

Assignment – 9.2

Name – M. Samgeetha

Hall Ticket Number – 2303A52088

Batch - 40

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 – 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 -1:

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

Prompts:

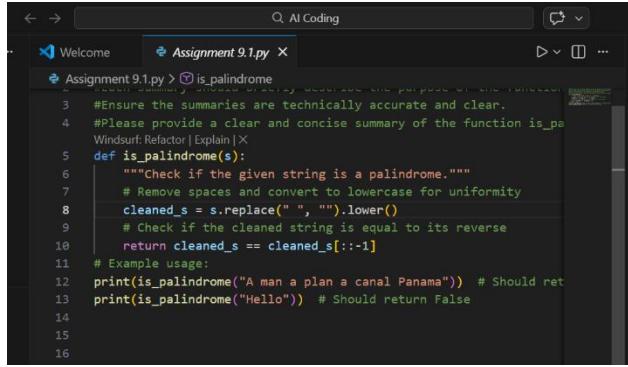
#Please read the following Python script and generate concise docstrings for each function.

#Each summary should briefly describe the purpose of the function, without including implementation details.

#Ensure the summaries are technically accurate and clear.

#Please provide a clear and concise summary of the function is_palindrome that indicates its purpose and functionality.

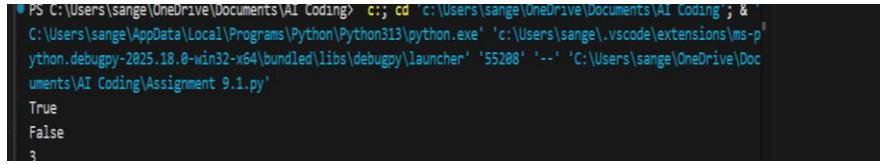
Code:



```
Assignment 9.1.py
```

```
1 #Please read the following Python script and generate concise docstrings for each function.
2 #Each summary should briefly describe the purpose of the function, without including
3 #implementation details.
4 #Ensure the summaries are technically accurate and clear.
5 #Please provide a clear and concise summary of the function is_palindrome that indicates its
6 #purpose and functionality.
7
8 def is_palindrome(s):
9     """Check if the given string is a palindrome."""
10    # Remove spaces and convert to lowercase for uniformity
11    cleaned_s = s.replace(" ", "").lower()
12    # Check if the cleaned string is equal to its reverse
13    return cleaned_s == cleaned_s[::-1]
14
15
16
```

Output:



```
PS C:\Users\sange\OneDrive\Documents\AI Coding> cd "c:/Users/sange/OneDrive/Documents/AI Coding"; & 'C:/Users/sange/AppData/Local/Programs/Python/313/python.exe' 'c:/Users/sange/.vscode/extensions/ms-python.python.debugpy-2025.18.0-win32-x64/bundled/libs/debugpy/launcher' '55208' --- 'C:/Users/sange/OneDrive/Documents/AI Coding/Assignment 9.1.py'
True
False
3
```

Analysis:

Step 1 – Input the Script

Provide the Python script containing multiple functions to the AI tool.

Step 2 – Extract Each Function

Identify each function's name, parameters, and purpose by reading or parsing the code.

Step 3 – Generate Summaries

Use AI prompting (zero-shot, few-shot, or context-based) to produce a **one-sentence summary** describing each function's purpose — no implementation details.

Step 4 – Insert Documentation

Add the generated summaries as **docstrings** inside each function or compile them into a documentation file.

Step 5 – Review and Refine

Check summaries for **accuracy, brevity, and consistency**, and edit if needed for clarity and correctness.

Task Description -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 -2:

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

Prompts:

#You are a Python documentation assistant. Explain only logic for conditions and loops, no syntax details.

#Example 1:

#Code:

#if x > 0:

y = x * 2

#Explanation:

#"This checks if x is positive. If true, it doubles x and stores in y."

#Example 2:

#Code:

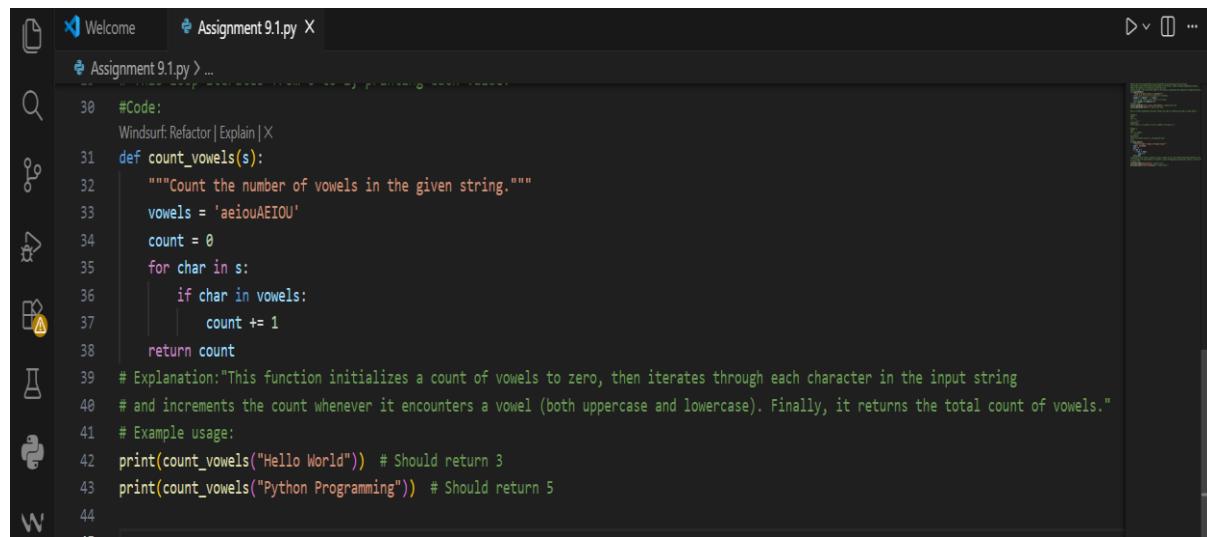
```
#for i in range(3):
```

```
    # print(i)
```

