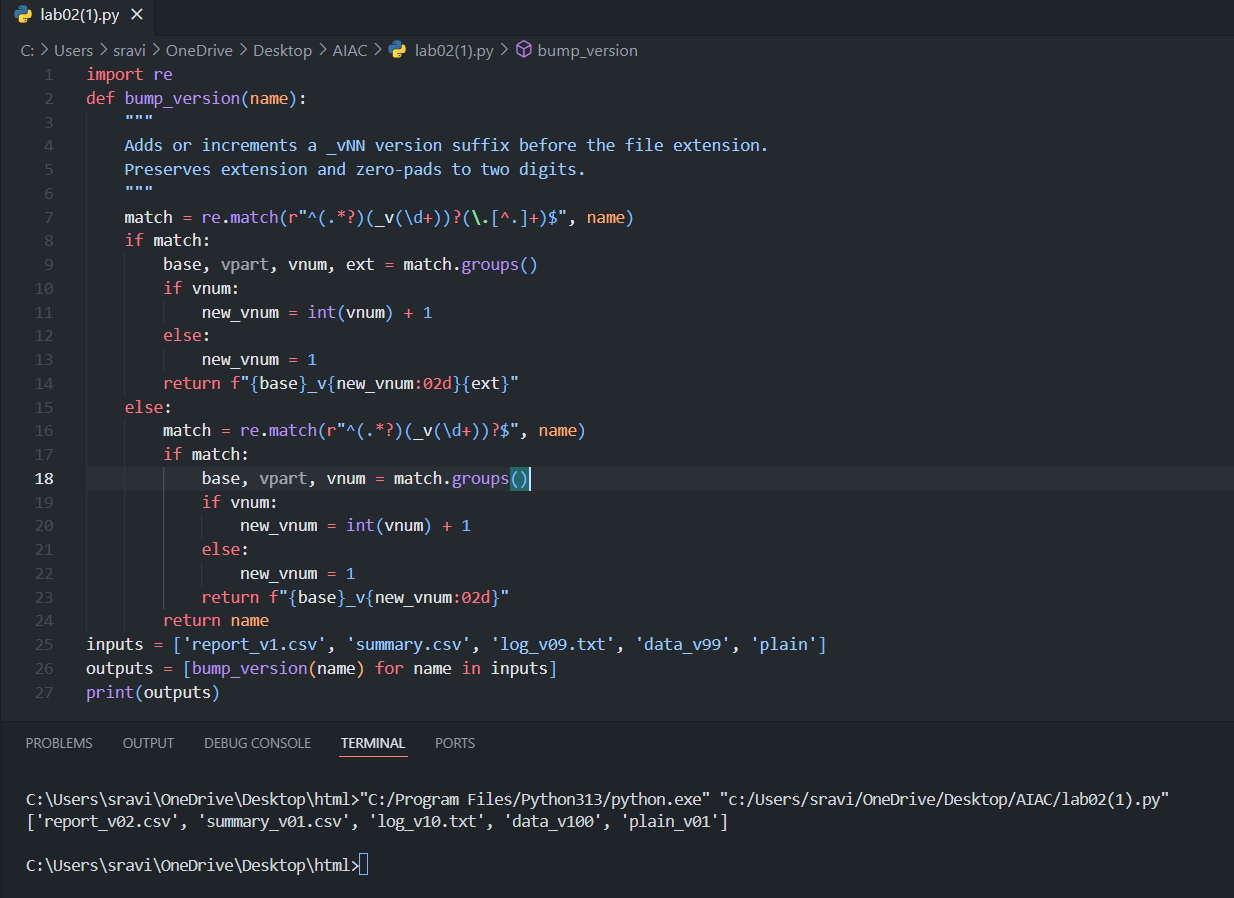
Lab test-02

Name: D. Sravika Reddy Hall-Ticket No: 2403a510d0

Batch No: 05 Course: AI Assisted Coding

**🡪D.1 — [S09D1] TDD: increment version suffix**

* Scenario (sports analytics):  
  Context:  
  File versioning in the sports analytics data pipeline uses a `\_vNN` suffix before the extension.  
  Your Task:  
  Create tests and implement bump\_version(name) that adds or increments `\_vNN` with zero-  
  padding.  
  Data & Edge Cases:  
  Handle names with and without existing suffix; preserve original extension.  
  AI Assistance Expectation:  
  Use AI to propose regex and test cases for edge names like `report\_v9.csv`, `summary.csv`.  
  Constraints & Notes:  
  Preserve original extension and base name.  
  Sample Input  
  ['report\_v1.csv', 'summary.csv', 'log\_v09.txt']  
  Sample Output  
  ['report\_v02.csv', 'summary\_v01.csv', 'log\_v10.txt']  
  Acceptance Criteria: Correct zero-padding; extension preserved
* **#Prompt:** Write a python function to add or increment a zero-padded \_vNN version suffix before the file extension in a filename.
* **Code and Output**:



