Lab test-02

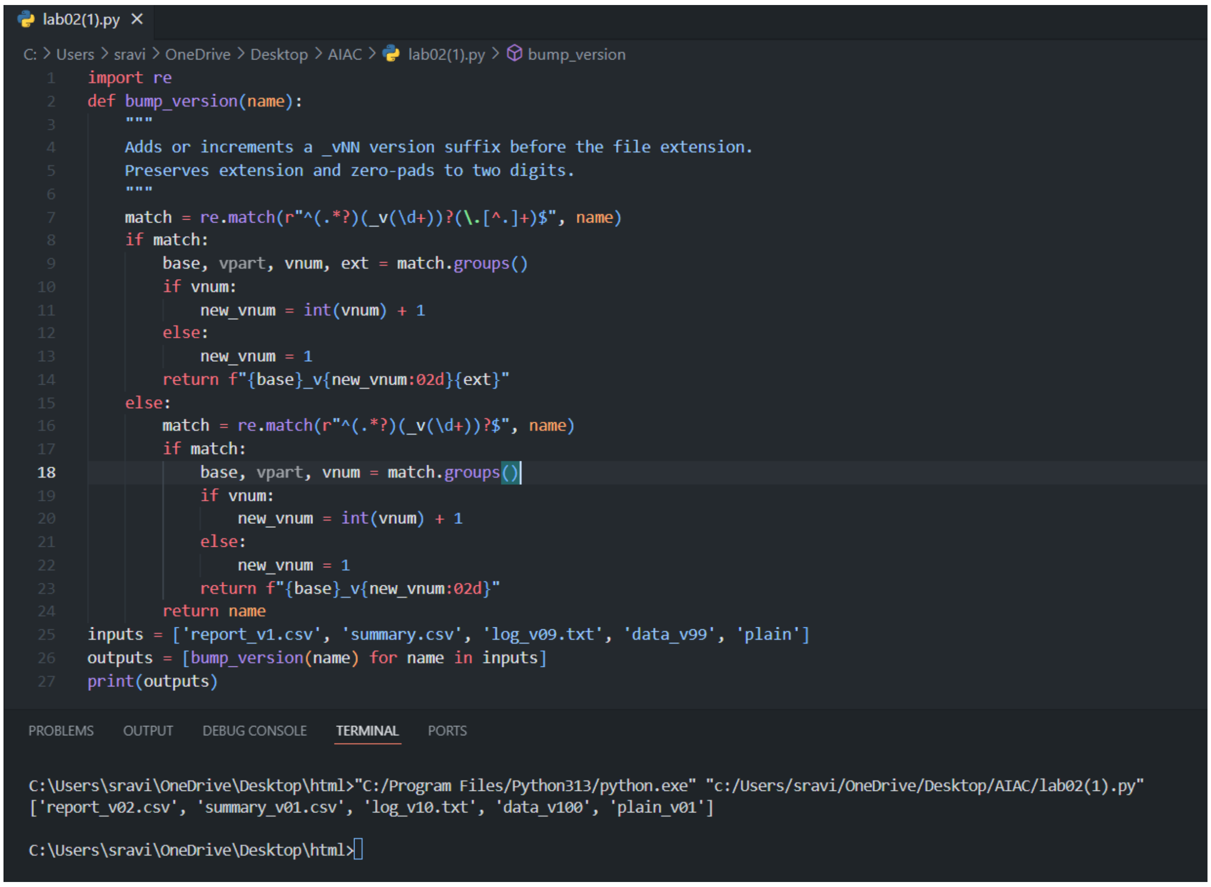
Name: D. Sravika Reddy	Hall-Ticket No: 2403a510d0

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**:


lab02(1).py
C: > Users > sravi > OneDrive > Desktop > AIAC > lab02(1).py > bump_version
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```python
import re
def bump_version(name):
    """
    Adds or increments a _vNN version suffix before the file extension.
    Preserves extension and zero-pads to two digits.
    """
    match = re.match(r"^(.*?)(_v(\d+))?(\.[^.]+)$", name)
    if match:
        base, vpart, vnum, ext = match.groups()
        if vnum:
            new_vnum = int(vnum) + 1
        else:
            new_vnum = 1
        return f"{base}_v{new_vnum:02d}{ext}"
    else:
        match = re.match(r"^(.*?)(_v(\d+))?$", name)
        if match:
            base, vpart, vnum = match.groups()
            if vnum:
                new_vnum = int(vnum) + 1
            else:
                new_vnum = 1
            return f"{base}_v{new_vnum:02d}"
        return name
inputs = ['report_v1.csv', 'summary.csv', 'log_v09.txt', 'data_v99', 'plain']
outputs = [bump_version(name) for name in inputs]
print(outputs)
```

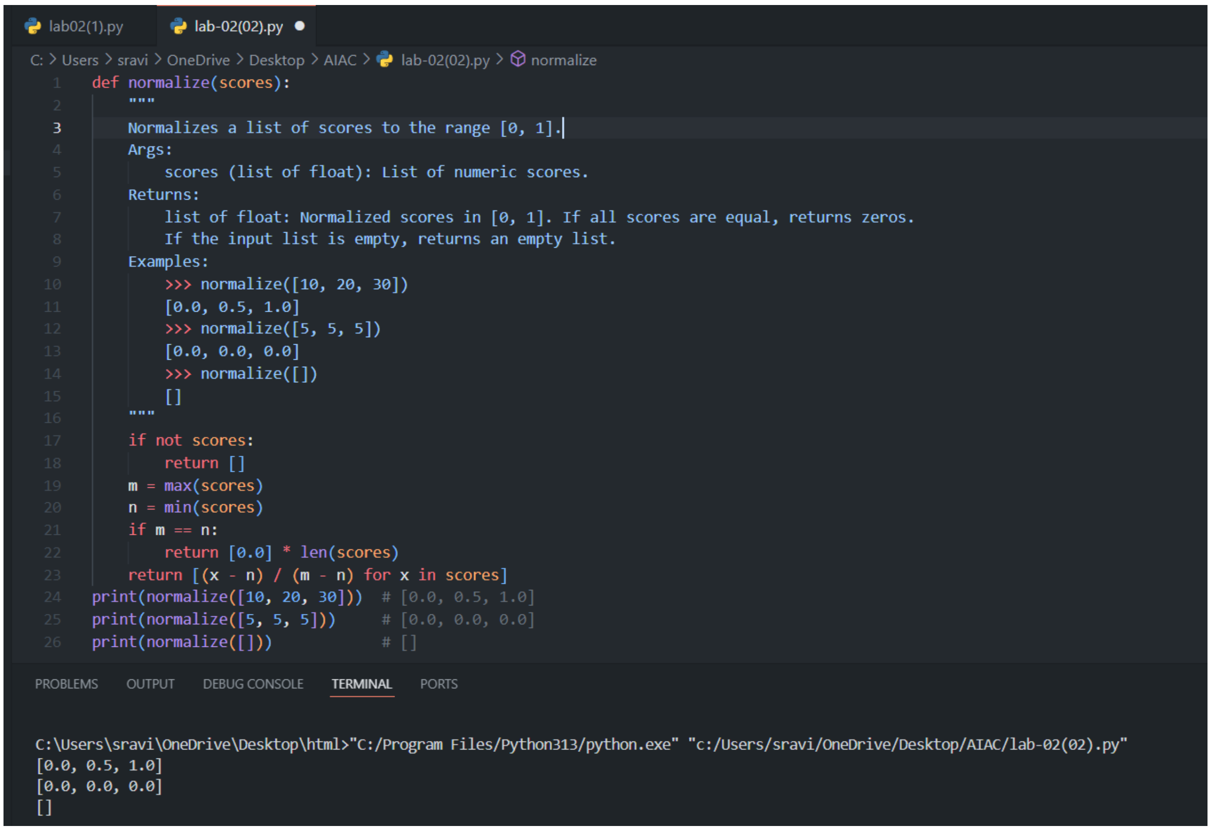
```
C:\Users\sravi\OneDrive\Desktop\html>"C:/Program Files/Python313/python.exe" "c:/Users/sravi/OneDrive/Desktop/AIAC/lab02(1).py"
['report_v02.csv', 'summary_v01.csv', 'log_v10.txt', 'data_v100', 'plain_v01']

C:\Users\sravi\OneDrive\Desktop\html>
```

- **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:**


lab02(1).py
lab-02(02).py
C: > Users > sravi > OneDrive > Desktop > AIAC > lab-02(02).py > normalize
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```python
def normalize(scores):
    """
    Normalizes a list of scores to the range [0, 1].
    Args:
        scores (list of float): List of numeric scores.
    Returns:
        list of float: Normalized scores in [0, 1]. If all scores are equal, returns zeros.
        If the input list is empty, returns an empty list.
    Examples:
        >>> normalize([10, 20, 30])
        [0.0, 0.5, 1.0]
        >>> normalize([5, 5, 5])
        [0.0, 0.0, 0.0]
        >>> normalize([])
        []
    """
    if not scores:
        return []
    m = max(scores)
    n = min(scores)
    if m == n:
        return [0.0] * len(scores)
    return [(x - n) / (m - n) for x in scores]
print(normalize([10, 20, 30]))  # [0.0, 0.5, 1.0]
print(normalize([5, 5, 5]))     # [0.0, 0.0, 0.0]
print(normalize([]))            # []
```

```
C:\Users\sravi\OneDrive\Desktop\html>"C:/Program Files/Python313/python.exe" "c:/Users/sravi/OneDrive/Desktop/AIAC/lab-02(02).py"
[0.0, 0.5, 1.0]
[0.0, 0.0, 0.0]
[]
```

- **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