

Name:P Pranay Kumar H.No:2303A51829

Batch:26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name:B. Tech		Assignment Type: Lab	Academic Year:2025-2026
Course Coordinator Name		Dr. Rishabh Mittal	
Instructor(s)Name		Mr. S Naresh Kumar	
		Ms. B. Swathi	
		Dr. Sasanko Shekhar Gantayat	
		Mr. Md Sallauddin	
		Dr. Mathivanan	
		Mr. Y Srikanth	
		Ms. N Shilpa	
		Dr. Rishabh Mittal (Coordinator)	
		Dr. R. Prashant Kumar	
		Mr. Ankushavali MD	
		Mr. B Viswanath	
		Ms. Sujitha Reddy	
		Ms. A. Anitha	
		Ms. M.Madhuri	
		Ms. Katherashala Swetha	
Ms. Velpula sumalatha			
Mr. Bingi Raju			
Mr. G. Kranthi			
Course Code		23CS002PC304	
Year/Sem		III/I	
Date and Day of Assignment		Week 5 - Thursday	
Duration		2 Hours	
AssignmentNumber:9.4 (Present assignment number)/24(Total number of assignments)			
Q.No.	Question		Expected Time to complete

1	<p>Lab 9 – Documentation Generation: Automatic Documentation and Code Comments</p> <p>Lab Objectives</p> <ul style="list-style-type: none"> • 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. <p>Lab Outcomes</p> <ul style="list-style-type: none"> • Generate structured code documentation using AI tools • Apply appropriate documentation styles to different code contexts • Improve code readability through selective commenting • Convert informal developer comments into professional documentation • Analyze and refine AI-generated documentation 	Week 5
	<p>Task 1: Auto-Generating Function Documentation in a Shared Codebase</p> <p>Scenario</p> <p>You have joined a development team where several utility functions are already implemented, but the code lacks proper documentation. New team members are struggling to understand how these functions should be used.</p> <p>Task Description</p> <p>You are given a Python script containing multiple functions without any docstrings.</p> <p>Using an AI-assisted coding tool:</p> <ul style="list-style-type: none"> • Ask the AI to automatically generate Google-style function docstrings for each function • Each docstring should include: <ul style="list-style-type: none"> ◦ A brief description of the function ◦ Parameters with data types ◦ Return values ◦ At least one example usage (if applicable) <p>Experiment with different prompting styles (zero-shot or context-based) to observe quality differences.</p> <p>Expected Outcome</p>	

- A Python script with well-structured Google-style docstrings
- Docstrings that clearly explain function behavior and usage
- Improved readability and usability of the codebase

```

AAC_A_94.py
AAC_A_94.py > ...
1 import math
2 import random
3 import string
4
5 def calculate_circle_area(radius):
6     if radius < 0:
7         return 0
8     return math.pi * radius * radius
9
10 def fahrenheit_to_celsius(fahrenheit):
11     return (fahrenheit - 32) * 5.0 / 9.0
12
13 def check_palindrome(text):
14     clean_text = ''.join(c.lower() for c in text if c.isalnum())
15     return clean_text == clean_text[::-1]
16
17 def fibonacci_iterative(n):
18     if n <= 0:
19         return []
20     elif n == 1:
21         return [0]
22     sequence = [0, 1]
23     while len(sequence) < n:
24         sequence.append(sequence[-1] + sequence[-2])
25     return sequence
26
27 def generate_random_password(length):
28     if length < 8:
29         length = 8
30     chars = string.ascii_letters + string.digits + string.punctuation
31     return ''.join(random.choice(chars) for _ in range(length))
32

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\shash\rful-crud28> c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55512' ''
Circle Area (r=5): 78.53981633974483
Random Password (len 12): 6s|37[X5i;%f
● PS C:\Users\shash\rful-crud28> c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55547' ''
Circle Area (r=5): 78.53981633974483
Fahrenheit to Celsius (32F): 0.0
Is 'A man, a plan, a canal: Panama' a palindrome?: True
Fibonacci (5): [0, 1, 1, 2, 3]
Random Password (len 12): D\Wu]Cnx*RI'
○ PS C:\Users\shash\rful-crud28>

```

-
-

Task 2: Enhancing Readability Through AI-Generated Inline Comments

Scenario

A Python program contains complex logic that works correctly but is difficult to understand at first glance. Future maintainers may find it hard to debug or extend this code.

