Name:O.ISRAEL H.No:2303A51825 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<br>Ms. B. Swathi<br>Dr. Sasanko Shekhar Gantayat<br>Mr. Md Sallauddin<br>Dr. Mathivanan<br>Mr. Y Srikanth<br>Ms. N Shilpa<br>Dr. Rishabh Mittal (Coordinator)<br>Dr. R. Prashant Kumar<br>Mr. Ankushavali MD<br>Mr. B Viswanath<br>Ms. Sujitha Reddy<br>Ms. A. Anitha<br>Ms. M.Madhuri<br>Ms. Katherashala Swetha<br>Ms. Velpula sumalatha<br>Mr. Bingi Raju<br>Mr. G. Kranthi | |
| Course Code | 23CS002PC304 | Course Title | AI Assisted Coding |
| Year/Sem | III/I | Regulation | R23 |
| Date and Day of Assignment | Week 5 - Thursday | Time(s) | 23CSBTB01 To 23CSBTB52 |
| Duration | 2 Hours | Applicable to Batches | All Batches |
| AssignmentNumber:9.4 (Present assignment number)/24(Total number of assignments) | | | |

| Q.No. | Question | *ExpectedTime to complete* |
|---|---|---|
| 1 | **Lab 9 – Documentation Generation: Automatic Documentation and Code Comments** | Week 5 |

**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.

**Lab Outcomes**

- 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

**Task 1: Auto-Generating Function Documentation in a Shared Codebase**

## Scenario

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.

## Task Description

You are given a Python script containing multiple functions without any docstrings.

Using an AI-assisted coding tool:

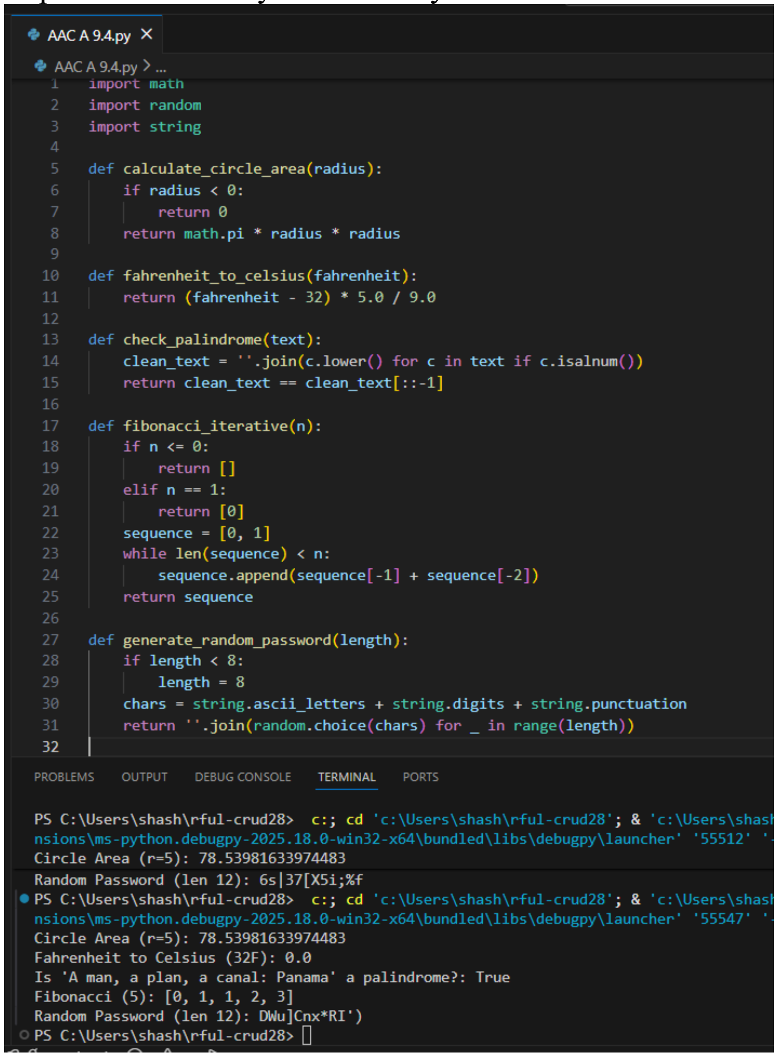
- Ask the AI to automatically generate **Google-style function docstrings** for each function
- Each docstring should include:
    - A brief description of the function
    - Parameters with data types
    - Return values
    - At least one example usage (if applicable)

Experiment with different prompting styles (zero-shot or context-based) to observe quality differences.

