AI ASS 9.4

Name:A.Harshita

H.T.NO:2303A52211

Batch:43

Task 1

Generate google-style docstrings for the following Python functions.

Each docstring should include:

- A brief description of the function

- Parameters with data types and short explanations

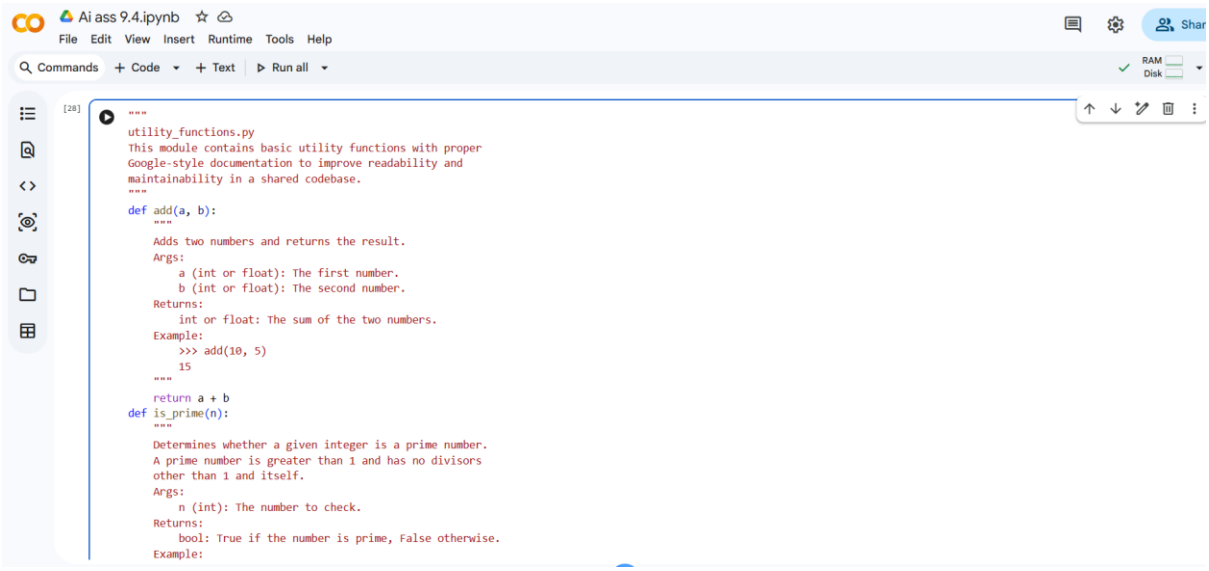
- Return values with data types

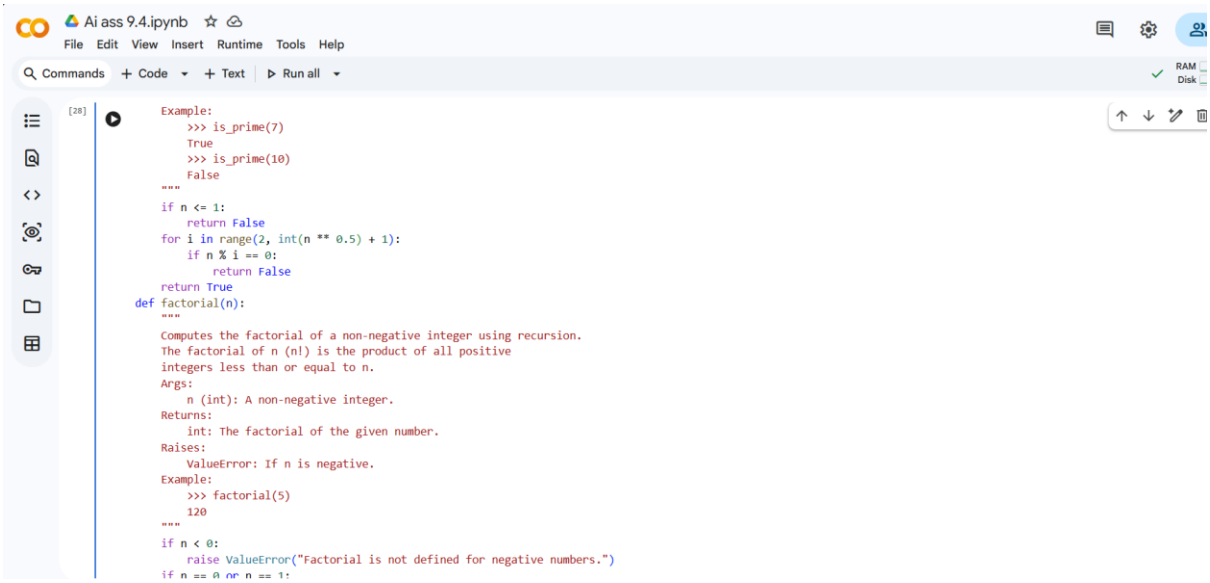
- At least one example usage in doctest format (if applicable)

