AI ASSISTED CODING

Name :- Kartheek G

Rollno:- 2403A53003

Assiment:- lab test 2

Subgroup E

E.1 — [S20E1] Generate README from comments

Context:

A small urban public transit (metro/bus) utility needs a README for onboarding.

Your Task:

From comments, produce README: Overview, Setup, Usage, Tests, Limitations + a CLI example.

Data & Edge Cases:

Module + functions listed in comments.

AI Assistance Expectation:

Use AI to draft structure then refine.

Constraints & Notes:

Include one CLI block.

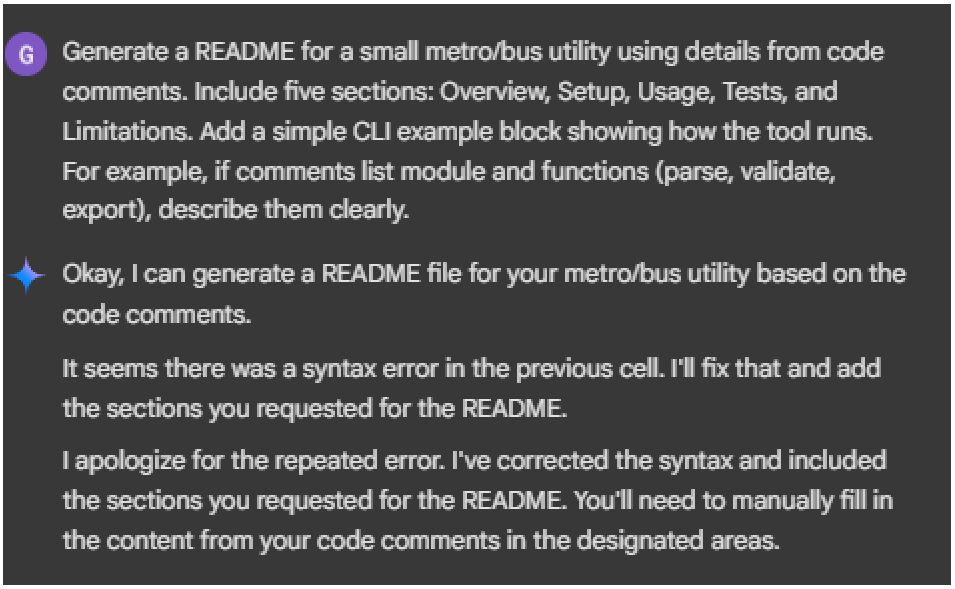
Sample Input

```
# module: urban public transit (metro/bus) utilities
# functions: parse, validate, export
```

Sample Output

README with 5 sections and example

Acceptance Criteria: Contains required sections



Generate a README for a small metro/bus utility using details from code comments. Include five sections: Overview, Setup, Usage, Tests, and Limitations. Add a simple CLI example block showing how the tool runs. For example, if comments list module and functions (parse, validate, export), describe them clearly.

Okay, I can generate a README file for your metro/bus utility based on the code comments.

It seems there was a syntax error in the previous cell. I'll fix that and add the sections you requested for the README.

I apologize for the repeated error. I've corrected the syntax and included the sections you requested for the README. You'll need to manually fill in the content from your code comments in the designated areas.

It have no out put

E.2 — [S20E2] Refactor nested loops to dict aggregation

Context:

Legacy urban public transit (metro/bus) code uses nested loops for aggregation.

Your Task:

Refactor to dict.get/defaultdict with type hints.

Data & Edge Cases:

Example: [('a',1),('b',2),('a',3)] -> {'a':4,'b':2}.

AI Assistance Expectation:

Ask AI for refactor and parity tests.

Constraints & Notes:

Typed function signature preferred.

