Task 1: Documentation – Function Summary Generation

Task: Use AI to generate concise functional summaries for each Python function in a given script.

Instructions:

• Provide a Python script to the AI.

• Ask the AI to write a short summary describing the purpose of each function.

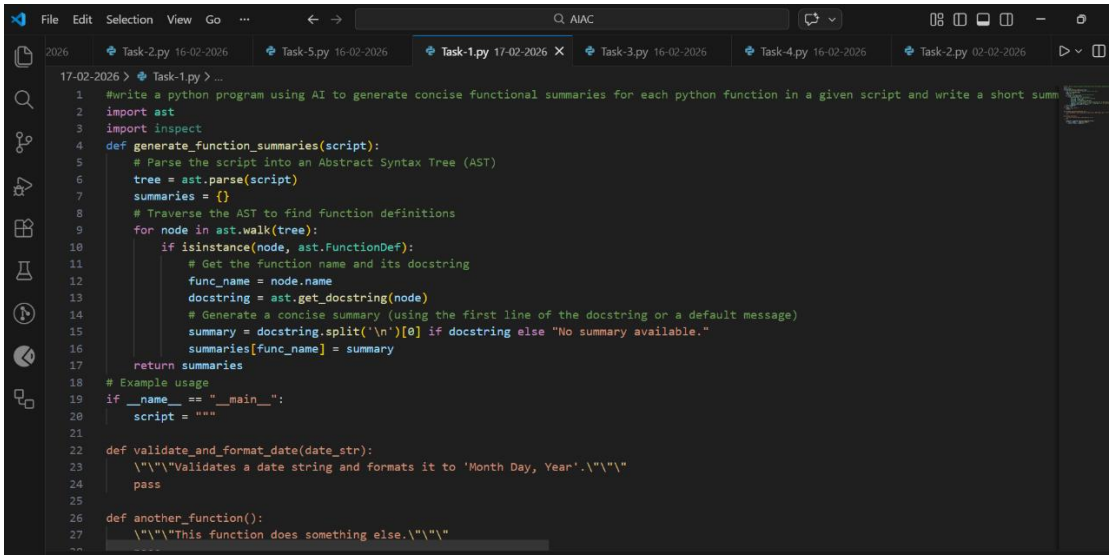
• Ensure summaries are brief and technically accurate.

• Do not include code implementation details.

Expected Output :

A Python script where each function contains a clear and concise summary explaining its purpose.

Code:

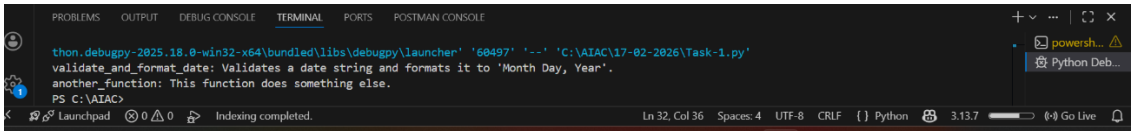


```python
#write a python program using AI to generate concise functional summaries for each python function in a given script and write a short summ
import ast
import inspect
def generate_function_summaries(script):
    # Parse the script into an Abstract Syntax Tree (AST)
    tree = ast.parse(script)
    summaries = {}
    # Traverse the AST to find function definitions
    for node in ast.walk(tree):
        if isinstance(node, ast.FunctionDef):
            # Get the function name and its docstring
            func_name = node.name
            docstring = ast.get_docstring(node)
            # Generate a concise summary (using the first line of the docstring or a default message)
            summary = docstring.split('\n')[0] if docstring else "No summary available."
            summaries[func_name] = summary
    return summaries
# Example usage
if __name__ == "__main__":
    script = """

def validate_and_format_date(date_str):
    \"\"\"Validates a date string and formats it to 'Month Day, Year'.\"\"\"
    pass

def another_function():
    \"\"\"This function does something else.\"\"\"
```