Task Description

You are provided with a Python script containing:

- Loops

- Conditional logic
- Algorithms (such as Fibonacci sequence, sorting, or searching)

Use AI assistance to:

- Automatically insert **inline comments only for complex or non-obvious logic**
- Avoid commenting on trivial or self-explanatory syntax

The goal is to improve clarity without cluttering the code.

Expected Outcome

- A Python script with concise, meaningful inline comments
- Comments that explain *why* the logic exists, not *what* Python syntax does
- Noticeable improvement in code readability

```

❶ AAC_A_9.4.py X
❷ AAC_A_9.4.py > ...
1 import math
2 import random
3 import string
4
5 def calculate_circle_area(radius):
6     if radius < 0:
7         return 0
8     return math.pi * radius * radius
9
10 def fahrenheit_to_celsius(fahrenheit):
11     return (fahrenheit - 32) * 5.0 / 9.0
12
13 def check_palindrome(text):
14     clean_text = ''.join(c.lower() for c in text if c.isalnum())
15     return clean_text == clean_text[::-1]
16
17 def fibonacci_iterative(n):
18     if n <= 0:
19         return []
20     elif n == 1:
21         return [0]
22     sequence = [0, 1]
23     while len(sequence) < n:
24         sequence.append(sequence[-1] + sequence[-2])
25     return sequence
26
27 def generate_random_password(length):
28     if length < 8:
29         length = 8
30     chars = string.ascii_letters + string.digits + string.punctuation
31     return ''.join(random.choice(chars) for _ in range(length))
32

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\shash\rful-crud28> c;; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\rnions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55512'
Circle Area (r=5): 78.53981633974483
Random Password (len 12): 6s[37]X5i;%f
❸ PS C:\Users\shash\rful-crud28> c;; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\rnions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55547'
Circle Area (r=5): 78.53981633974483
Fahrenheit to Celsius (32F): 0.0
Is 'A man, a plan, a canal: Panama' a palindrome?: True
Fibonacci (5): [0, 1, 1, 2, 3]
Random Password (len 12): DWuJnx*RI'
❹ PS C:\Users\shash\rful-crud28>

```

```

AAC A 9.4.py X
❶ AAC A 9.4.py > ...
11     |     return (fahrenheit - 32) * 5.0 / 9.0
12
13 def check_palindrome(text):
14     clean_text = ''.join(c.lower() for c in text if c.isalnum())
15     return clean_text == clean_text[::-1]
16
17 def fibonacci_iterative(n):
18     if n <= 0:
19         return []
20     elif n == 1:
21         return [0]
22     sequence = [0, 1]
23     while len(sequence) < n:
24         sequence.append(sequence[-1] + sequence[-2])
25     return sequence
26
27 def generate_random_password(length):
28     if length < 8:
29         length = 8
30     chars = string.ascii_letters + string.digits + string.punctuation
31     return ''.join(random.choice(chars) for _ in range(length))
32
33 if __name__ == "__main__":
34     print(f"Circle Area (r=5): {calculate_circle_area(5)}")
35     print(f"Fahrenheit to Celsius (32F): {fahrenheit_to_celsius(32)}")
36     print(f"Is 'A man, a plan, a canal: Panama' a palindrome?: {check_palindrome('A man, a plan, a canal: Panama')}")
37     print(f"Fibonacci (5): {fibonacci_iterative(5)}")
38     print(f"Random Password (len 12): {generate_random_password(12)}")
39

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\shash\rful-crud28> cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\ShashiDhar\python.exe' 'c:\Users\shash\ms-python.debugger-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55512' '--' 'c:\Users\shash\rful-crud28\AAC A 9.4.py'
Circle Area (r=5): 78.53981633974483
Random Password (len 12): 6s|37[X51;%f
PS C:\Users\shash\rful-crud28> cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\ShashiDhar\python.exe' 'c:\Users\shash\ms-python.debugger-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55547' '--' 'c:\Users\shash\rful-crud28\AAC A 9.4.py'
Fahrenheit to Celsius (32F): 0.0
Is 'A man, a plan, a canal: Panama' a palindrome?: True
Fibonacci (5): [0, 1, 1, 2, 3]
Random Password (len 12): 0wU]Cnx*RI'

```

Task 3: Generating Module-Level Documentation for a Python Package

Scenario

Your team is preparing a Python module to be shared internally (or uploaded to a repository). Anyone opening the file should immediately understand its purpose and structure.

