

Task 1: Documentation – Function Summary Generation

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

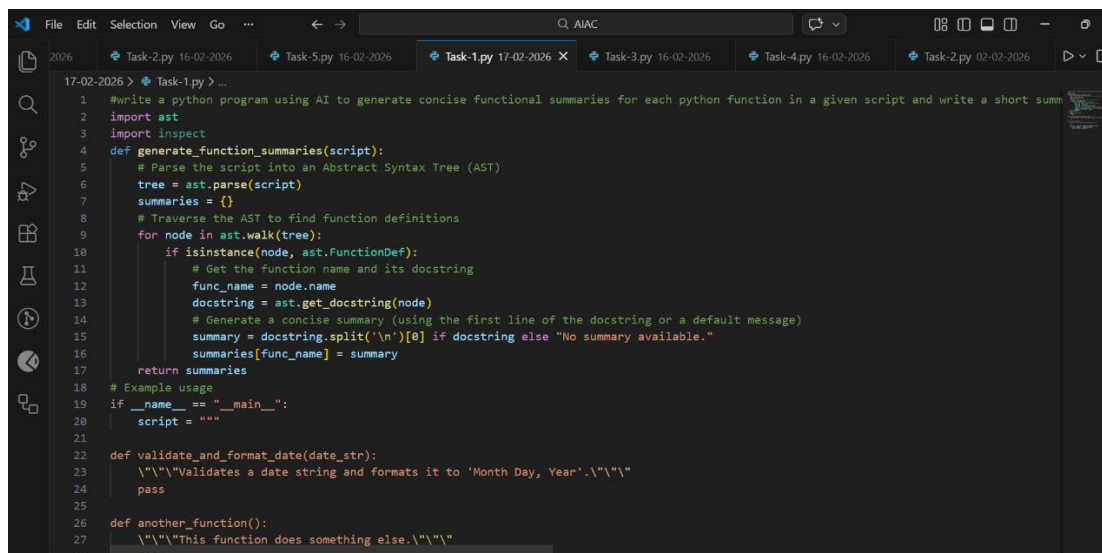
Instructions:

- 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 :

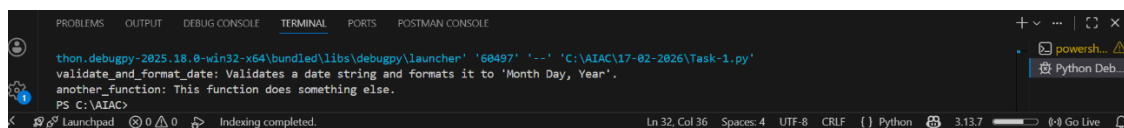
A Python script where each function contains a clear and concise summary explaining its purpose.

Code:



```
17-02-2026 > Task-1.py > ...
1 #write a python program using AI to generate concise functional summaries for each python function in a given script and write a short summ
2 import ast
3 import inspect
4 def generate_function_summaries(script):
5     # Parse the script into an Abstract Syntax Tree (AST)
6     tree = ast.parse(script)
7     summaries = {}
8     # Traverse the AST to find function definitions
9     for node in ast.walk(tree):
10         if isinstance(node, ast.FunctionDef):
11             # Get the function name and its docstring
12             func_name = node.name
13             docstring = ast.get_docstring(node)
14             # Generate a concise summary (using the first line of the docstring or a default message)
15             summary = docstring.split('\n')[0] if docstring else "No summary available."
16             summaries[func_name] = summary
17     return summaries
18 # Example usage
19 if __name__ == "__main__":
20     script = """
21
22 def validate_and_format_date(date_str):
23     \"\"\"Validates a date string and formats it to 'Month Day, Year'.\"\"\"
24     pass
25
26 def another_function():
27     \"\"\"This function does something else.\"\"\"
28 """
```