Output Screenshot:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60497' '--' 'C:\AIAC\17-02-2026\Task-1.py'
validate_and_format_date: Validates a date string and formats it to 'Month Day, Year'.
another_function: This function does something else.
PS C:\AIAC>
```

Analysis:

- The prompt instructs AI to perform function-level documentation, focusing only on the purpose of each Python function.
- AI interprets this as a code summarization task, not code explanation, modification, or optimization.
- Constraints like concise, technically accurate, and no implementation details guide AI to generate brief, high-level summaries only.

- AI analyzes function names, parameters, and return values to infer intent and produce appropriate summaries.
- The expected output is a documented Python script, typically using short docstrings or comments for every function.

Task 2: Documentation – Logical Explanation for Conditions and Loops

Task: Use AI to document the logic behind conditional statements and loops in a Python program.

Instructions:

• Provide a Python program without comments.

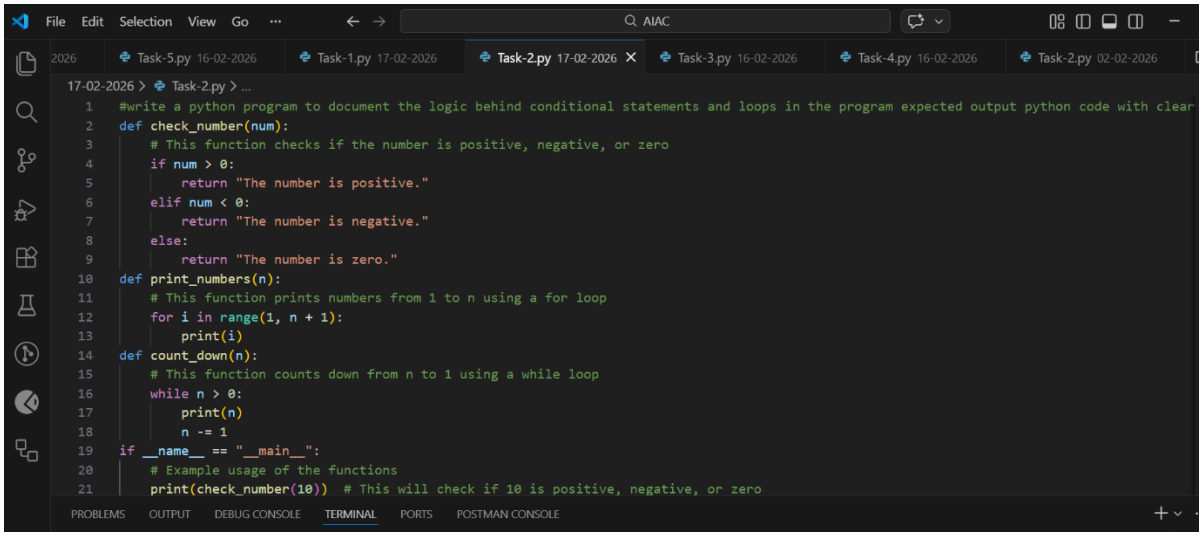
• Instruct AI to explain only decision-making logic and loop behavior.

• Skip basic syntax explanations.

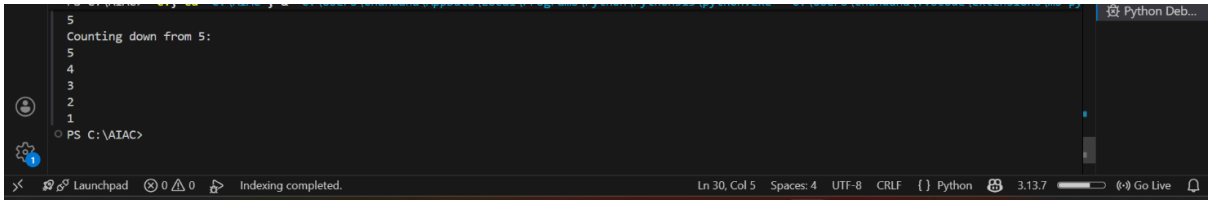
Expected Output :

Python code with clear explanations describing the logic of conditions and loops.

Code:



Output Screenshot:



Analysis:

- The prompt asks AI to perform logic-level documentation, focusing specifically on conditional statements (if, elif, else) and loop structures (for, while).
