ASSIGNMENT : 9.3

H.T.N.O : 2403A51292

BATCH : 12

Task :1

Basic Docstring Generation

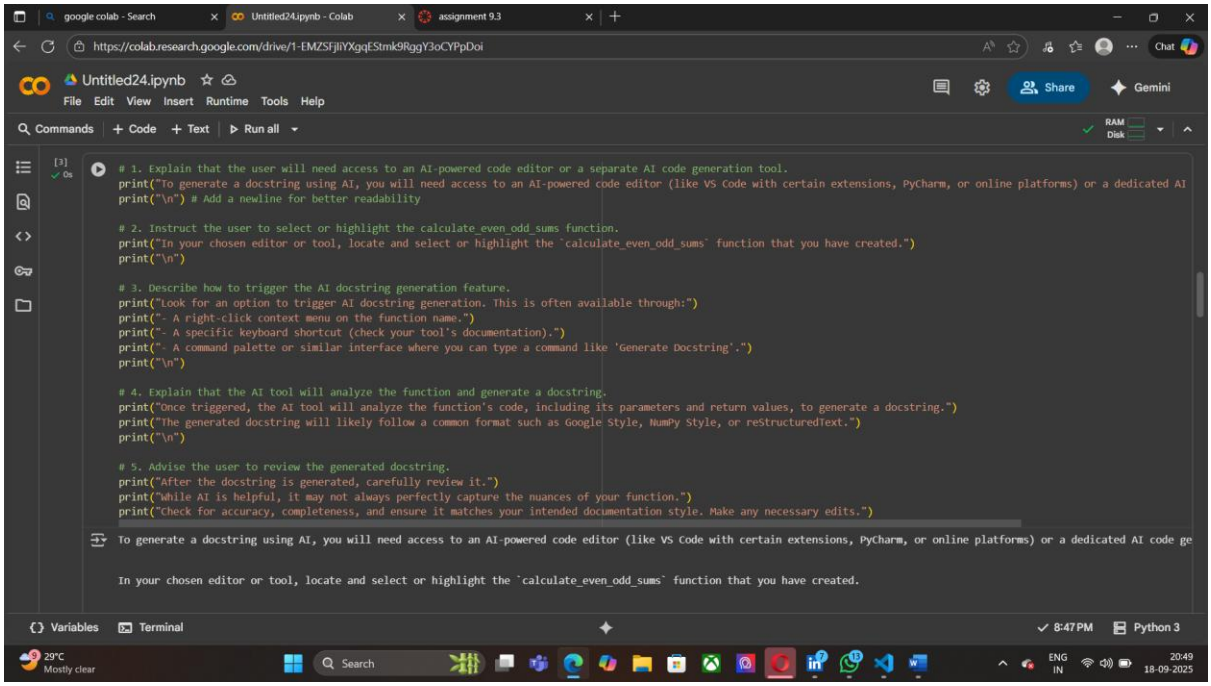
• Write python function to return sum of even and odd numbers in the given list.
• Incorporate manual docstring in code with Google Style
• Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.
• Compare the AI-generated docstring with your manually written one

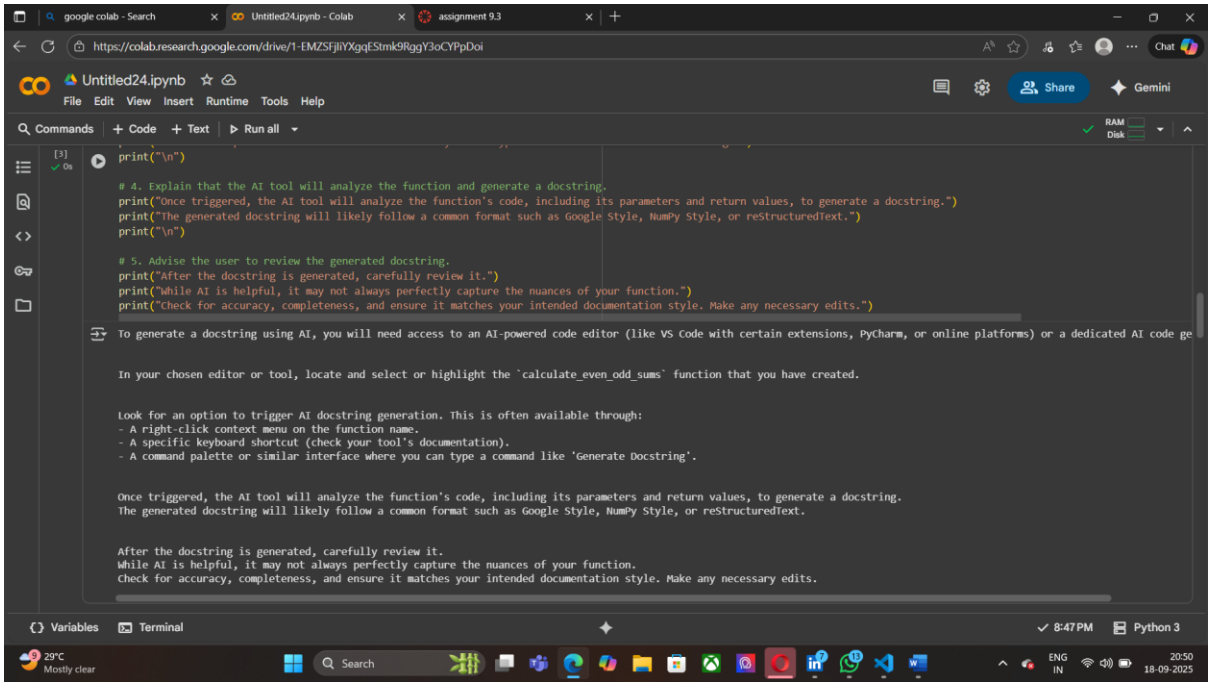
Expected Output#2: Students critically analyze AI-generated code comments.