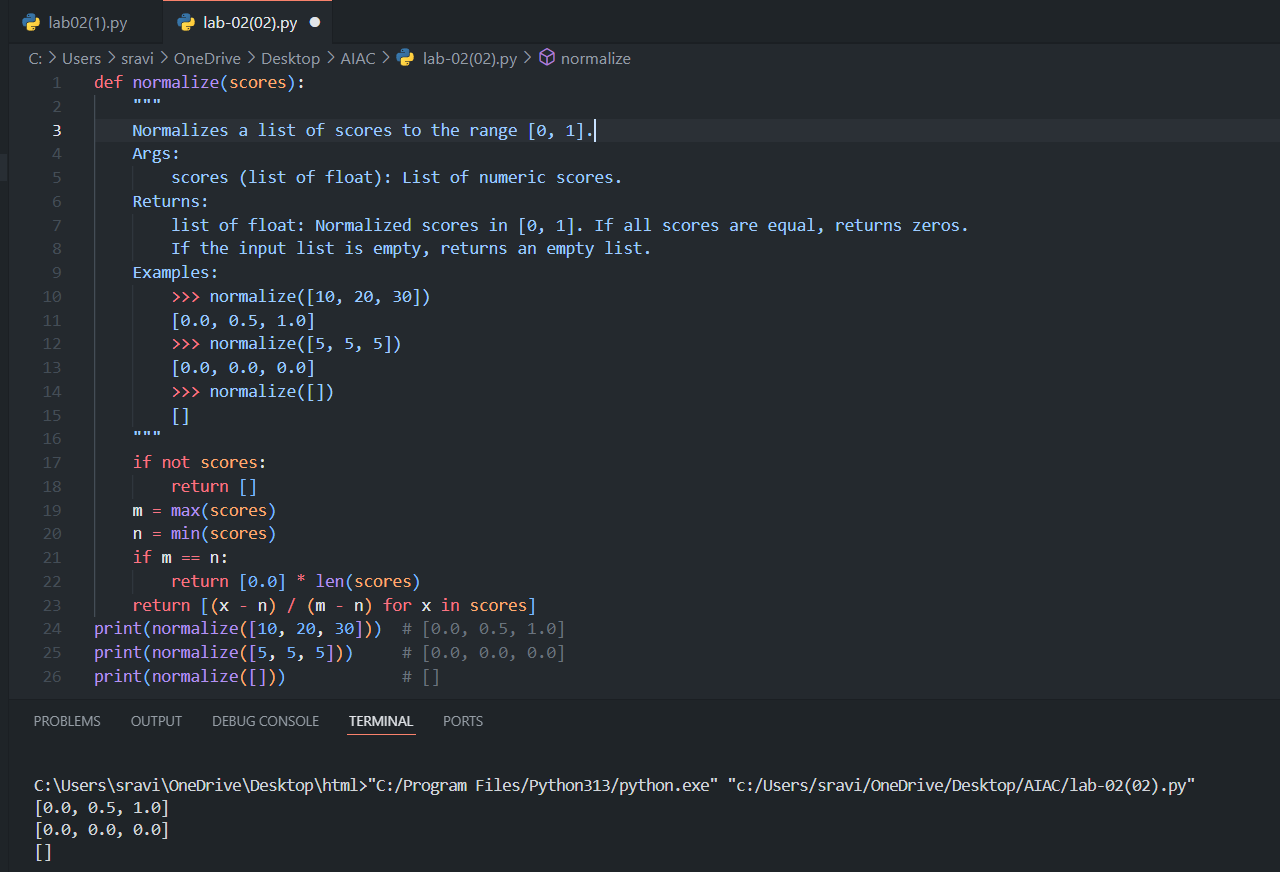
* **Obervations:**
* The function correctly increments the version if present, or adds \_v01 if missing.
* Zero-padding is applied for numbers less than 10.
* The file extension and base name are preserved.
* Handles edge cases like no extension or high version numbers.

**🡪D.2 — [S09D2] Generate docstrings and usage examples**

* Scenario (sports analytics):  
  Context:

Data analysts in sports analytics normalize metrics to [0,1] for comparability.  
Your Task:  
Add Google-style docstrings and handle the edge-case where all scores are equal (avoid divide-  
by-zero).  
Data & Edge Cases:  
Empty lists return empty; if max==min, return zeros of the same length.  
AI Assistance Expectation:  
Use AI to draft docstrings with Args/Returns/Examples and generate unit tests for edge-cases.  
Constraints & Notes:  
Add tests demonstrating the m==n case.  
Sample Input  
def normalize(scores):  
m = max(scores); n = min(scores)  
return [(x-n)/(m-n) for x in scores]  
Sample Output  
Docstring includes Args/Returns/Examples; guard for m==n  
Acceptance Criteria: Doc quality and guard confirmed by tests

* **#Prompt:** Write a Python function to normalize a list of scores to the range [0, 1], with Google-style docstrings and handling the case where all scores are equal.
* **Code & Output:**



* **Observation:**
* The function returns an empty list for empty input.
* If all scores are equal, it returns a list of zeros (avoids divide-by-zero).
* The Google-style docstring includes Args, Returns, and Examples.
* The normalization logic is correct and robust for edge cases