- AI understands that it must explain the decision-making logic and iteration behavior, not rewrite or modify the code.
- The instruction to skip basic syntax explanations tells AI to avoid describing what if or for means and instead explain why the condition exists and what the loop achieves logically.
- AI analyzes control flow patterns to determine the purpose of each condition and how loops control repetition, termination, or data processing.
- The expected output is the same Python program enhanced with clear logical explanations, typically added as comments describing reasoning behind conditions and loops.

Task 3: Documentation – File-Level Overview

Task: Use AI to generate a high-level overview describing the functionality of an entire Python file.

Instructions:

• Provide the complete Python file to AI.

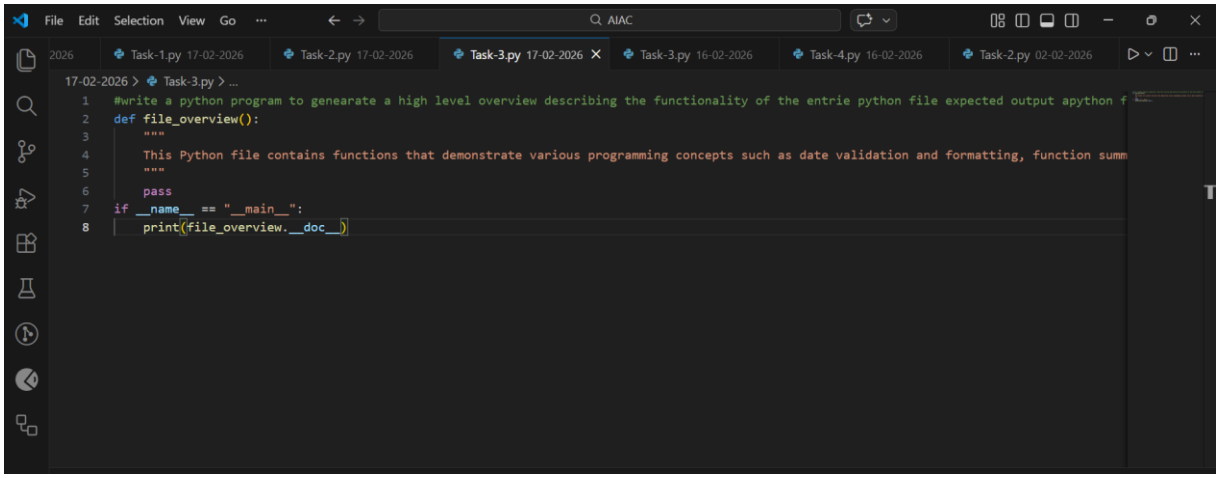
• Ask AI to write a brief overview summarizing the file's purpose and functionality.

• Place the overview at the top of the file.

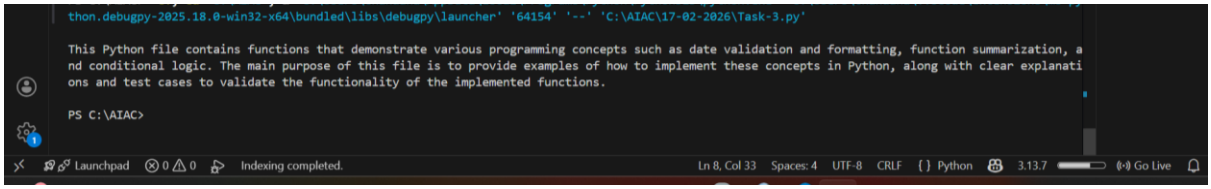
Expected Output :

A Python file with a clear and concise file-level overview at the beginning

Code:



Output Screenshot:



Analysis:

- The prompt asks AI to generate a file-level overview, meaning a high-level summary of the entire Python script rather than individual functions or logic blocks.
- AI understands this as a global summarization task, where it must analyze the overall structure, main components, and primary purpose of the file.