Task Description

Provide a complete Python module to an AI tool and instruct it to automatically generate a **module-level docstring** at the top of the file that includes:

- The purpose of the module
- Required libraries or dependencies
- A brief description of key functions and classes
- A short example of how the module can be used

Focus on clarity and professional tone.

Expected Outcome

- A well-written multi-line module-level docstring
- Clear overview of what the module does and how to use it
- Documentation suitable for real-world projects or repositories

The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are two tabs: 'AAC A 9.4.py X' and 'AAC A 9.4.py > ...'. The main area contains the following Python code:

```
1 import math
2 import random
3 import string
4
5 def calculate_circle_area(radius):
6     if radius < 0:
7         return 0
8     return math.pi * radius * radius
9
10 def fahrenheit_to_celsius(fahrenheit):
11     return (fahrenheit - 32) * 5.0 / 9.0
12
13 def check_palindrome(text):
14     clean_text = ''.join(c.lower() for c in text if c.isalnum())
15     return clean_text == clean_text[::-1]
16
17 def fibonacci_iterative(n):
18     if n <= 0:
19         return []
20     elif n == 1:
21         return [0]
22     sequence = [0, 1]
23     while len(sequence) < n:
24         sequence.append(sequence[-1] + sequence[-2])
25     return sequence
26
27 def generate_random_password(length):
28     if length < 8:
29         length = 8
30     chars = string.ascii_letters + string.digits + string.punctuation
31     return ''.join(random.choice(chars) for _ in range(length))
```

Below the code, there are several tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The TERMINAL tab displays the following command-line output:

```
PS C:\Users\shash\rful-crud28> c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\miniconda3\envs\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '5165'
Circle Area (r=5): 78.53981633974483
Random Password (len 12): |fs^QAG1sj7W
● PS C:\Users\shash\rful-crud28> c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\miniconda3\envs\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '5168'
Circle Area (r=5): 78.53981633974483
Fahrenheit to Celsius (32F): 0.0
Is 'A man, a plan, a canal: Panama' a palindrome?: True
Fibonacci (5): [0, 1, 1, 2, 3]
Random Password (len 12): +'KO*,A{IR}N
○ PS C:\Users\shash\rful-crud28>
```

```

AAC A 9.4.py X
AAC A 9.4.py > ...

14 def check_palindrome(text):
15     clean_text = ''.join(c.lower() for c in text if c.isalnum())
16     return clean_text == clean_text[::-1]
17
18 def fibonacci_iterative(n):
19     if n <= 0:
20         return []
21     elif n == 1:
22         return [0]
23     sequence = [0, 1]
24     while len(sequence) < n:
25         sequence.append(sequence[-1] + sequence[-2])
26     return sequence
27
28 def generate_random_password(length):
29     if length < 8:
30         length = 8
31     chars = string.ascii_letters + string.digits + string.punctuation
32     return ''.join(random.choice(chars) for _ in range(length))
33
34 if __name__ == "__main__":
35     print("Circle Area (r=5): {calculate_circle_area(5)}")
36     print("Fahrenheit to Celsius (32F): {fahrenheit_to_celsius(32)}")
37     print("Is 'A man, a plan, a canal: Panama' a palindrome?: {check_palindrome('A man, a plan, a canal: Panama')}")
38     print("Fibonacci (5): {fibonacci_iterative(5)}")
39     print("Random Password (len 12): {generate_random_password(12)}")
40

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\shash\rful-crud28> c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\OneDrive - Microsoft Corp\Documents\AAC A 9.4.py'
Circle Area (r=5): 78.53981633974483
Random Password (len 12): |fs^QAGISj7W
PS C:\Users\shash\rful-crud28> c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\OneDrive - Microsoft Corp\Documents\AAC A 9.4.py'
Circle Area (r=5): 78.53981633974483
Fahrenheit to Celsius (32F): 0.0
Is 'A man, a plan, a canal: Panama' a palindrome?: True
Fibonacci (5): [0, 1, 1, 2, 3]
Random Password (len 12): +'K0*^A{IR}N
PS C:\Users\shash\rful-crud28> []