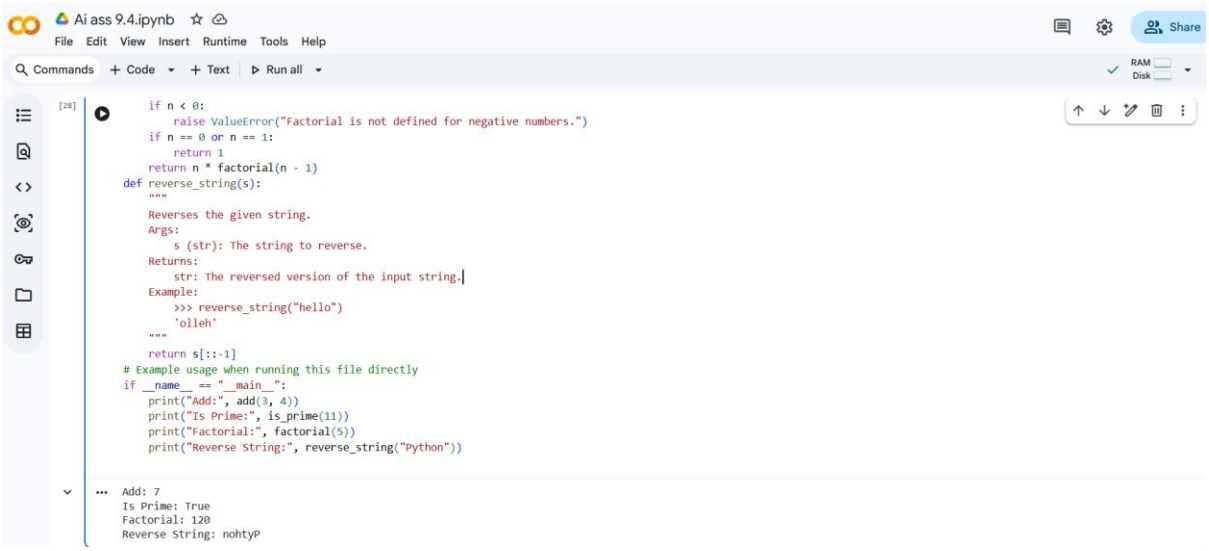
Do not modify the original function logic.

Ensure the formatting follows the Google Python Style Guide.

Keep the tone professional and clear for use in a shared codebase.







Task2

You are a senior Python developer reviewing a codebase for readability improvements.

The following Python script contains complex logic, including loops, conditional statements, and algorithms. The code works correctly but lacks clarity.

Your task:

- Insert concise inline comments ONLY where the logic is complex or non-obvious.