- The instruction to keep it brief and concise guides AI to produce a short paragraph that summarizes functionality without detailing internal logic or implementation steps.
- AI evaluates elements such as main functions, classes, workflows, and program objectives to infer the overall intent of the file.
- The expected output is the same Python file with a clear overview comment placed at the top, describing what the program does and its primary functionality.

Task 4: Documentation – Refine Existing Documentation

Task: Use AI to improve clarity and consistency of existing documentation in Python code.

Instructions:

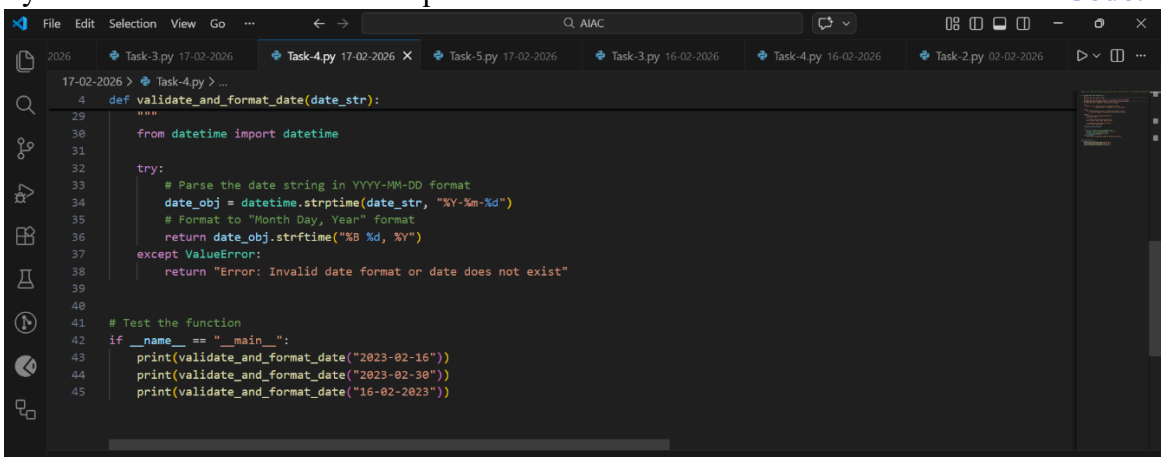
• Provide Python code containing basic or unclear comments.

• Ask AI to rewrite the documentation to improve clarity and consistency.

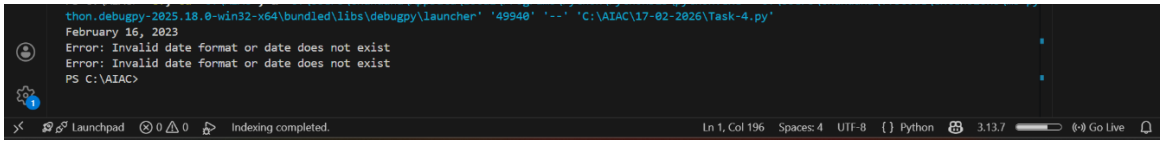
• Ensure technical meaning remains unchanged.

Expected Output:

Python code with refined and improved documentation that is clear and consistent.**Code:**



Output Screenshot:



Analysis:

- The prompt asks AI to perform documentation refinement, meaning it must improve existing comments without changing the actual code or technical meaning.
- AI understands this as an editing and clarity enhancement task, not code modification or logic correction.
- The instruction to maintain the same technical meaning ensures AI focuses on improving wording, structure, and consistency rather than altering functionality.