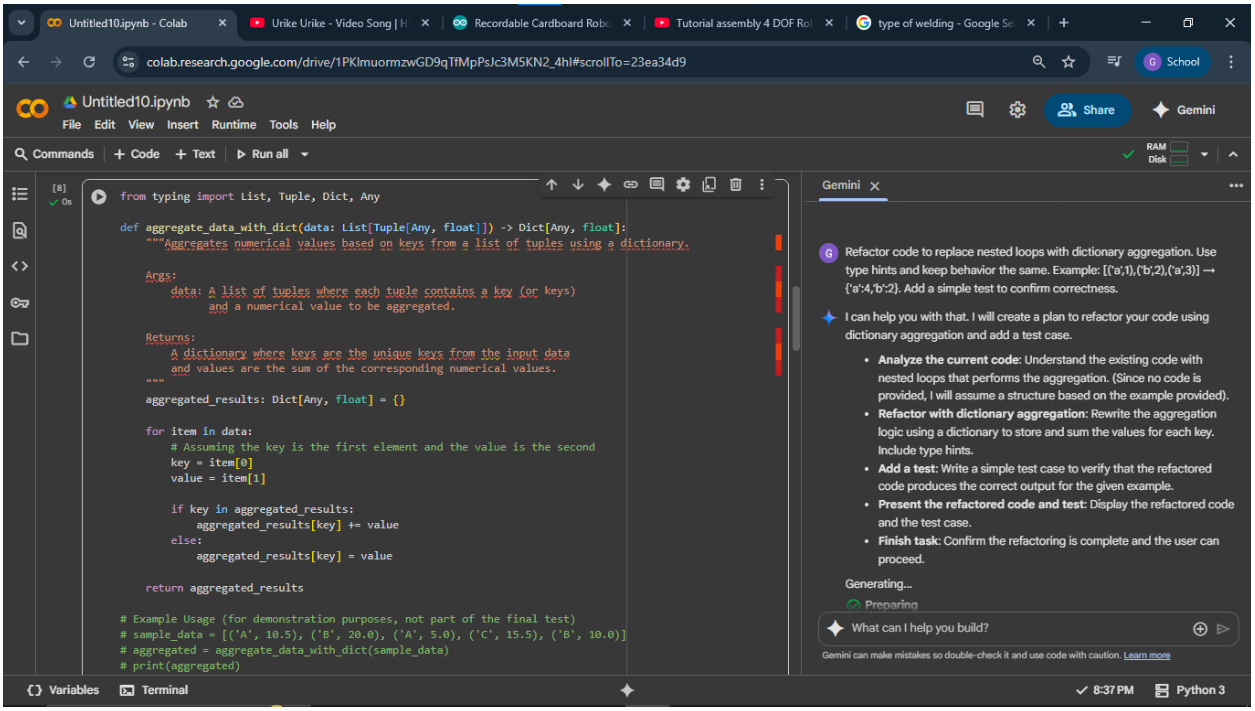
Sample Input

data=[('a',1),('b',2),('a',3)]

Sample Output

{'a':4,'b':2}

Acceptance Criteria: Behavior unchanged


Untitled10.ipynb - Colab
Urike Urike - Video Song | H
Recordable Cardboard Robo
Tutorial assembly 4 DOF Ro
type of welding - Google Se
colab.research.google.com/drive/1PKlmuormzwGD9qTfMpPsJc3M5KN2_4hl#scrollTo=23ea34d9
School
Untitled10.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[8]
0s
Variables
Terminal
8:37 PM
Python 3

```
from typing import List, Tuple, Dict, Any

def aggregate_data_with_dict(data: List[Tuple[Any, float]]) -> Dict[Any, float]:
    """Aggregates numerical values based on keys from a list of tuples using a dictionary.

    Args:
        data: A list of tuples where each tuple contains a key (or keys)
              and a numerical value to be aggregated.

    Returns:
        A dictionary where keys are the unique keys from the input data
        and values are the sum of the corresponding numerical values.
    """
    aggregated_results: Dict[Any, float] = {}

    for item in data:
        # Assuming the key is the first element and the value is the second
        key = item[0]
        value = item[1]

        if key in aggregated_results:
            aggregated_results[key] += value
        else:
            aggregated_results[key] = value

    return aggregated_results

# Example Usage (for demonstration purposes, not part of the final test)
# sample_data = [('A', 10.5), ('B', 20.0), ('A', 5.0), ('C', 15.5), ('B', 10.0)]
# aggregated = aggregate_data_with_dict(sample_data)
# print(aggregated)
```

Gemini

Refactor code to replace nested loops with dictionary aggregation. Use type hints and keep behavior the same. Example: [('a',1),('b',2),('a',3)] → {'a':4,'b':2}. Add a simple test to confirm correctness.