```

Task 4: Converting Developer Comments into Structured Docstrings

Scenario

In a legacy project, developers have written long explanatory comments inside functions instead of proper docstrings. The team now wants to standardize documentation.

Task Description

You are given a Python script where functions contain detailed inline comments explaining their logic.

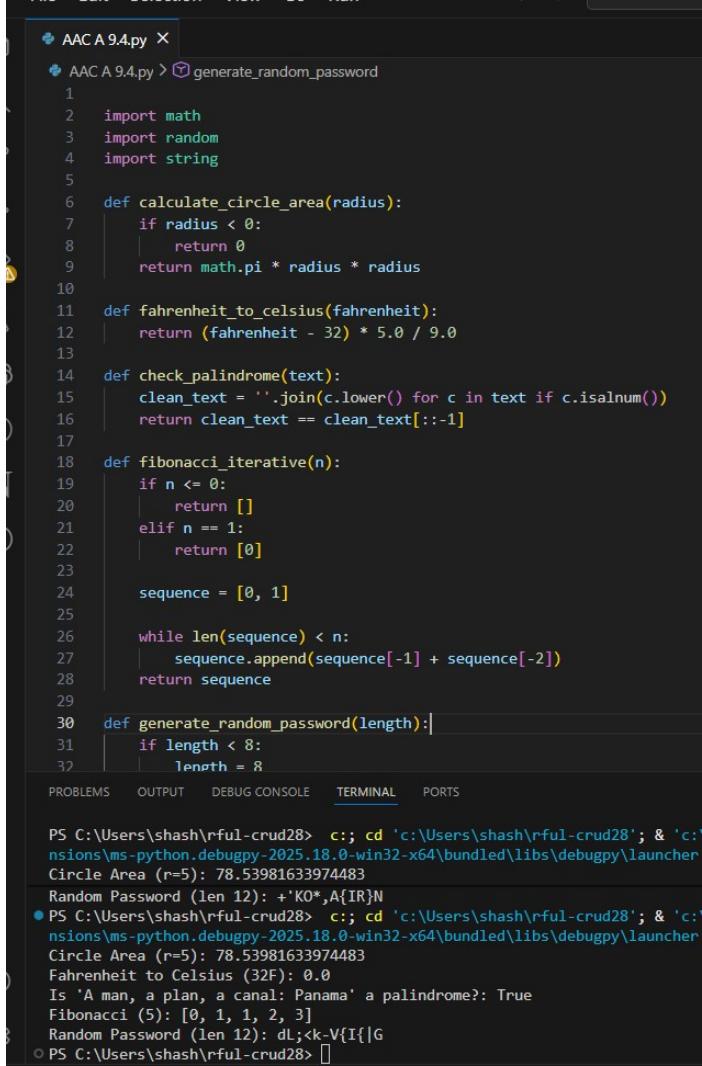
Use AI to:

- Automatically convert these comments into structured **Google-style or NumPy-style docstrings**
- Preserve the original meaning and intent of the comments
- Remove redundant inline comments after conversion

Expected Outcome

- Functions with clean, standardized docstrings
- Reduced clutter inside function bodies