Prompt:

Write a Python function that returns the sum of even and odd numbers in a given list.

- Manually add a Google-style docstring to the function describing its purpose, parameters, and return values.
- Use an AI-assisted tool (such as Copilot or Cursor AI) to generate a docstring for the same function.
- Compare the AI-generated docstring with your manually written one and note any differences in clarity, detail, or formatting.





TASK 2 :

Automatic Inline Comments

• Write python program for sru_student class with attributes like name, roll no., hostel_status and fee_update method and display_details method.

• Write comments manually for each line/code block

• Ask an AI tool to add inline comments explaining each line/step.

• Compare the AI-generated comments with your manually written one.

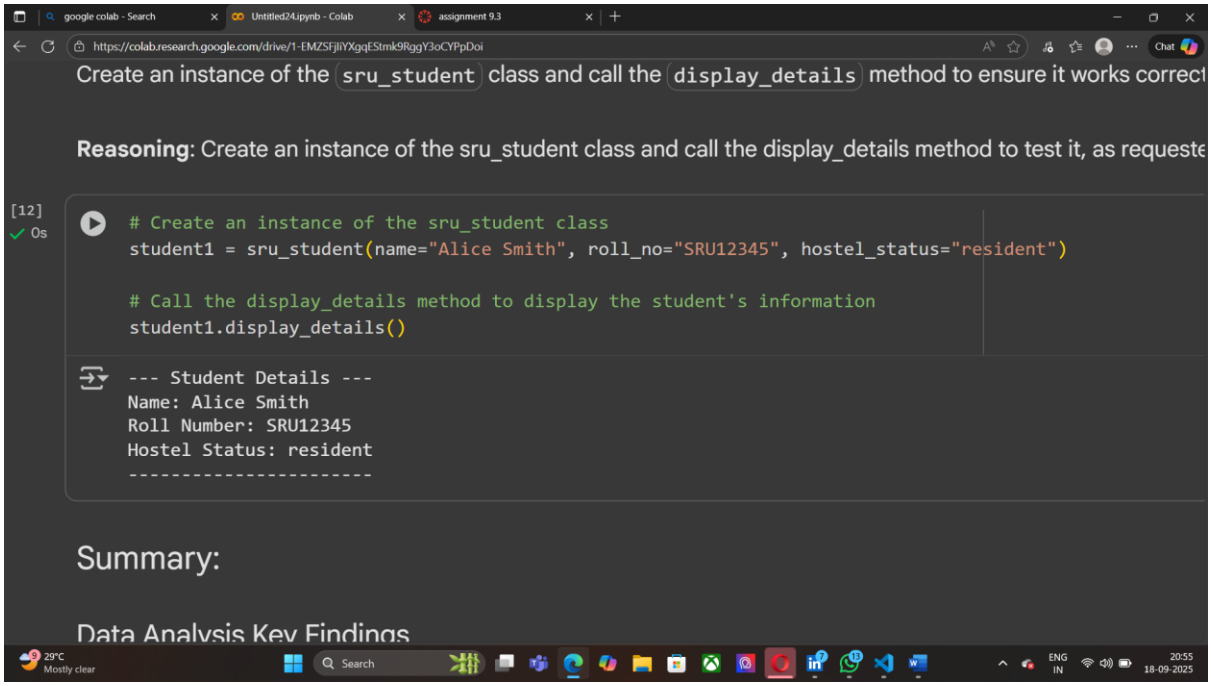
Expected Output#2: Students critically analyze AI-generated code comments.

Prompt:

Write a Python program defining an sru_student class with the following:

- Attributes: name, roll_no, hostel_status
- Methods: fee_update and display_details

1. Manually add inline comments for each line or code block, explaining the logic.
2. Use an AI tool (such as Copilot or Cursor AI) to generate inline comments for the same code.
3. Compare the AI-generated comments with your manually written ones and analyze the differences in clarity, accuracy, and usefulness.



Task 3 :

• Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
• Incorporate manual docstring in code with NumPy Style
• Use AI assistance to generate a module-level docstring + individual function docstrings.
• Compare the AI-generated docstring with your manually written one.
Expected Output#3: Students learn structured documentation for multi-function scripts

Prompt:

Write a Python script containing 3–4 functions (for example, a calculator with add, subtract, multiply, and divide functions).