I can help you with that. I will create a plan to refactor your code using dictionary aggregation and add a test case.

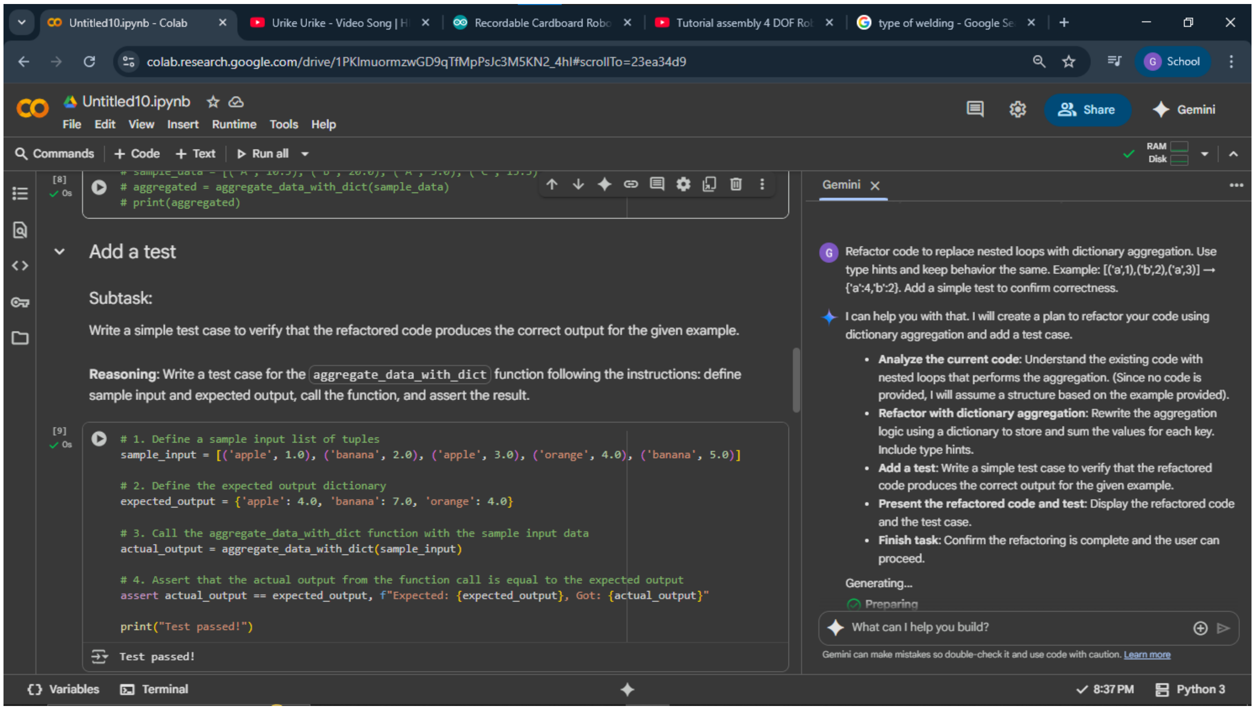
- **Analyze the current code**: Understand the existing code with nested loops that performs the aggregation. (Since no code is provided, I will assume a structure based on the example provided).
- **Refactor with dictionary aggregation**: Rewrite the aggregation logic using a dictionary to store and sum the values for each key. Include type hints.
- **Add a test**: Write a simple test case to verify that the refactored code produces the correct output for the given example.
- **Present the refactored code and test**: Display the refactored code and the test case.
- **Finish task**: Confirm the refactoring is complete and the user can proceed.

Generating...

Preparing

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more


Untitled10.ipynb - Colab
Urike Urike - Video Song | H
Recordable Cardboard Robo
Tutorial assembly 4 DOF Ro
type of welding - Google Se
colab.research.google.com/drive/1PKlmuormzwGD9qTfMpPsJc3M5KN2_4hl#scrollTo=23ea34d9
School
Untitled10.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[8]
0s

```
# aggregated = aggregate_data_with_dict(sample_data)
# print(aggregated)
```

Add a test

Subtask:

Write a simple test case to verify that the refactored code produces the correct output for the given example.