## Expected Outcome

- A Python script with well-structured Google-style docstrings
- Docstrings that clearly explain function behavior and usage

- Improved readability and usability of the codebase



```python
import math
import random
import string

def calculate_circle_area(radius):
    if radius < 0:
        return 0
    return math.pi * radius * radius

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5.0 / 9.0

def check_palindrome(text):
    clean_text = ''.join(c.lower() for c in text if c.isalnum())
    return clean_text == clean_text[::-1]

def fibonacci_iterative(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    sequence = [0, 1]
    while len(sequence) < n:
        sequence.append(sequence[-1] + sequence[-2])
    return sequence

def generate_random_password(length):
    if length < 8:
        length = 8
    chars = string.ascii_letters + string.digits + string.punctuation
    return ''.join(random.choice(chars) for _ in range(length))

```

```
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash
nsions\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
nsions\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): DWu]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)

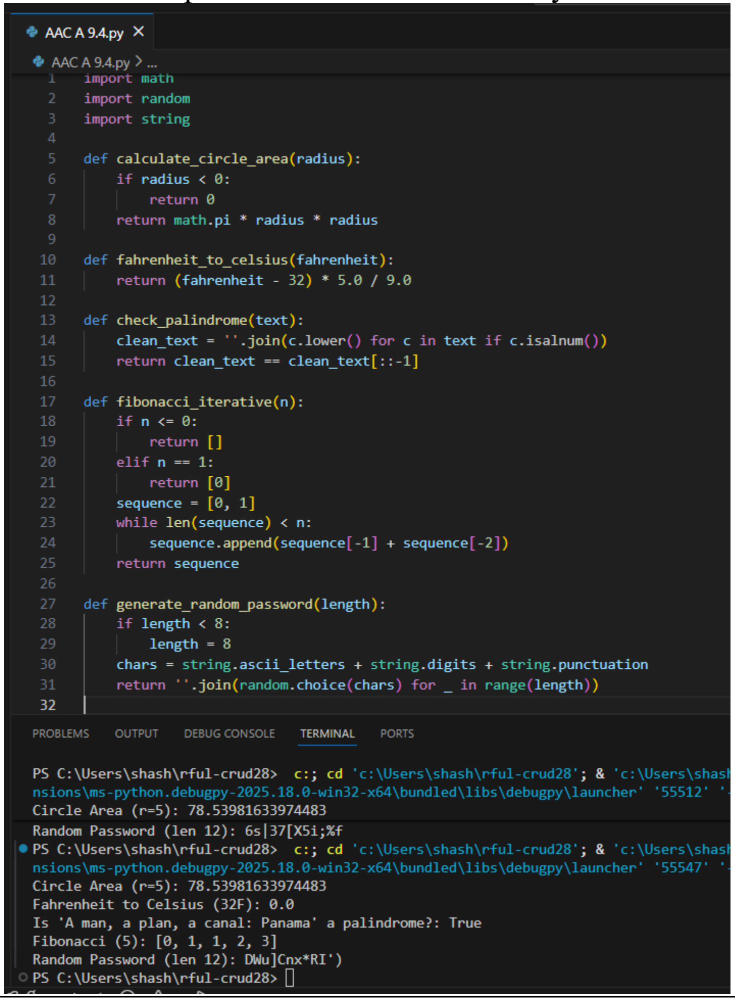
Use AI assistance to:

- 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

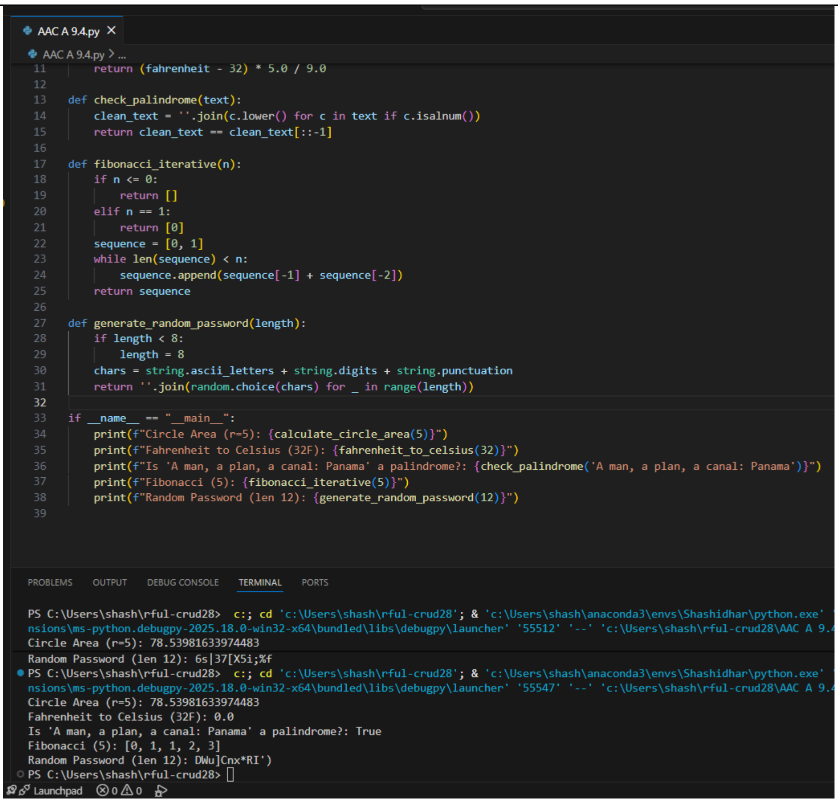
- A Python script with concise, meaningful inline comments
- Comments that explain *why* the logic exists, not *what* Python syntax does
- Noticeable improvement in code readability