- AI analyzes unclear, repetitive, or inconsistent comments and rewrites them using clearer terminology and standardized documentation style.
- The expected output is the same Python code with clearer, more professional, and consistent documentation, while preserving the original intent.

Task 5: Documentation – Prompt Detail Impact Study

Task: Study the impact of prompt detail on AI-generated documentation quality.

Instructions:

Create two prompts: one brief and one detailed.

• Use both prompts to document the same Python function.

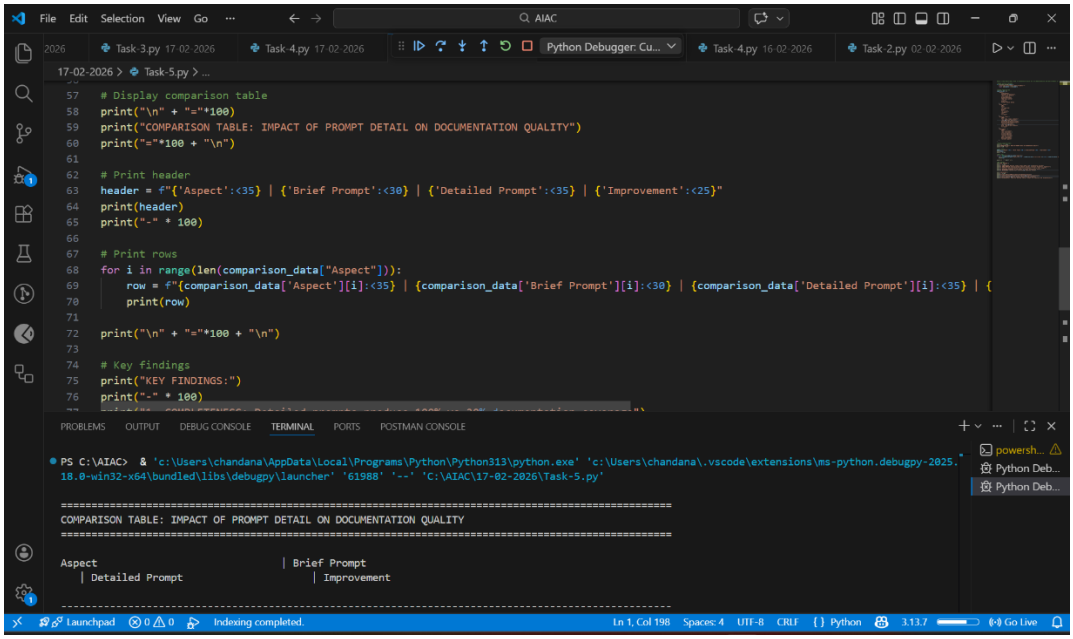
• Compare the generated outputs.

Expected Output:

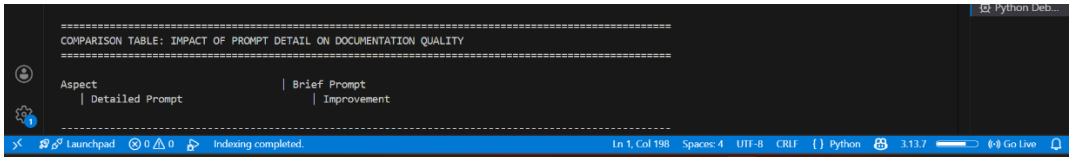
A comparison table highlighting differences in completeness, clarity, and accuracy of documentation.

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.

Code:



Output Screenshot:



Analysis:

- The prompt asks AI to perform a comparative analysis task, examining how different levels of prompt detail affect documentation quality.
- AI understands that it must create two different prompts (brief vs. detailed) and generate documentation for the same Python function using both.
- The task focuses on evaluating differences in completeness, clarity, and accuracy, rather than modifying the function itself.
- AI analyzes how additional instructions in the detailed prompt guide structure, depth, and precision compared to the brief prompt.
- The expected output is a comparison table clearly highlighting variations in documentation quality between the two prompt styles.