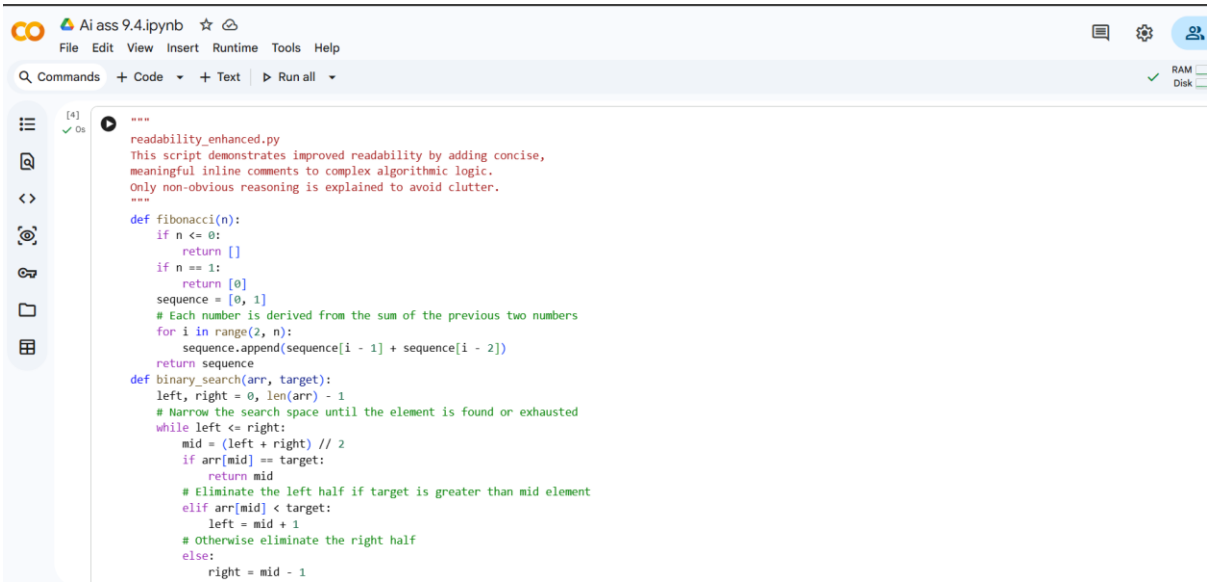
- Explain WHY the logic exists, not what basic Python syntax does.

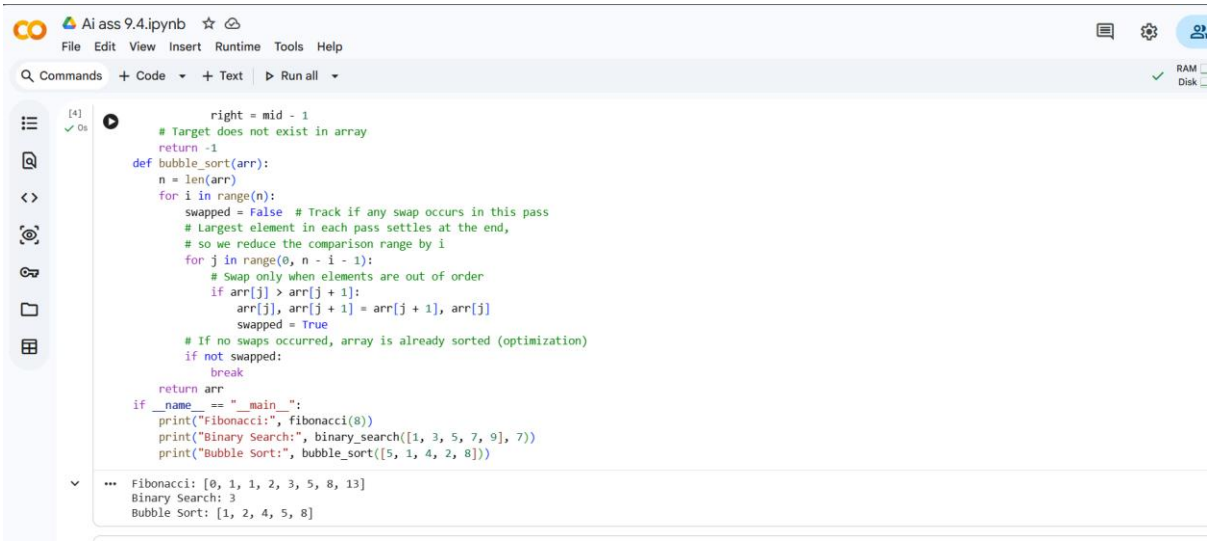
- Avoid commenting on trivial or self-explanatory lines.

- Do not modify the original logic or structure.

- Keep comments professional and minimal to avoid clutter.

Return the improved version of the script with meaningful inline comments added.





Task 3

You are a Python documentation expert preparing a module for internal or public use.

Analyze the following complete Python module and generate a professional multi-line module-level docstring to be placed at the top of the file.