```python
import math
import random
import string

def calculate_circle_area(radius):
    if radius < 0:
        return 0
    return math.pi * radius * radius

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5.0 / 9.0

def check_palindrome(text):
    clean_text = ''.join(c.lower() for c in text if c.isalnum())
    return clean_text == clean_text[::-1]

def fibonacci_iterative(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    sequence = [0, 1]
    while len(sequence) < n:
        sequence.append(sequence[-1] + sequence[-2])
    return sequence

def generate_random_password(length):
    if length < 8:
        length = 8
    chars = string.ascii_letters + string.digits + string.punctuation
    return ''.join(random.choice(chars) for _ in range(length))

```

```
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash
nsions\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
nsions\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): DWu]Cnx*RI')
PS C:\Users\shash\rful-crud28> 
```

-



AAC A 9.4.py

```python
    return (fahrenheit - 32) * 5.0 / 9.0

def check_palindrome(text):
    clean_text = ''.join(c.lower() for c in text if c.isalnum())
    return clean_text == clean_text[::-1]

def fibonacci_iterative(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    sequence = [0, 1]
    while len(sequence) < n:
        sequence.append(sequence[-1] + sequence[-2])
    return sequence

def generate_random_password(length):
    if length < 8:
        length = 8
    chars = string.ascii_letters + string.digits + string.punctuation
    return ''.join(random.choice(chars) for _ in range(length))

if __name__ == "__main__":
    print(f"Circle Area (r=5): {calculate_circle_area(5)}")
    print(f"Fahrenheit to Celsius (32F): {fahrenheit_to_celsius(32)}")
    print(f"Is 'A man, a plan, a canal: Panama' a palindrome?: {check_palindrome('A man, a plan, a canal: Panama')}")
    print(f"Fibonacci (5): {fibonacci_iterative(5)}")
    print(f"Random Password (len 12): {generate_random_password(12)}")
```