Output Screenshot:



```
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.

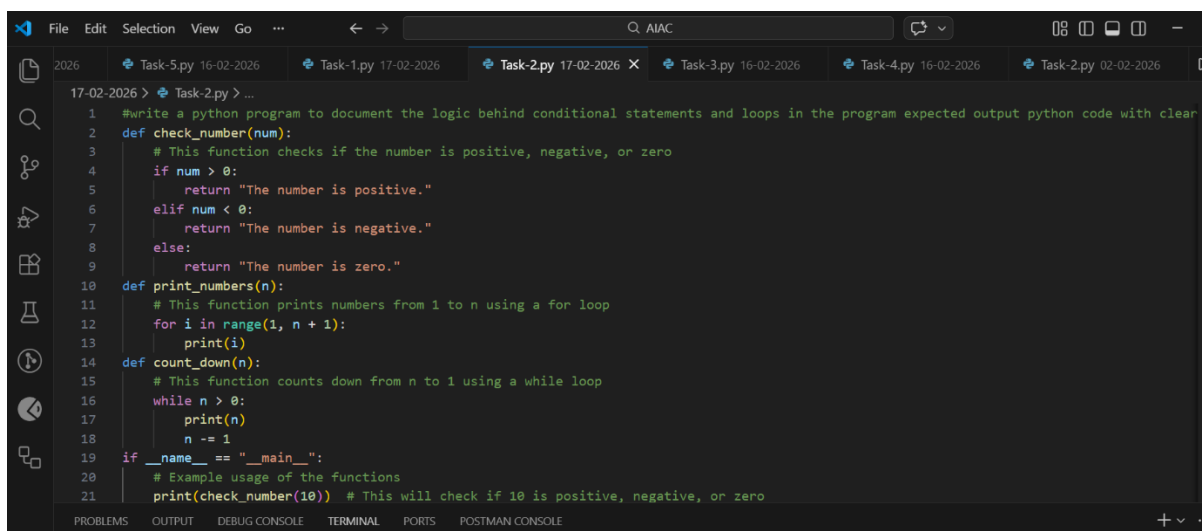
Instructions:

- 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.

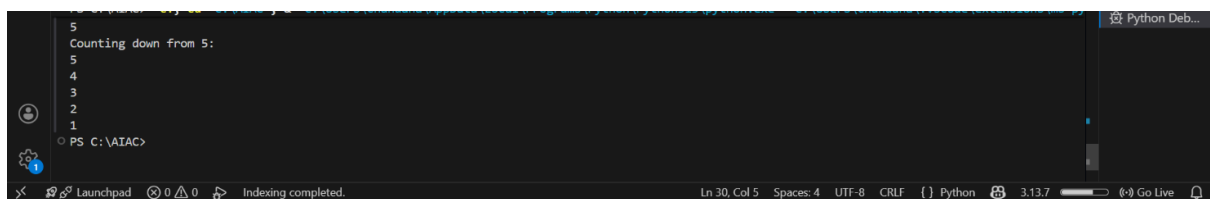
Code:



```

17-02-2026 > Task-2.py > ...
1 #write a python program to document the logic behind conditional statements and loops in the program expected output python code with clear
2 def check_number(num):
3     # This function checks if the number is positive, negative, or zero
4     if num > 0:
5         return "The number is positive."
6     elif num < 0:
7         return "The number is negative."
8     else:
9         return "The number is zero."
10
11 def print_numbers(n):
12     # This function prints numbers from 1 to n using a for loop
13     for i in range(1, n + 1):
14         print(i)
15
16 def count_down(n):
17     # This function counts down from n to 1 using a while loop
18     while n > 0:
19         print(n)
20         n -= 1
21
22 if __name__ == "__main__":
23     # Example usage of the functions
24     print(check_number(10)) # This will check if 10 is positive, negative, or zero
  
```

Output Screenshot:



```

5 Counting down from 5:
5
4
3
2
1
PS C:\AIAC>
  
```

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.

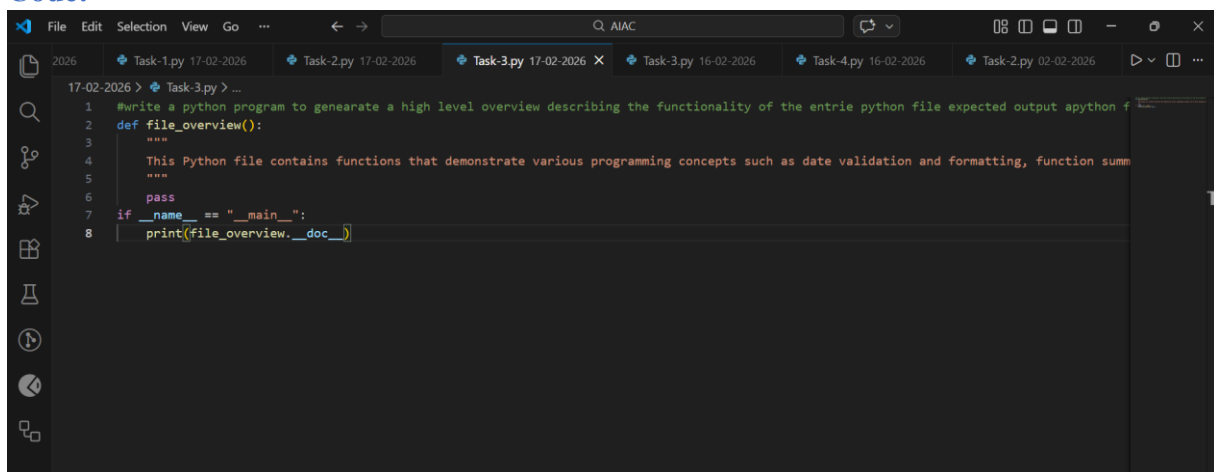
Instructions:

- 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:

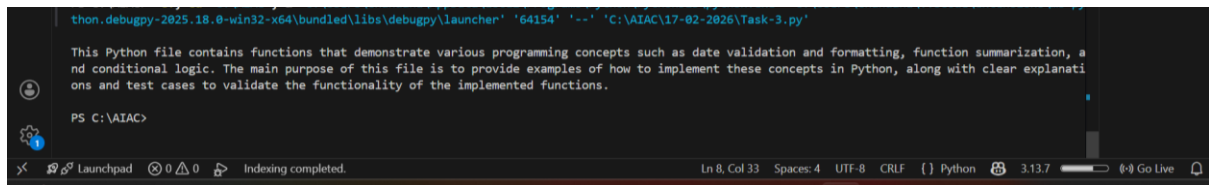