#Explanation:

#"This loop iterates from 0 to 2, printing each value."

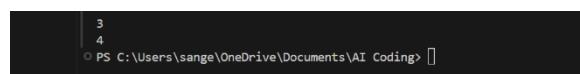
Code:



The screenshot shows a code editor window titled "Assignment 9.1.py X". The code is as follows:

```
30 #Code:
31     Windsurf Refactor | Explain | X
32 def count_vowels(s):
33     """Count the number of vowels in the given string."""
34     vowels = 'aeiouAEIOU'
35     count = 0
36     for char in s:
37         if char in vowels:
38             count += 1
39     # Explanation:"This function initializes a count of vowels to zero, then iterates through each character in the input string
40     # and increments the count whenever it encounters a vowel (both uppercase and lowercase). Finally, it returns the total count of vowels."
41     # Example usage:
42     print(count_vowels("Hello World")) # Should return 3
43     print(count_vowels("Python Programming")) # Should return 5
44
```

Output:



```
3
4
PS C:\Users\sange\OneDrive\Documents\AI Coding> []
```

Analysis:

Step 1 – Provide the Code

Give the AI the Python program **without comments** that you want documented.

Step 2 – Define the Role

Tell the AI:

"You are a Python documentation assistant."

This focuses it on **producing explanations**, not fixing or rewriting code.

Step 3 – Limit the Scope

Instruct the AI to explain **only the logic of if/elif/else statements and loops**, skipping syntax or basic Python keywords.

Step 4 – Give Examples (Optional)

Provide **few-shot examples** showing how a condition or loop should be explained in one concise sentence.

This helps AI **match style and clarity**.

Step 5 – Generate and Review

Ask the AI to analyze your code, produce explanations, and then **review them for accuracy, conciseness, and clarity** before adding as comments or a separate documentation file.

Task Description -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 -3:

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

Code:

```

Assignment 9.1.py > ...
15 #You are a Python documentation assistant. Explain only logic for conditions and loops, no syntax details.
16 #Example 1:
17 #Code:
18 #if x > 0:
19 #    y = x * 2
20 #Explanation:
21 #This checks if x is positive. If true, it doubles x and stores in y."
22 #Example 2:
23 #Code:
24 #for i in range(3):
25 #    print(i)
26 #Explanation:
27 #This loop iterates from 0 to 2, printing each value."
28 #Code:
29 Windsurf: Refactor | Explain | X
30 def count_vowels(s):
31     """(variable) vowels: Literal['aeiouAEIOU'] | string."""
32     vowels = 'aeiouAEIOU'
33     count = 0
34     for char in s:
35         if char in vowels:
36             count += 1
37     return count
38 # Explanation: This function initializes a count of vowels to zero, then iterates through each character in the input string
39 # and increments the count whenever it encounters a vowel (both uppercase and lowercase). Finally, it returns the total count of vowels.
40 # Example usage:
41 print(count_vowels("Hello World")) # Should return 3
42 print(count_vowels("Python Programming")) # Should return 5
43 #Act as a software architect. Analyze the following Python file and write a high-level module overview (3-4 sentences) that describes the
44 """
45 This script provides functions to check if a string is a palindrome and count the number of vowels in a given string.
46 """
47 Windsurf: Refactor | Explain | X
48 def is_palindrome(s):
49

```

Ln 32, Col 14 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit) ⓘ Go Live Windsurf (...) □

```

Assignment 9.1.py > ...
40     print(count_vowels("Hello World")) # Should return 3
41     print(count_vowels("Python Programming")) # Should return 5
42     #Act as a software architect. Analyze the following Python file and write a high-level module overview (3-4 sentences) that describes the
43     """
44     This script provides functions to check if a string is a palindrome and count the number of vowels in a given string.
45     """
46     Windsurf: Refactor | Explain | X
47     def is_palindrome(s):
48         """Check if the given string is a palindrome."""
49         cleaned_s = s.replace(" ", "").lower()
50         return cleaned_s == cleaned_s[::-1]
51     Windsurf: Refactor | Explain | X
52     def count_vowels(s):
53         """Count the number of vowels in the given string."""
54         vowels = 'aeiouAEIOU'
55         count = 0
56         for char in s:
57             if char in vowels:
58                 count += 1
59     # Example usage:
60     print(is_palindrome("A man a plan a canal Panama")) # Should return True
61     print(is_palindrome("Hello")) # Should return False
62     print(count_vowels("Hello World")) # Should return 3
63     print(count_vowels("Python")) # Should return 5
64

