System Requirements Specification Index

For

Normalizing Data using NumPy Broadcasting

(Topic: Advanced NumPy)

Version 1.0

Data Normalizer Console

Project Abstract
The Data Normalizer Console is a Python-based application that allows users to normalize student test scores. The primary goal of the application is to standardize scores by transforming the data, making it more consistent for analysis and comparison. By applying normalization techniques, the system adjusts scores based on their mean and standard deviation. This ensures that data with varying ranges or scales are brought to a comparable level, making further analysis more meaningful. The system guarantees that it handles empty or invalid data gracefully and can adjust for cases where data points have no variability (i.e., all scores are the same).

Business Requirements:

Constraints

Input Requirements

1. **Student Test Scores**
    - **Must be a numeric list or array of test scores.**
    - **The list must contain numeric values only (integers or floats).**
    - **Example: `[90, 85, 78, 92, 88]`**

Output Requirements

1. **Normalized Scores**
    - **Normalized data must be calculated by subtracting the mean and dividing by the standard deviation.**
    - **If the standard deviation is 0 (i.e., all values are identical), the output should be the same as the input scores.**
    - **Example output (for the given example input): `[0.35, -0.23, -1.25, 0.92, 0.21]`**

Conversion Constraints

Input Validation:

1. **Numeric Input**

- **Ensure input is numeric and can be converted to a `float`.**
- **Raise an error if input contains non-numeric values or is not convertible.**

Normalization Calculation:

1. **Normalization Logic**
   - **The formula for normalization is: Normalized Score=Score−MeanStandard Deviation\text{Normalized Score} = \frac{\text{Score} - \text{Mean}}{\text{Standard Deviation}}Normalized Score=Standard DeviationScore−Mean**
   - **If the standard deviation is 0 (i.e., all values are identical), return the original scores without modification.**

Required Output Format:

1. **Display Format**
   - **The normalized data should be displayed or returned as an array of normalized scores.**
   - **The normalization output should be consistent in terms of precision, formatted to match the input data.**
2. **Example:**
   - **Input: `[90, 85, 78, 92, 88]`**
   - **Output: `[-0.23, -0.92, -1.67, 0.58, -0.35]`**

Template Code Structure:

1. **Initialization Function:**
   - **`__init__(self, scores)`**
     - **Takes an array of scores and initializes the `self.scores` attribute.**
     - **Ensures the data is valid (numeric and convertible to float).**
2. **Normalization Function:**
   - **`normalize_data(self)`**
     - **Normalizes the data by subtracting the mean and dividing by the standard deviation.**
     - **If the standard deviation is 0, returns the original data unchanged.**
3. **Error Handling:**

- **Raises a `ValueError` if the input data is invalid (non-numeric or empty).**
- **Handles cases where the standard deviation is 0, avoiding division by zero errors.**

Execution Steps to Follow:

- All actions like build, compile, running application, running test cases will be through Command Terminal.
- To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
- This editor Auto Saves the code
- If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use **CTRL+Shift+B** -command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
- These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
- To setup environment:

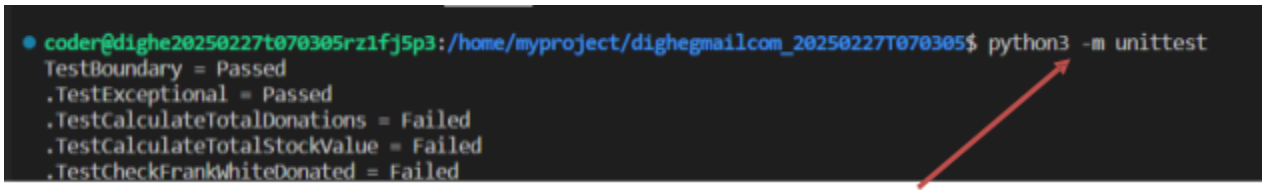
  You can run the application without importing any packages
- To launch application:
  python3 mainclass.py
  To run Test cases:
  python3 -m unittest

- Before Final Submission also, you need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

Screen shot to run the program

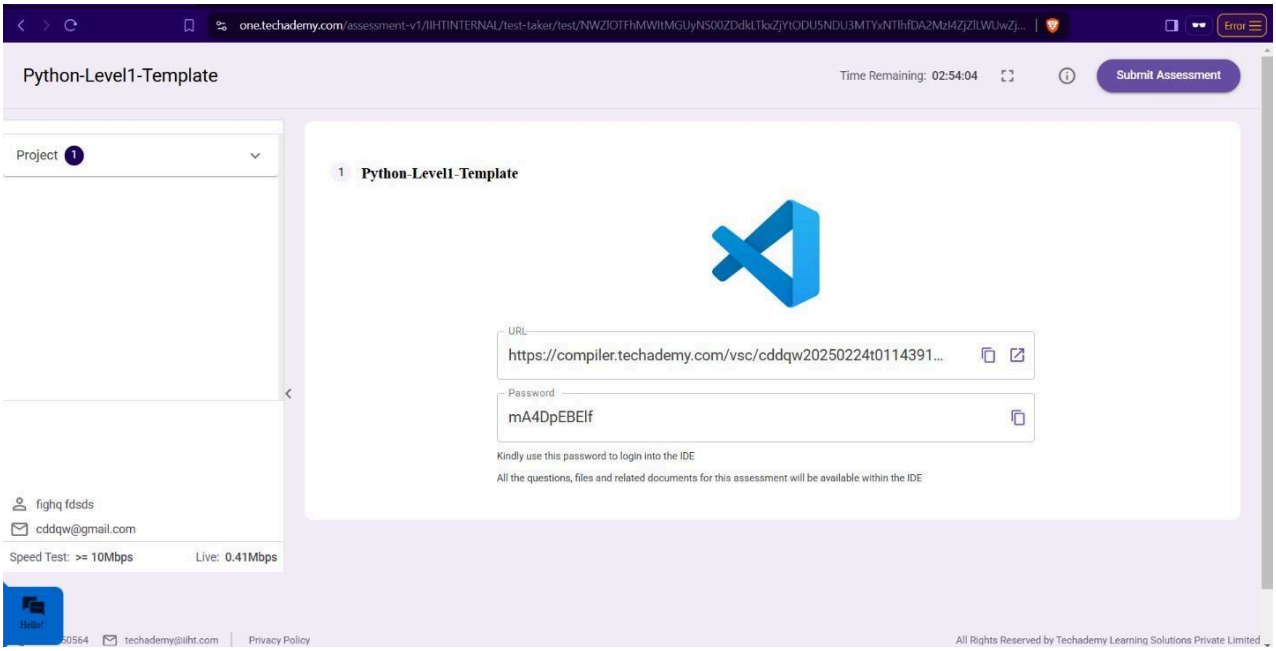
To run the application

python3 mainclass.py python3



To run the testcase

python3 -m unittest7



- **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.**