```

1  #write a python program to generate a high level overview describing the functionality of the entire python file expected output a python f
2  def file_overview():
3      """
4      This Python file contains functions that demonstrate various programming concepts such as date validation and formatting, function summ
5      """
6      pass
7  if __name__ == "__main__":
8      print(file_overview.__doc__)

```

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

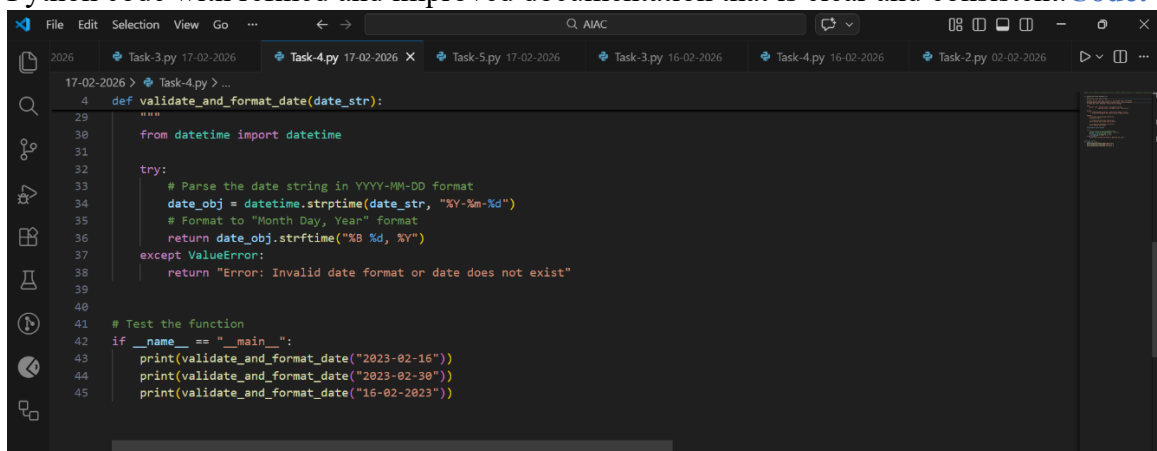
Task: Use AI to improve clarity and consistency of existing documentation in Python code.

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:

A screenshot of a Python IDE window. The title bar reads 'thon.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '49940' '--' 'C:\AIAC\17-02-2026\Task-4.py'. The main text area shows the date 'February 16, 2023' followed by two error messages: 'Error: Invalid date format or date does not exist' and 'Error: Invalid date format or date does not exist'. Below the errors is a command prompt prompt 'PS C:\AIAC>'. The status bar at the bottom shows 'Ln 1, Col 196', 'Spaces: 4', 'UTF-8', 'Python', '3.13.7', and a 'Go Live' button.

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:

The screenshot shows a VS Code editor with a Python file named 'Task-5.py'. The script is designed to generate a comparison table for documentation quality based on different prompt details. The code includes comments and uses f-strings for formatting. The terminal output shows the execution of the script, which prints a header, a table of comparisons, and key findings.

```
17-02-2026 > Task-5.py > ...
57 # Display comparison table
58 print("\n" + "="*100)
59 print("COMPARISON TABLE: IMPACT OF PROMPT DETAIL ON DOCUMENTATION QUALITY")
60 print("="*100 + "\n")
61
62 # Print header
63 header = f"{'Aspect':<35} | {'Brief Prompt':<30} | {'Detailed Prompt':<35} | {'Improvement':<25}"
64 print(header)
65 print("-" * 100)
66
67 # Print rows
68 for i in range(len(comparison_data["Aspect"])):
69     row = f"{comparison_data['Aspect'][i]:<35} | {comparison_data['Brief Prompt'][i]:<30} | {comparison_data['Detailed Prompt'][i]:<35} | {
70     print(row)
71
72 print("\n" + "="*100 + "\n")
73
74 # Key findings
75 print("KEY FINDINGS:")
76 print("-" * 100)
```

The terminal output shows the following table:

Aspect	Brief Prompt	Detailed Prompt	Improvement
Detailed Prompt			

Output Screenshot:

The screenshot shows the terminal output of the Python script. It displays a comparison table with the following structure:

Aspect	Brief Prompt	Detailed Prompt	Improvement
Detailed Prompt			

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.