```

Output:

```

PS C:\Users\sange\OneDrive\Documents\AI Coding> cd 'c:\Users\sange\OneDrive\Documents\AI Coding'; &
C:\Users\sange\AppData\Local\Programs\Python\Python313\python.exe 'c:\Users\sange\.vscode\extensions\ms-python.python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53506' '--' 'C:\Users\sange\OneDrive\Documents\AI Coding\Assignment 9.1.py'
● True
False
3
4
True
False
3
1
○ PS C:\Users\sange\OneDrive\Documents\AI Coding> []

```

Ln 61, Col 28 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit) ⓘ Go Live

Analysis:

Role Constraint: "Software architect" shifts the AI from describing syntax to describing system intent.

Structural Requirement: Placing it "before imports" ensures PEP 257 compliance for module-level documentation.

Synthesis Instruction: "Do not list functions" prevents the AI from being redundant and forces high-level abstraction.

Brevity Control: The "3-sentence" limit ensures the overview remains a summary, not a manual.

Output Format: Requesting the "complete script" makes the result immediately deployable.

Task Description -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 -4:

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

Prompts:

#Give the AI a Python file containing **existing comments or docstrings**, even if they are basic, unclear, or inconsistent.

#Example snippet:

```
#def compute_average(values):  
    # computes avg  
    #return sum(values)/len(values)
```

Analysis:

- Input:** Provide Python code with existing comments or docstrings, even if unclear.
- Role:** Assign AI as a documentation assistant focused on clarity and consistency.
- Scope:** Instruct AI to rewrite comments/docstrings without changing technical meaning.
- Examples (Optional):** Show a few before-and-after comment samples to guide style.
- Output:** Generate the Python file with refined, clear, and consistent documentation ready for review.

Task Description -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 -5:

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

Prompts:

```
#“You are a Python documentation assistant. Write a short, concise docstring for the following function explaining its purpose.”
```

```
#“You are a Python documentation assistant. Write a clear, detailed docstring for the following function. Include: function purpose, input parameters with types, return value with type, and any edge cases. Keep it accurate, concise, and professional.”
```

```
#“After generating outputs using both prompts, create a table comparing them on completeness, clarity, and accuracy. Highlight differences clearly.”
```

```
#“Use one brief prompt and one detailed prompt to document the same Python function. Compare the AI outputs and summarize findings in a table format focusing on completeness, clarity, and accuracy.”
```

```
#“Analyze the two docstrings generated for the same function. Score each on completeness (0–3), clarity (0–3), and accuracy (0–3). Provide a short explanation for each score to highlight differences.”
```

Code:

```
Assignment 9.1.py > ...
91  #"You are a Python documentation assistant. Write a clear, detailed docstring for the following function.
92  #"After generating outputs using both prompts, create a table comparing them on completeness, clarity,
93  #"Use one brief prompt and one detailed prompt to document the same Python function. Compare the AI's
94  #"Analyze the two docstrings generated for the same function. Score each on completeness (0-3), clarity
95  Windsurf: Refactor | Explain | X
96  def factorial(n):
97      """Calculate the factorial of a non-negative integer n."""
98      if n < 0:
99          raise ValueError("Factorial is undefined for negative numbers.")
100     result = 1
101     for i in range(1, n + 1):
102         result *= i
103     return result
104 # Example usage:
105 print(factorial(5)) # Should return 120
106 print(factorial(0)) # Should return 1
107 print(factorial(-1)) # Should raise ValueError
108
109
```

Output:

```
3.0
0
120
1
```

Analysis:

Step 1 – Select a Python Function

Choose a representative Python function to document. Make sure it has clear inputs, outputs, and some logic so differences in prompt detail will show.

Step 2 – Create Two Prompts

- **Brief Prompt:** Ask AI for a short, concise docstring describing only the function's purpose.
- **Detailed Prompt:** Ask AI for a full docstring including purpose, inputs with types, return values, and edge cases.

Step 3 – Generate Documentation

Use both prompts on the same function to get two separate AI-generated docstrings.

Step 4 – Compare Outputs

Evaluate the two outputs on:

1. **Completeness:** Are purpose, inputs, outputs, and edge cases covered?
 2. **Clarity:** Is the documentation easy to understand?
 3. **Accuracy:** Does it preserve technical meaning?
-

Step 5 – Summarize Findings

Create a table or short report showing differences between brief vs detailed prompts, highlighting which prompt produces more complete, clear, and accurate documentation.