with the new module-level docstring added.  CODE:-      **OUTPUT:-**    **OBSERVATIONS:-**    **Task Description #4** (Documentation – Convert Comments to Structured Docstrings)   * Task: Use AI to transform existing inline comments into structured function docstrings following Google style. * Instructions:   + Provide AI with Python code containing inline comments.   + Ask AI to move relevant details from comments into function docstrings.   + Verify that the new docstrings keep the meaning intact while improving structure. * Expected Output #4:   + Python code with comments replaced by clear, standardized docstrings.   PROMPT:-  I am providing you with a Python script that contains inline comments inside functions.  Please transform these inline comments into well-structured \*\*Google-style docstrings\*\* for each function, moving all relevant information from the comments into the docstrings.  Make sure the docstrings include:  - A concise function description.  - Parameter descriptions with type hints.  - Return type and description (if applicable).  - Any other important information previously present in the comments.  Remove the inline comments once they are moved to the docstrings.  Return the updated Python code with the new docstrings.  CODE:-      **OUTPUT:-**    **OBSERVATIONS:-**    **Task Description #5** (Documentation – Review and Correct Docstrings)   * Task: Use AI to identify and correct inaccuracies in existing docstrings. * Instructions:   + Provide Python code with outdated or incorrect docstrings.   + Instruct AI to rewrite each docstring to match the current code behavior.   + Ensure corrections follow Google-style formatting. * Expected Output #5:   + Python file with updated, accurate, and standardized docstrings.   **PROMPT:-**  I’m providing a Python script where some functions have outdated or incorrect docstrings.  Please carefully review and \*\*rewrite each docstring\*\* so that it accurately reflects the function’s current behavior.  Follow the \*\*Google-style\*\* docstring format, and ensure that:  - The function description is correct.  - Parameters and return types are accurate and fully described.  - Any removed or changed functionality is no longer referenced.  - The formatting is clean and consistent.  Do not change the function code — only correct the docstrings.  Return the updated Python code.  **CODE:-**      **OUTPUT:-**    **OBSERVATIONS:-**  **Task Description #6** (Documentation – Prompt Comparison Experiment)   * Task: Compare documentation output from a vague prompt and a detailed prompt for the same Python function. * Instructions:   + Create two prompts: one simple (“Add comments to this function”) and one detailed (“Add Google-style docstrings with parameters, return types, and examples”).   + Use AI to process the same Python function with both prompts.   + Analyze and record differences in quality, accuracy, and completeness. * Expected Output #6:   + A comparison table showing the results from both prompts with observations.   **PROMPTS:-**  **🔹 Vague Prompt**  Add comments to this function.  **🔹 Detailed Prompt**  Please add a structured Google-style docstring to the following Python function. Make sure to include:  - A concise description of the function’s purpose  - Parameters with type hints and descriptions  - Return type and its description  - An example usage (without actual output values)  Use proper formatting for Google-style docstrings  **CODE:-**  PYTHON FUNCTION CODE:    CODE FOR VAGUE PROMPT:    CODE FOR DETAILED PROMPT:-    **OUTPUT:-**    **OBSERVATIONS:-** | Week5 - Monday |  |