```
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55512' '--' 'c:\Users\shash\rful-crud28\AAC A 9.4
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\anaconda3\envs\Shashidhar\python.exe'
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55547' '--' 'c:\Users\shash\rful-crud28\AAC A 9.4
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): DWu]Cnx*RI')
PS C:\Users\shash\rful-crud28>
```

## 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

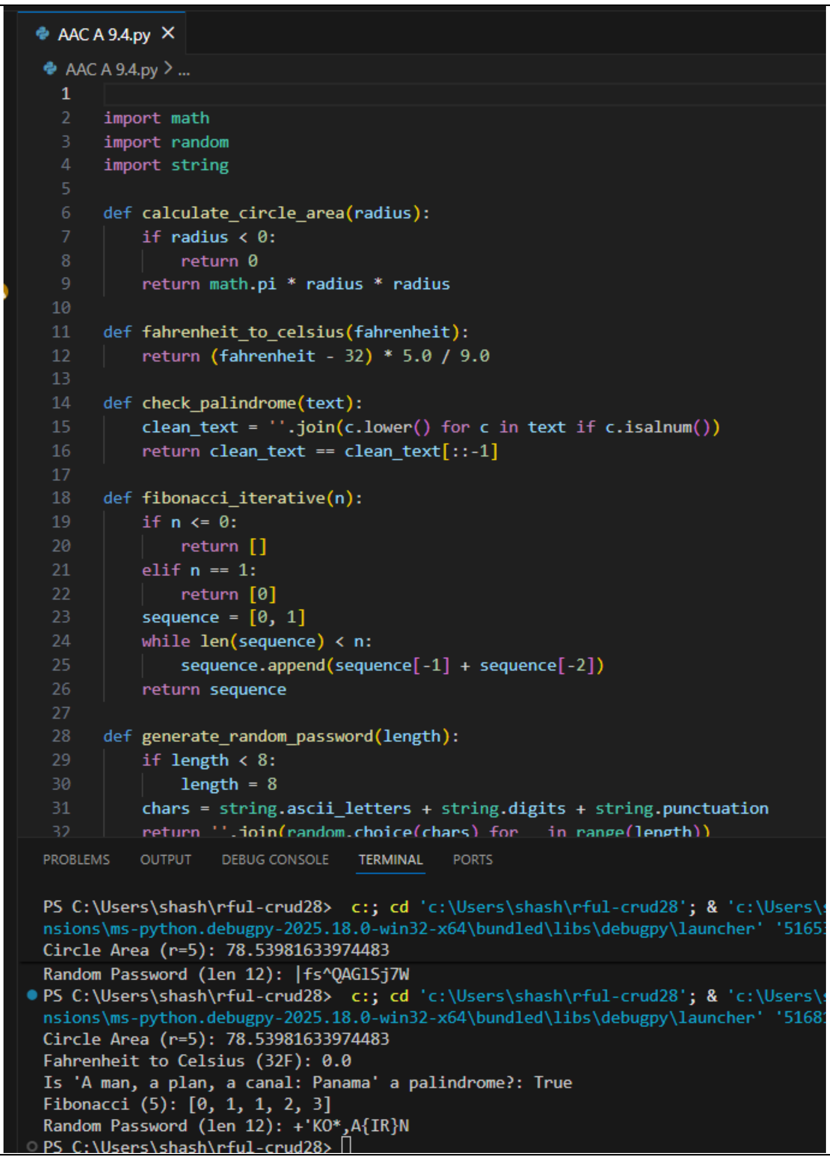
Provide a complete Python module to an AI tool and instruct it to automatically generate a **module-level docstring** at the top of the file that includes:

- 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

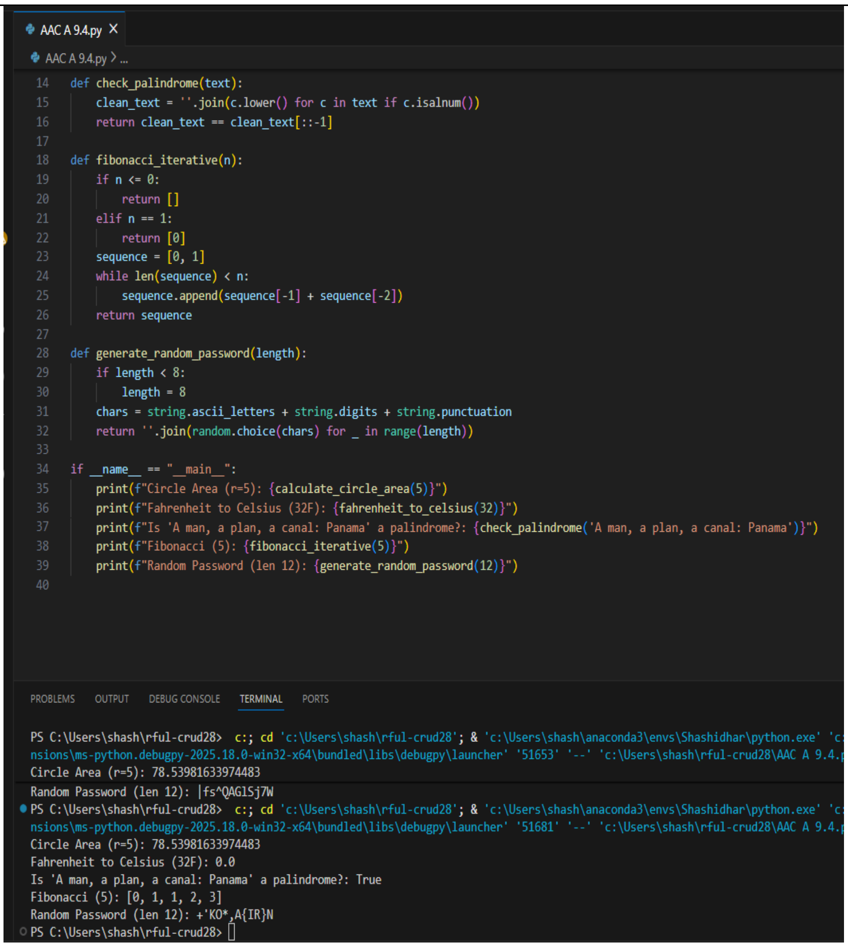
- A well-written multi-line module-level docstring
- Clear overview of what the module does and how to use it
- Documentation suitable for real-world projects or repositories

|  |  |  |  |
|---|---|---|---|
|  | • |  |  |

AAC A 9.4.py