The docstring must include:

- A clear explanation of the module's purpose

- Required libraries or dependencies

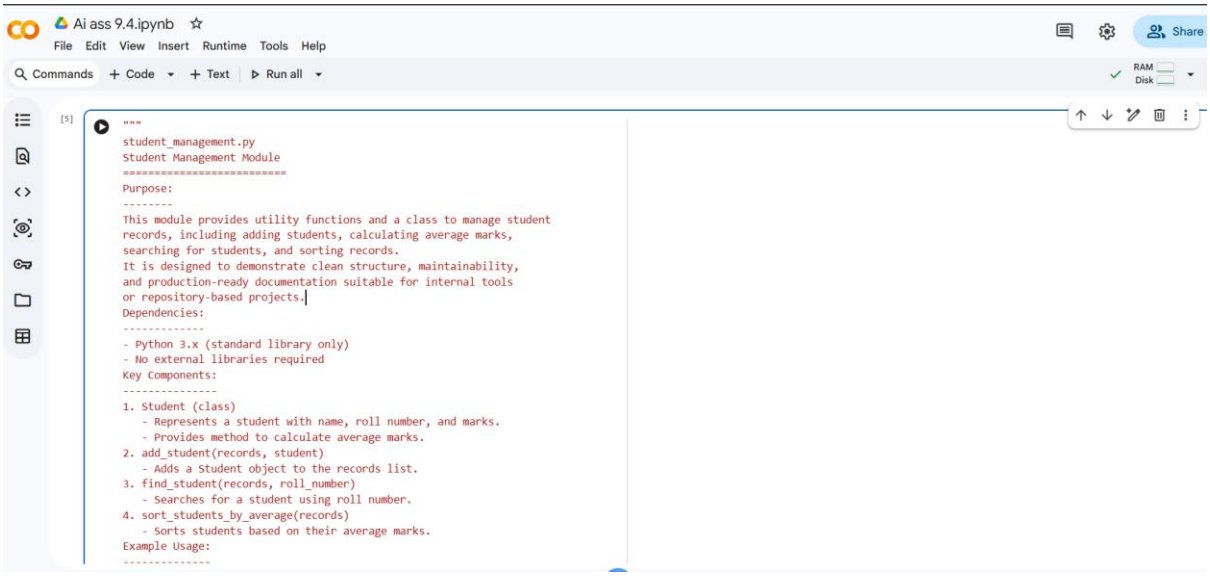
- A brief description of key functions and classes

- A short example demonstrating how to use the module

- Professional tone suitable for production code

Do not modify the existing code.

Only generate the module-level docstring.



Task 4

You are a Python refactoring specialist helping standardize documentation in a legacy project.

The following script contains detailed inline comments inside functions explaining their behavior. These should be converted into structured docstrings.

Your task:

- Convert relevant explanatory comments into proper Google-style (or NumPy-style) docstrings.

- Preserve the original meaning and intent.

- Include:

  - Description

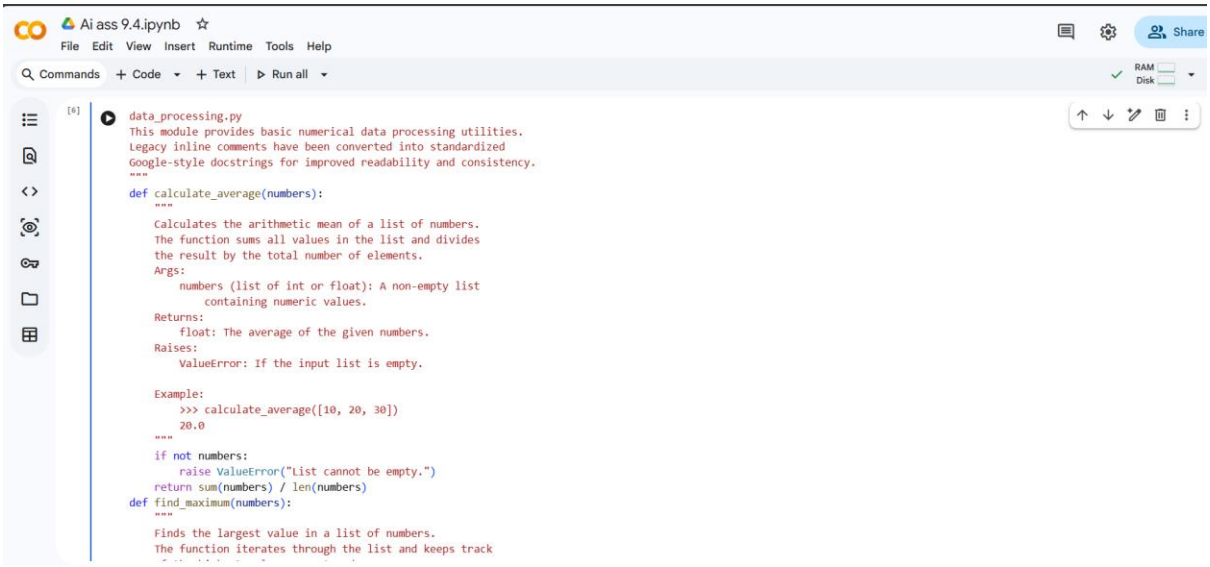
  - Parameters with types

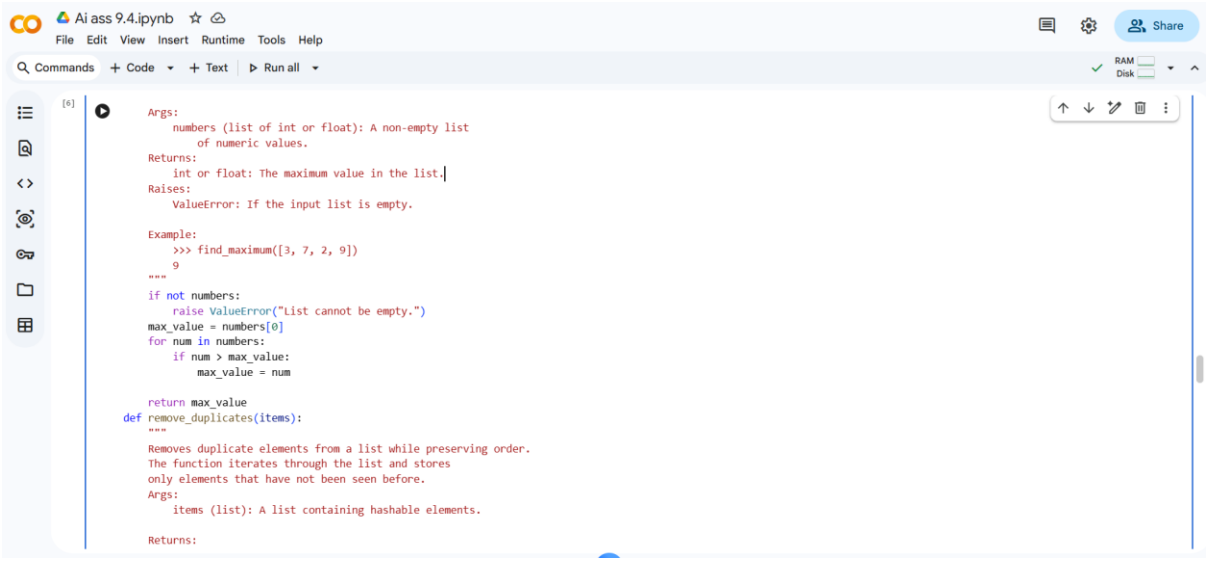
  - Return values with types

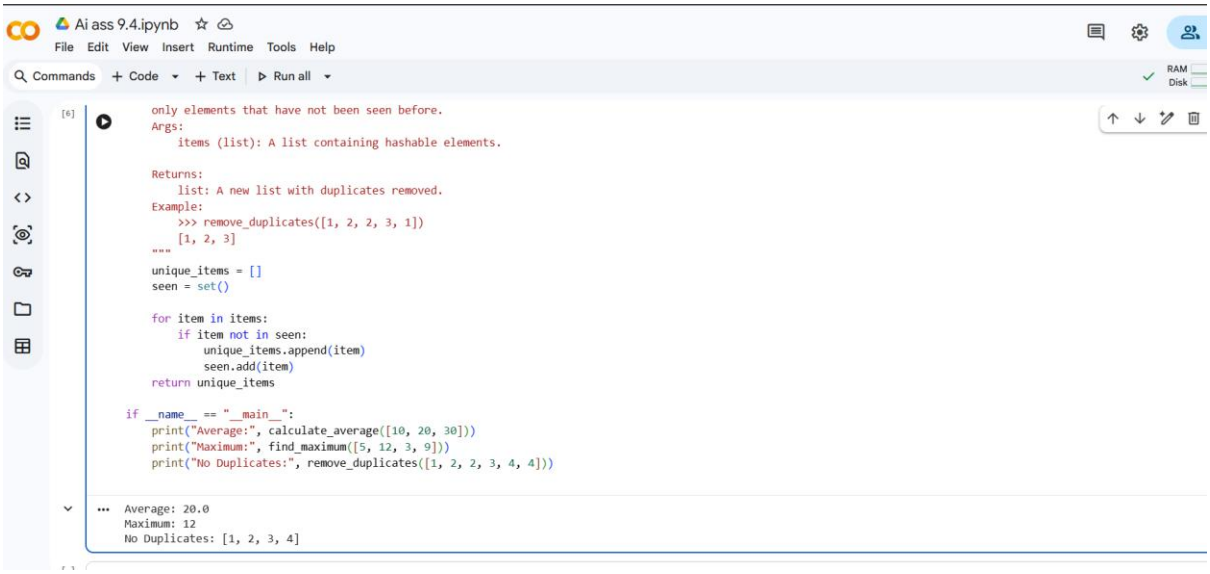
- Remove redundant inline comments after conversion.

- Do not modify function logic.

Return the cleaned and standardized version of the script.


Ai ass 9.4.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[6]
data_processing.py
This module provides basic numerical data processing utilities.
Legacy inline comments have been converted into standardized
Google-style docstrings for improved readability and consistency.
"""
def calculate_average(numbers):
"""
Calculates the arithmetic mean of a list of numbers.
The function sums all values in the list and divides
the result by the total number of elements.
Args:
numbers (list of int or float): A non-empty list
containing numeric values.
Returns:
float: The average of the given numbers.
Raises:
ValueError: If the input list is empty.
Example:
>>> calculate_average([10, 20, 30])
20.0
"""
if not numbers:
raise ValueError("List cannot be empty.")
return sum(numbers) / len(numbers)
def find_maximum(numbers):
"""
Finds the largest value in a list of numbers.
The function iterates through the list and keeps track


Ai ass 9.4.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[6]
Args:
numbers (list of int or float): A non-empty list
of numeric values.
Returns:
int or float: The maximum value in the list.
Raises:
ValueError: If the input list is empty.
Example:
>>> find_maximum([3, 7, 2, 9])
9
"""
if not numbers:
raise ValueError("List cannot be empty.")
max_value = numbers[0]
for num in numbers:
if num > max_value:
max_value = num
return max_value
def remove_duplicates(items):
"""
Removes duplicate elements from a list while preserving order.
The function iterates through the list and stores
only elements that have not been seen before.
Args:
items (list): A list containing hashable elements.
Returns:



Task 5

You are a Python developer building an internal documentation scaffolding tool.Design a small Python utility that:

- Reads a given .py file

- Automatically detects all functions and classes

- Inserts placeholder Google-style docstrings if none exist

- Preserves indentation and formatting

- Does not modify existing docstrings

The generated placeholder docstrings should include:

- Short description placeholder - Args section

- Returns section (if applicable)

The goal is documentation scaffolding, not perfect documentation.