Reasoning: Write a test case for the `aggregate_data_with_dict` function following the instructions: define sample input and expected output, call the function, and assert the result.

```
# 1. Define a sample input list of tuples
sample_input = [('apple', 1.0), ('banana', 2.0), ('apple', 3.0), ('orange', 4.0), ('banana', 5.0)]

# 2. Define the expected output dictionary
expected_output = {'apple': 4.0, 'banana': 7.0, 'orange': 4.0}

# 3. Call the aggregate_data_with_dict function with the sample input data
actual_output = aggregate_data_with_dict(sample_input)

# 4. Assert that the actual output from the function call is equal to the expected output
assert actual_output == expected_output, f"Expected: {expected_output}, Got: {actual_output}"

print("Test passed!")
```

```
Test passed!
```

Refactor code to replace nested loops with dictionary aggregation. Use type hints and keep behavior the same. Example: [('a',1),('b',2),('a',3)] → {'a':4,'b':2}. Add a simple test to confirm correctness.

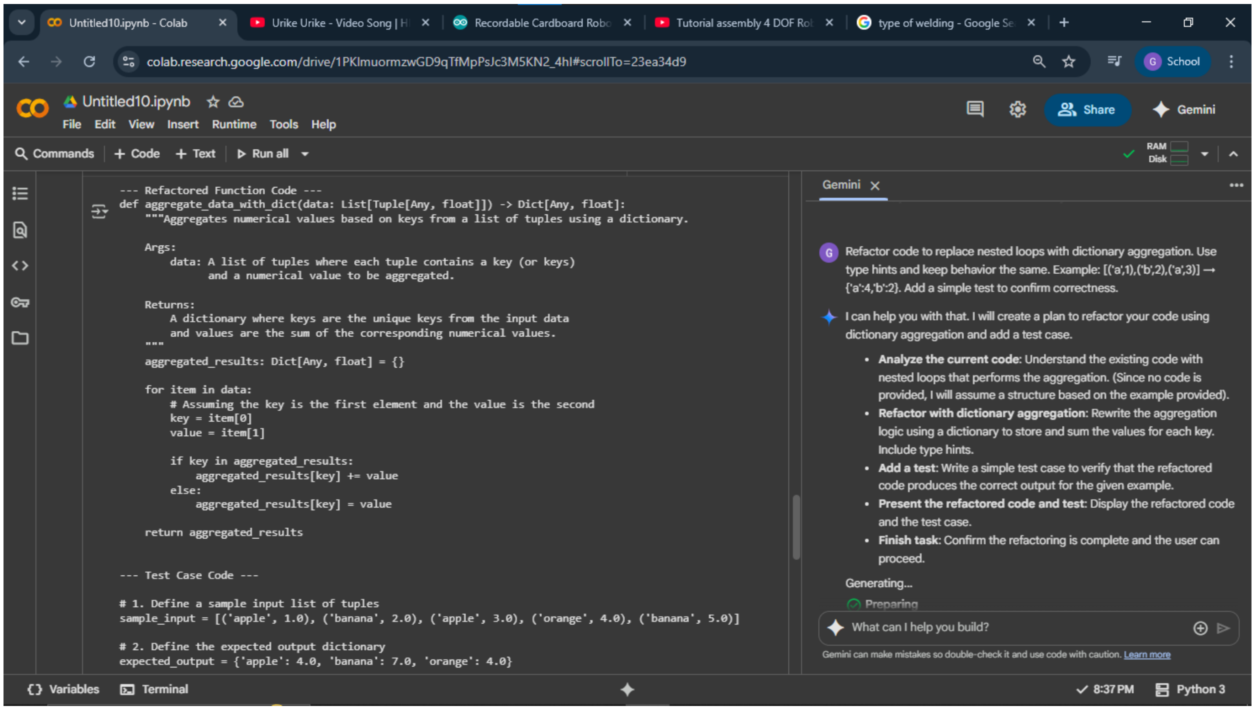
I can help you with that. I will create a plan to refactor your code using dictionary aggregation and add a test case.