```python

import math
import random
import string

def calculate_circle_area(radius):
    if radius < 0:
        return 0
    return math.pi * radius * radius

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5.0 / 9.0

def check_palindrome(text):
    clean_text = ''.join(c.lower() for c in text if c.isalnum())
    return clean_text == clean_text[::-1]

def fibonacci_iterative(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    sequence = [0, 1]
    while len(sequence) < n:
        sequence.append(sequence[-1] + sequence[-2])
    return sequence

def generate_random_password(length):
    if length < 8:
        length = 8
    chars = string.ascii_letters + string.digits + string.punctuation
    return ''.join(random.choice(chars) for   in range(length))
```

```
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '5165
Circle Area (r=5): 78.53981633974483
Random Password (len 12): |fs^QAGlSj7W
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\
nsions\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>
```



```python
def check_palindrome(text):
    clean_text = ''.join(c.lower() for c in text if c.isalnum())
    return clean_text == clean_text[::-1]

def fibonacci_iterative(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    sequence = [0, 1]
    while len(sequence) < n:
        sequence.append(sequence[-1] + sequence[-2])
    return sequence

def generate_random_password(length):
    if length < 8:
        length = 8
    chars = string.ascii_letters + string.digits + string.punctuation
    return ''.join(random.choice(chars) for _ in range(length))

if __name__ == "__main__":
    print(f"Circle Area (r=5): {calculate_circle_area(5)}")
    print(f"Fahrenheit to Celsius (32F): {fahrenheit_to_celsius(32)}")
    print(f"Is 'A man, a plan, a canal: Panama' a palindrome?: {check_palindrome('A man, a plan, a canal: Panama')}")
    print(f"Fibonacci (5): {fibonacci_iterative(5)}")
    print(f"Random Password (len 12): {generate_random_password(12)}")

```