- Improved consistency across the codebase

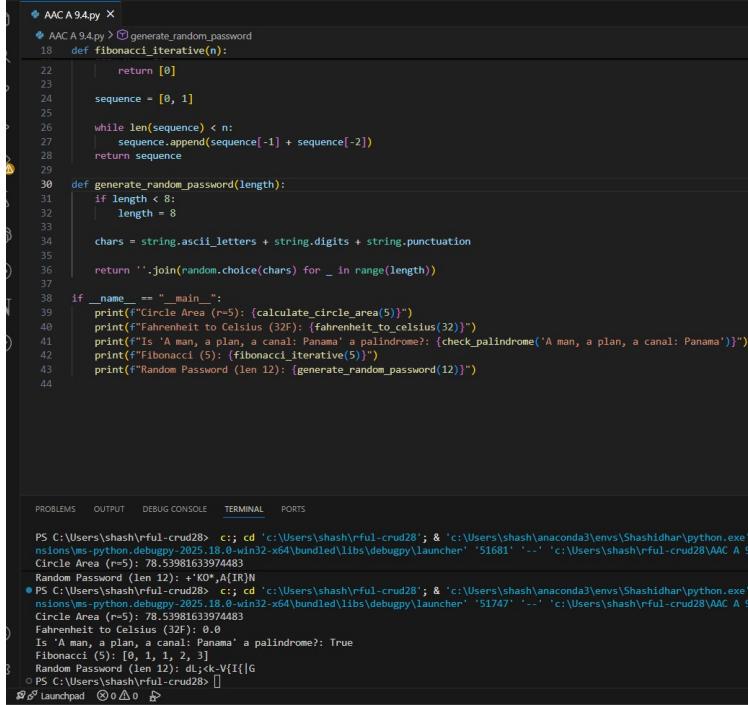


```

AAC A 9.4.py X
AAC A 9.4.py > generate_random_password
1
2 import math
3 import random
4 import string
5
6 def calculate_circle_area(radius):
7     if radius < 0:
8         return 0
9     return math.pi * radius * radius
10
11 def fahrenheit_to_celsius(fahrenheit):
12     return (fahrenheit - 32) * 5.0 / 9.0
13
14 def check_palindrome(text):
15     clean_text = ''.join(c.lower() for c in text if c.isalnum())
16     return clean_text == clean_text[::-1]
17
18 def fibonacci_iterative(n):
19     if n <= 0:
20         return []
21     elif n == 1:
22         return [0]
23
24     sequence = [0, 1]
25
26     while len(sequence) < n:
27         sequence.append(sequence[-1] + sequence[-2])
28     return sequence
29
30 def generate_random_password(length):
31     if length < 8:
32         length = 8
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\shash\rful-crud28> c;; cd 'c:\Users\shash\rful-crud28'; & 'c:
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher'
Circle Area (r=5): 78.53981633974483
Random Password (len 12): +K0*,A{IR}N
PS C:\Users\shash\rful-crud28> c;; cd 'c:\Users\shash\rful-crud28'; & 'c:
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher'
Circle Area (r=5): 78.53981633974483
Fahrenheit to Celsius (32F): 0.0
Is 'A man, a plan, a canal: Panama' a palindrome?: True
Fibonacci (5): [0, 1, 1, 2, 3]
Random Password (len 12): dl;<k-V{I{|G
PS C:\Users\shash\rful-crud28>

```



```

AAC A9.4.py
18 def fibonacci_iterative(n):
19     if n == 0:
20         return [0]
21     sequence = [0, 1]
22     while len(sequence) < n:
23         sequence.append(sequence[-1] + sequence[-2])
24     return sequence
25
26 def generate_random_password(length):
27     if length < 8:
28         length = 8
29
30     chars = string.ascii_letters + string.digits + string.punctuation
31
32     return ''.join(random.choice(chars) for _ in range(length))
33
34 if __name__ == "__main__":
35     print("Circle Area (r=5):", calculate_circle_area(5))
36     print("Fahrenheit to Celsius (32F):", fahrenheit_to_celsius(32))
37     print("Is 'A man, a plan, a canal: Panama' a palindrome?:", check_palindrome('A man, a plan, a canal: Panama'))
38     print("Fibonacci (5):", fibonacci_iterative(5))
39     print("Random Password (len 12):", generate_random_password(12))
40
41
42
43
44

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\shashirful-crud28> cd c:\Users\shashirful-crud28\; & "c:\Users\shashirful-crud28\anaconda3\envs\Shashidhar\python.exe" nelson\ms-python.debugger_2025.18.0.win32-x64\bundled\libs\debugpy\launcher"51681" -- "c:\Users\shashirful-crud28\AAC_A 9 Circle Area (r=5): 78.53981633974483
Random Password (len 12): +K0*A[IR]N
PS C:\Users\shashirful-crud28> cd c:\Users\shashirful-crud28\; & "c:\Users\shashirful-crud28\anaconda3\envs\Shashidhar\python.exe" nelson\ms-python.debugger_2025.18.0-win32-x64\bundled\libs\debugpy\launcher"51747" -- "c:\Users\shashirful-crud28\AAC_A 9 Circle Area (r=5): 78.53981633974483
Fahrenheit to Celsius (32F): 0.0
Is 'A man, a plan, a canal: Panama' a palindrome?: True
Fibonacci (5): [0, 1, 1, 2, 3]
Random Password (len 12): dLjkck-V{I{[]}
PS C:\Users\shashirful-crud28>