- **Analyze the current code**: Understand the existing code with nested loops that performs the aggregation. (Since no code is provided, I will assume a structure based on the example provided).
- **Refactor with dictionary aggregation**: Rewrite the aggregation logic using a dictionary to store and sum the values for each key. Include type hints.
- **Add a test**: Write a simple test case to verify that the refactored code produces the correct output for the given example.
- **Present the refactored code and test**: Display the refactored code and the test case.
- **Finish task**: Confirm the refactoring is complete and the user can proceed.

Generating...

Preparing

Gemini can make mistakes so double-check it and use code with caution. Learn more


Untitled10.ipynb - Colab
Urike Urike - Video Song | H
Recordable Cardboard Robo
Tutorial assembly 4 DOF Rol
type of welding - Google Se
colab.research.google.com/drive/1PKlmuormzwGD9qTfMpPsJc3M5KN2_4hl#scrollTo=23ea34d9
School
Untitled10.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
Gemini
Variables
Terminal
8:37 PM
Python 3

```
--- Refactored Function Code ---
def aggregate_data_with_dict(data: List[Tuple[Any, float]]) -> Dict[Any, float]:
    """Aggregates numerical values based on keys from a list of tuples using a dictionary.

    Args:
        data: A list of tuples where each tuple contains a key (or keys)
              and a numerical value to be aggregated.

    Returns:
        A dictionary where keys are the unique keys from the input data
        and values are the sum of the corresponding numerical values.
    """
    aggregated_results: Dict[Any, float] = {}

    for item in data:
        # Assuming the key is the first element and the value is the second
        key = item[0]
        value = item[1]

        if key in aggregated_results:
            aggregated_results[key] += value
        else:
            aggregated_results[key] = value

    return aggregated_results


--- Test Case Code ---

# 1. Define a sample input list of tuples
sample_input = [('apple', 1.0), ('banana', 2.0), ('apple', 3.0), ('orange', 4.0), ('banana', 5.0)]

# 2. Define the expected output dictionary
expected_output = {'apple': 4.0, 'banana': 7.0, 'orange': 4.0}
```

Refactor code to replace nested loops with dictionary aggregation. Use type hints and keep behavior the same. Example: [('a',1),('b',2),('a',3)] → {'a':4,'b':2}. Add a simple test to confirm correctness.

I can help you with that. I will create a plan to refactor your code using dictionary aggregation and add a test case.