```
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51653' '--' 'c:\Users\shash\rful-crud28\AAC A 9.4.
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\anaconda3\envs\Shashidhar\python.exe' 'c:
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51681' '--' 'c:\Users\shash\rful-crud28\AAC A 9.4.
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> 
```

## 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

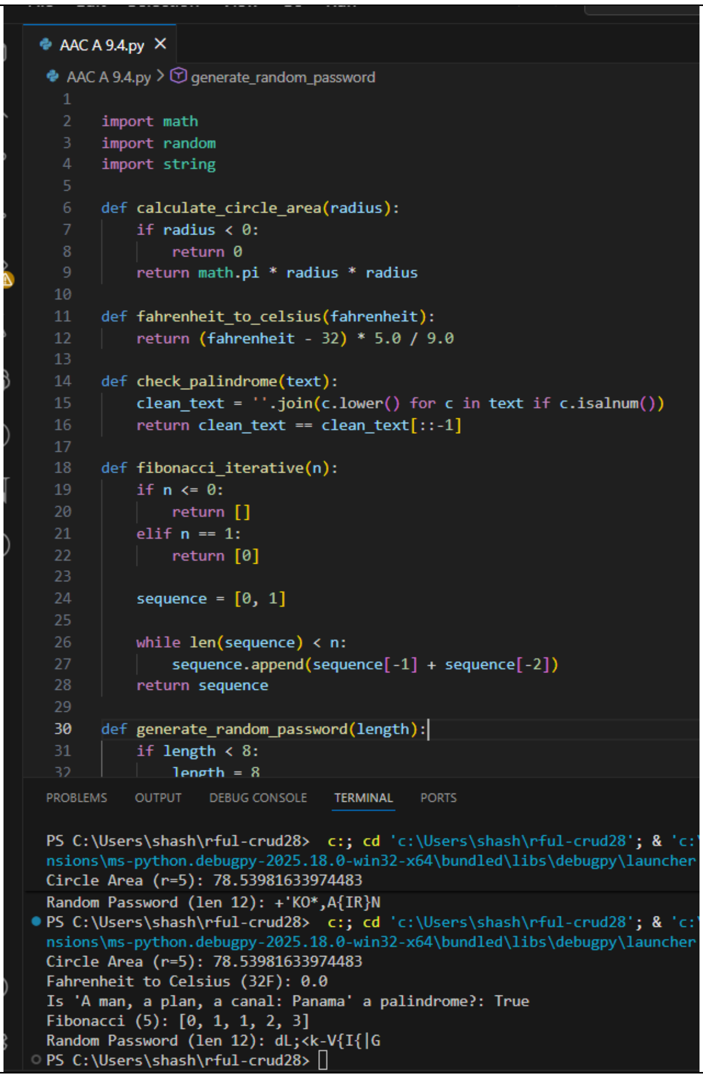
You are given a Python script where functions contain detailed inline comments explaining their logic.
Use AI to:

- 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 | |

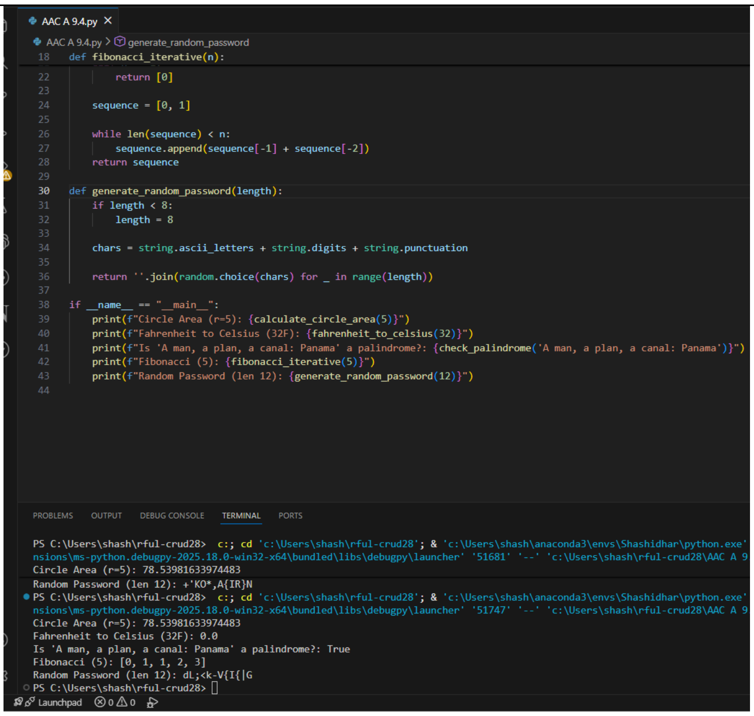


AAC A 9.4.py > generate_random_password