1. Manually add NumPy-style docstrings to each function.
2. Use an AI tool (such as Copilot or Cursor AI) to generate a module-level docstring and individual function docstrings.
3. Compare the AI-generated docstrings with your manually written ones, noting differences in structure, clarity, and completeness.

google colab - Search | Untitled24.ipynb - Colab | assignment 9.3

https://colab.research.google.com/drive/1-EMZSFjliYXgqEStmk9RggY3oCYPpDoi#scrollTo=5d6d987b

Untitled24.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[17]

```python
# 1. Explain that the user will need access to an AI-powered code editor or a separate AI code generation tool.
print("To generate docstrings using an AI tool, you will need access to an AI-powered code editor (like VS Code with certain extensions, PyCharm, or online platforms that offer
print("\n") # Add a newline for better readability

# 2. Instruct the user to open the Python script containing the calculator functions in their chosen environment.
print("Open the Python script containing the `add`, `subtract`, `multiply`, and `divide` functions in your chosen development environment or AI tool.")
print("\n") # Add a newline for better readability

# 3. Describe how to trigger the AI docstring generation feature.
print("To generate docstrings:")
print("- For a module-level docstring, some tools might offer an option when the file is open or when selecting the entire file content. Look for options like 'Generate Module D
print("- For individual function docstrings, select or place your cursor within the `add`, `subtract`, `multiply`, and `divide` function definitions one by one. Then, look for c
print("These options are typically found in a right-click context menu on the code, within an 'AI' or 'Code Actions' menu, or via a dedicated command palette.")
print("\n") # Add a newline for better readability

# 4. Explain that the AI tool will analyze the code and generate docstrings.
print("Once triggered, the AI tool will analyze the code. For the module, it will look at the overall content. For each function, it will analyze the function signature (name, p
print("The AI will then generate docstrings, often attempting to follow common styles like NumPy, Google, or reStructuredText, based on its training data or your tool's settings
print("\n") # Add a newline for better readability

# 5. Advise the user to carefully review the AI-generated docstrings.
print("After the AI generates the docstrings, it is crucial to carefully review them.")
print("Check for:")
print("- Accuracy: Does the docstring correctly describe what the code does?")
print("- Completeness: Does it include all parameters, return values, and exceptions?")
print("- Style: Does it adhere to the desired style (NumPy style in this case)?")
print("Make any necessary edits to ensure the docstrings are accurate, complete, and consistent with your project's documentation standards.")
```

```
To generate docstrings using an AI tool, you will need access to an AI-powered code editor (like VS Code with certain extensions, PyCharm, or online platforms that offer AI codin
```

Variables Terminal 8:58 PM Python 3

29°C Mostly clear Search ENG IN 20:58 18-09-2025

google colab - Search | Untitled24.ipynb - Colab | assignment 9.3

https://colab.research.google.com/drive/1-EMZSFjliYXgqEStmk9RggY3oCYPpDoi#scrollTo=5d6d987b

Untitled24.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[17]

```python
print("\n") # Add a newline for better readability

# 5. Advise the user to carefully review the AI-generated docstrings.
print("After the AI generates the docstrings, it is crucial to carefully review them.")
print("Check for:")
print("- Accuracy: Does the docstring correctly describe what the code does?")
print("- Completeness: Does it include all parameters, return values, and exceptions?")
print("- Style: Does it adhere to the desired style (NumPy style in this case)?")
print("Make any necessary edits to ensure the docstrings are accurate, complete, and consistent with your project's documentation standards.")
```

```
To generate docstrings using an AI tool, you will need access to an AI-powered code editor (like VS Code with certain extensions, PyCharm, or online platforms that offer AI codin


Open the Python script containing the `add`, `subtract`, `multiply`, and `divide` functions in your chosen development environment or AI tool.


To generate docstrings:
- For a module-level docstring, some tools might offer an option when the file is open or when selecting the entire file content. Look for options like 'Generate Module Docstring
- For individual function docstrings, select or place your cursor within the `add`, `subtract`, `multiply`, and `divide` function definitions one by one. Then, look for options l
These options are typically found in a right-click context menu on the code, within an 'AI' or 'Code Actions' menu, or via a dedicated command palette.


Once triggered, the AI tool will analyze the code. For the module, it will look at the overall content. For each function, it will analyze the function signature (name, parameter
The AI will then generate docstrings, often attempting to follow common styles like NumPy, Google, or reStructuredText, based on its training data or your tool's settings.


After the AI generates the docstrings, it is crucial to carefully review them.
Check for:
- Accuracy: Does the docstring correctly describe what the code does?
- Completeness: Does it include all parameters, return values, and exceptions?
- Style: Does it adhere to the desired style (NumPy style in this case)?
Make any necessary edits to ensure the docstrings are accurate, complete, and consistent with your project's documentation standards.
```

Variables Terminal 8:58 PM Python 3

29°C Mostly clear Search ENG IN 20:58 18-09-2025