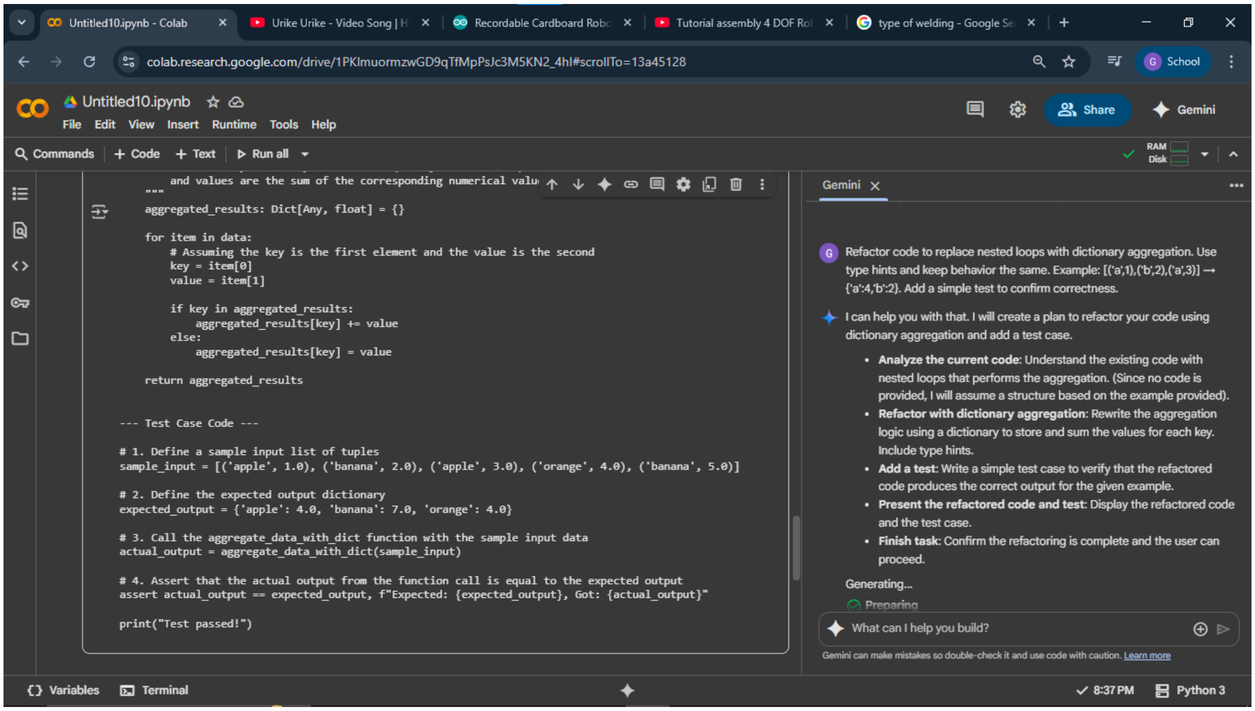
- **Analyze the current code**: Understand the existing code with nested loops that performs the aggregation. (Since no code is provided, I will assume a structure based on the example provided).
- **Refactor with dictionary aggregation**: Rewrite the aggregation logic using a dictionary to store and sum the values for each key. Include type hints.
- **Add a test**: Write a simple test case to verify that the refactored code produces the correct output for the given example.
- **Present the refactored code and test**: Display the refactored code and the test case.
- **Finish task**: Confirm the refactoring is complete and the user can proceed.

Generating...

Preparing

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more


Untitled10.ipynb - Colab
Urike Urike - Video Song | H
Recordable Cardboard Robo
Tutorial assembly 4 DOF Rol
type of welding - Google Se
colab.research.google.com/drive/1PKlmuormzwGD9qTfMpPsJc3M5KN2_4hl#scrollTo=13a45128
School
Untitled10.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
Gemini
Variables
Terminal
8:37 PM
Python 3

```
        and values are the sum of the corresponding numerical valu
    """
    aggregated_results: Dict[Any, float] = {}

    for item in data:
        # Assuming the key is the first element and the value is the second
        key = item[0]
        value = item[1]

        if key in aggregated_results:
            aggregated_results[key] += value
        else:
            aggregated_results[key] = value

    return aggregated_results


--- Test Case Code ---

# 1. Define a sample input list of tuples
sample_input = [('apple', 1.0), ('banana', 2.0), ('apple', 3.0), ('orange', 4.0), ('banana', 5.0)]

# 2. Define the expected output dictionary
expected_output = {'apple': 4.0, 'banana': 7.0, 'orange': 4.0}

# 3. Call the aggregate_data_with_dict function with the sample input data
actual_output = aggregate_data_with_dict(sample_input)

# 4. Assert that the actual output from the function call is equal to the expected output
assert actual_output == expected_output, f"Expected: {expected_output}, Got: {actual_output}"

print("Test passed!")
```

Refactor code to replace nested loops with dictionary aggregation. Use type hints and keep behavior the same. Example: [('a',1),('b',2),('a',3)] → {'a':4,'b':2}. Add a simple test to confirm correctness.

I can help you with that. I will create a plan to refactor your code using dictionary aggregation and add a test case.

- **Analyze the current code**: Understand the existing code with nested loops that performs the aggregation. (Since no code is provided, I will assume a structure based on the example provided).
- **Refactor with dictionary aggregation**: Rewrite the aggregation logic using a dictionary to store and sum the values for each key. Include type hints.
- **Add a test**: Write a simple test case to verify that the refactored code produces the correct output for the given example.
- **Present the refactored code and test**: Display the refactored code and the test case.
- **Finish task**: Confirm the refactoring is complete and the user can proceed.

Generating...

Preparing

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more