```python

import math
import random
import string

def calculate_circle_area(radius):
    if radius < 0:
        return 0
    return math.pi * radius * radius

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5.0 / 9.0

def check_palindrome(text):
    clean_text = ''.join(c.lower() for c in text if c.isalnum())
    return clean_text == clean_text[::-1]

def fibonacci_iterative(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]

    sequence = [0, 1]

    while len(sequence) < n:
        sequence.append(sequence[-1] + sequence[-2])
    return sequence

def generate_random_password(length):
    if length < 8:
        length = 8
```

```
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): +'KO*,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 A 9.4.py > generate_random_password

```python
def fibonacci_iterative(n):
        return [0]

    sequence = [0, 1]

    while len(sequence) < n:
        sequence.append(sequence[-1] + sequence[-2])
    return sequence

def generate_random_password(length):
    if length < 8:
        length = 8

    chars = string.ascii_letters + string.digits + string.punctuation

    return ''.join(random.choice(chars) for _ in range(length))

if __name__ == "__main__":
    print(f"Circle Area (r=5): {calculate_circle_area(5)}")
    print(f"Fahrenheit to Celsius (32F): {fahrenheit_to_celsius(32)}")
    print(f"Is 'A man, a plan, a canal: Panama' a palindrome?: {check_palindrome('A man, a plan, a canal: Panama')}")
    print(f"Fibonacci (5): {fibonacci_iterative(5)}")
    print(f"Random Password (len 12): {generate_random_password(12)}")
```

```
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51681' '--' 'c:\Users\shash\rful-crud28\AAC A 9
Circle Area (r=5): 78.53981633974483
Random Password (len 12): +'KO*,A{IR}N
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51747' '--' 'c:\Users\shash\rful-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): dL;<k-V{I{|G
PS C:\Users\shash\rful-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

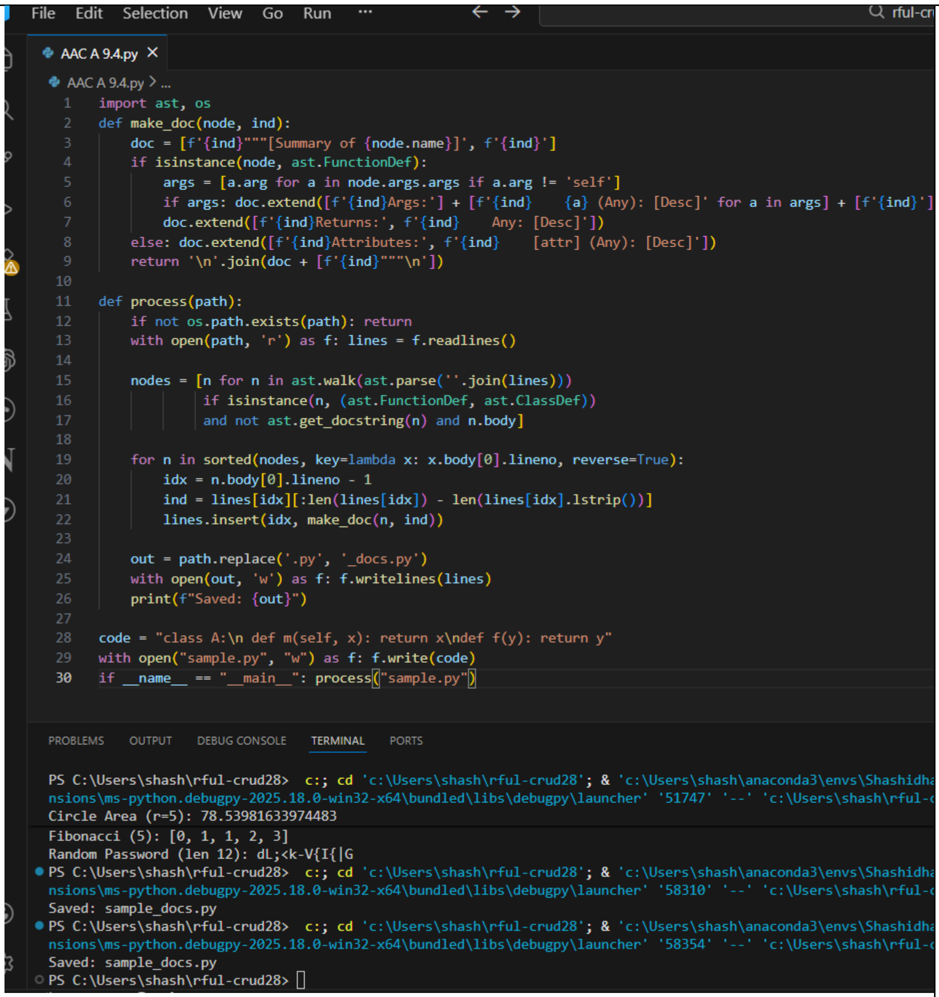
Design a small Python utility that:

- 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


File Edit Selection View Go Run ···
rful-cr
AAC A 9.4.py
AAC A 9.4.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```python
import ast, os
def make_doc(node, ind):
    doc = [f'{ind}"""[Summary of {node.name}]', f'{ind}']
    if isinstance(node, ast.FunctionDef):
        args = [a.arg for a in node.args.args if a.arg != 'self']
        if args: doc.extend([f'{ind}Args:'] + [f'{ind}    {a} (Any): [Desc]' for a in args] + [f'{ind}']
        doc.extend([f'{ind}Returns:', f'{ind}    Any: [Desc]'])
    else: doc.extend([f'{ind}Attributes:', f'{ind}    [attr] (Any): [Desc]'])
    return '\n'.join(doc + [f'{ind}"""\n'])

def process(path):
    if not os.path.exists(path): return
    with open(path, 'r') as f: lines = f.readlines()

    nodes = [n for n in ast.walk(ast.parse(''.join(lines)))
             if isinstance(n, (ast.FunctionDef, ast.ClassDef))
             and not ast.get_docstring(n) and n.body]

    for n in sorted(nodes, key=lambda x: x.body[0].lineno, reverse=True):
        idx = n.body[0].lineno - 1
        ind = lines[idx][:len(lines[idx]) - len(lines[idx].lstrip())]
        lines.insert(idx, make_doc(n, ind))

    out = path.replace('.py', '_docs.py')
    with open(out, 'w') as f: f.writelines(lines)
    print(f"Saved: {out}")

code = "class A:\n def m(self, x): return x\ndef f(y): return y"
with open("sample.py", "w") as f: f.write(code)
if __name__ == "__main__": process("sample.py")
```

```
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidha
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51747' '--' 'c:\Users\shash\rful-c
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\Shashidha
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58310' '--' 'c:\Users\shash\rful-c
Saved: sample_docs.py
PS C:\Users\shash\rful-crud28>  c:; cd 'c:\Users\shash\rful-crud28'; & 'c:\Users\shash\anaconda3\envs\Shashidha
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58354' '--' 'c:\Users\shash\rful-c
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**