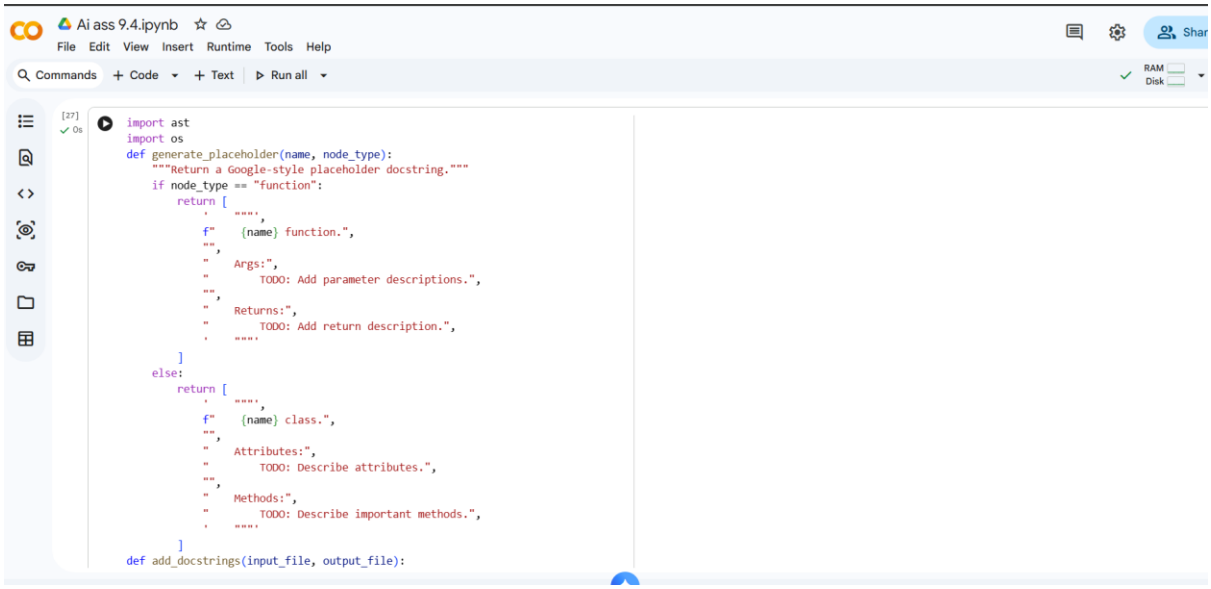
Provide:

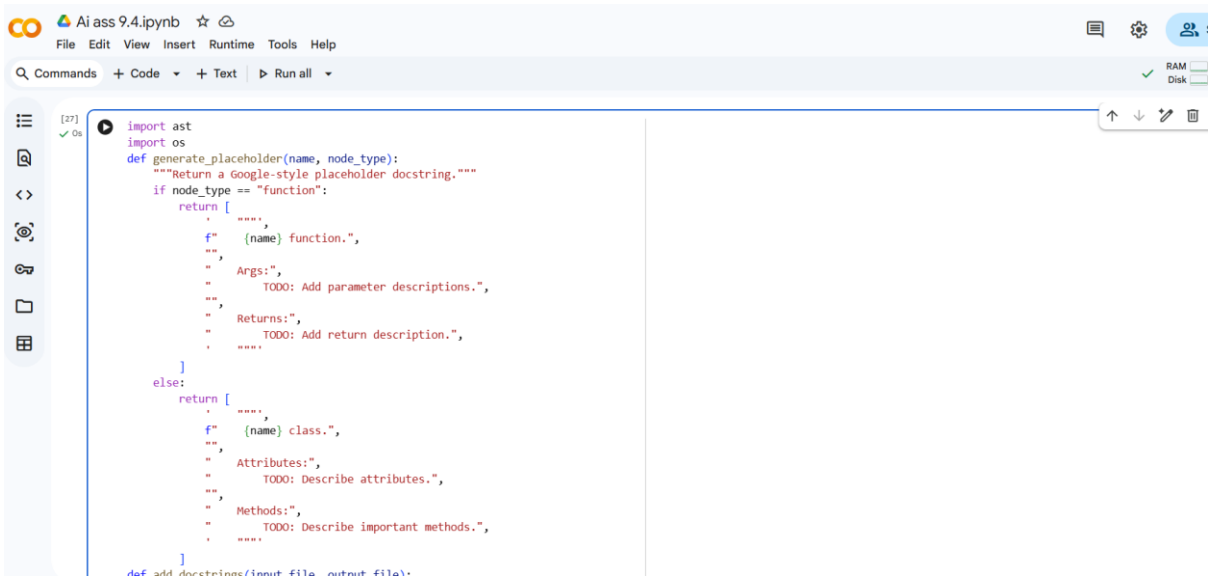
1. A complete working Python script

2. Clear explanation of how it works

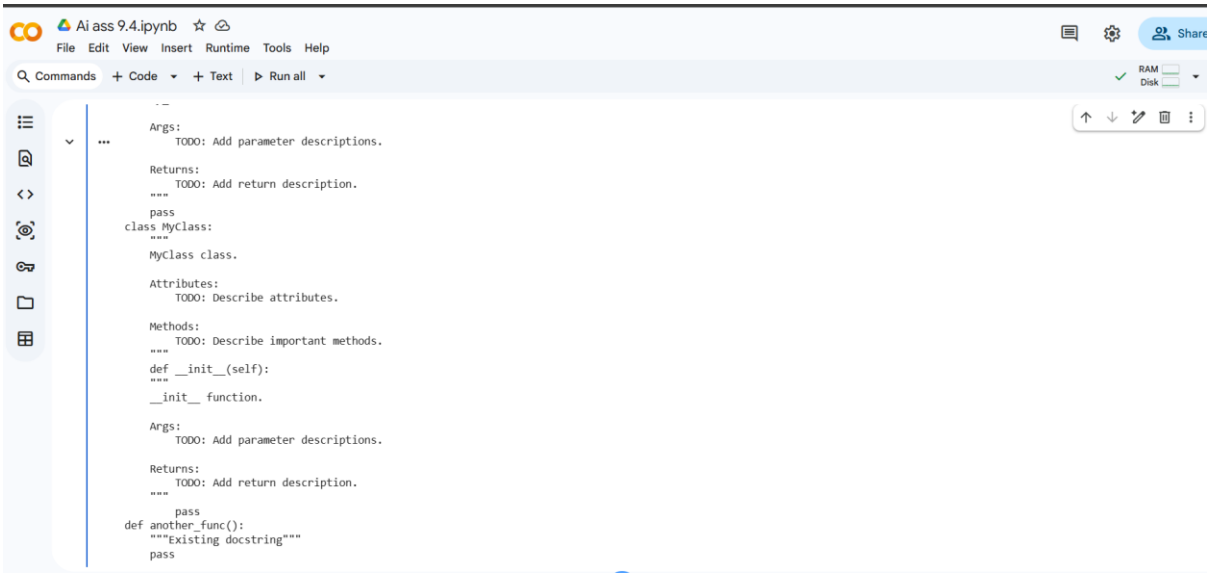
3. Example of how to run it

Ensure the solution is clean, readable, and suitable for internal tooling.


Ai ass 9.4.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
RAM
Disk
[27]
0s
import ast
import os
def generate_placeholder(name, node_type):
"""Return a Google-style placeholder docstring."""
if node_type == "function":
return [
' """',
f" {name} function.",
"",
" Args:",
" TODO: Add parameter descriptions.",
"",
" Returns:",
" TODO: Add return description.",
' """'
]
else:
return [
' """',
f" {name} class.",
"",
" Attributes:",
" TODO: Describe attributes.",
"",
" Methods:",
" TODO: Describe important methods.",
' """'
]
def add_docstrings(input_file, output_file):


Ai ass 9.4.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
RAM
Disk
[27]
0s
import ast
import os
def generate_placeholder(name, node_type):
"""Return a Google-style placeholder docstring."""
if node_type == "function":
return [
' """',
f" {name} function.",
"",
" Args:",
" TODO: Add parameter descriptions.",
"",
" Returns:",
" TODO: Add return description.",
' """'
]
else:
return [
' """',
f" {name} class.",
"",
" Attributes:",
" TODO: Describe attributes.",
"",
" Methods:",
" TODO: Describe important methods.",
' """'
]


Ai ass 9.4.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
RAM
Disk
[27]
0s
dummy_code = '''
def my_function(arg1, arg2):
pass
class MyClass:
def __init__(self):
pass
def another_func():
"""Existing docstring"""
pass
'''
with open("input.py", "w") as f:
f.write(dummy_code)
add_docstrings("input.py", "output.py")
print("Documentation scaffolding completed successfully!")
# Optionally, print the content of the output file
print("\nContent of output.py:")
with open("output.py", "r") as f:
print(f.read())
# Clean up the dummy files
os.remove("input.py")
os.remove("output.py")
Documentation scaffolding completed successfully!
Content of output.py:
def my_function(arg1, arg2):
"""
my_function function.


Ai ass 9.4.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
RAM
Disk
Args:
TODO: Add parameter descriptions.
Returns:
TODO: Add return description.
"""
pass
class MyClass:
"""
MyClass class.
Attributes:
TODO: Describe attributes.
Methods:
TODO: Describe important methods.
"""
def __init__(self):
"""
__init__ function.
Args:
TODO: Add parameter descriptions.
Returns:
TODO: Add return description.
"""
pass
def another_func():
"""Existing docstring"""
pass