```

Task 5: Building a Mini Automatic Documentation Generator

Scenario

Your team wants a simple internal tool that helps developers start documenting new Python files quickly, without writing documentation from scratch.

Task Description

Design a small Python utility that:

- Reads a given .py file
- Automatically detects:
 - Functions
 - Classes
- Inserts **placeholder Google-style docstrings** for each detected function or class

AI tools may be used to assist in generating or refining this utility.

Note: The goal is **documentation scaffolding**, not perfect documentation.

Expected Outcome

- A working Python script that processes another .py file
- Automatically inserted placeholder docstrings
- Clear demonstration of how AI can assist in documentation automation

The screenshot shows a code editor with a Python script named AAC A 9.4.py. The script uses the ast module to parse and modify Python code, specifically adding docstrings to function definitions. It includes a process function that reads a file, adds docstrings, and writes the modified code back to the file. The terminal below shows the execution of the script, creating a file named sample_docs.py from sample.py.

```

File Edit Selection View Go Run ...
AAC A 9.4.py ...
1 import ast, os
2 def make_doc(node, ind):
3     doc = [f'{ind}'''[Summary of {node.name}]]', f'{ind}']
4     if isinstance(node, ast.FunctionDef):
5         args = [a.arg for a in node.args.args if a.arg != 'self']
6         if args: doc.extend([f'{ind}Args: '] + [f'{ind}    {a} (Any): [Desc]' for a in args] + [f'{ind}'])
7         doc.extend([f'{ind}Returns: ', f'{ind}    Any: [Desc]'])
8     else: doc.extend([f'{ind}Attributes: ', f'{ind}    [attr] (Any): [Desc]'])
9     return '\n'.join(doc + [f'{ind}''''\n'])
10
11 def process(path):
12     if not os.path.exists(path): return
13     with open(path, 'r') as f: lines = f.readlines()
14
15     nodes = [n for n in ast.walk(ast.parse(''.join(lines)))]
16     if isinstance(n, (ast.FunctionDef, ast.ClassDef)):
17         and not ast.get_docstring(n) and n.body]
18
19     for n in sorted(nodes, key=lambda x: x.body[0].lineno, reverse=True):
20         idx = n.body[0].lineno - 1
21         ind = lines[idx][:len(lines[idx]) - len(lines[idx].lstrip())]
22         lines.insert(idx, make_doc(n, ind))
23
24     out = path.replace('.py', '_docs.py')
25     with open(out, 'w') as f: f.writelines(lines)
26     print(f"Saved: {out}")
27
28 code = "class A:\n    def m(self, x): return x\n    def f(y): return y"
29 with open("sample.py", "w") as f: f.write(code)
30 if __name__ == "__main__": process(["sample.py"])

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\shash\rful-crud28> c;; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar_nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51747' '--' 'c:\Users\shash\rful-crud28'
Circle Area (r=5): 78.53981633974483
Fibonacci (5): [0, 1, 1, 2, 3]
Random Password (len 12): dL;k-V[I{|G
● PS C:\Users\shash\rful-crud28> c;; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar_nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58310' '--' 'c:\Users\shash\rful-crud28'
Saved: sample_docs.py
● PS C:\Users\shash\rful-crud28> c;; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar_nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58354' '--' 'c:\Users\shash\rful-crud28'
Saved: sample_docs.py
○ PS C:\Users\shash\rful-crud28> []

```

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