System Requirements Specification Index

For

Matrix Operations using NumPy

(Topic: Mathematical and Statistical Operations)

Version 1.0

Matrix Operations Console

Project Abstract

The Matrix Operations Console is a Python-based console application designed to handle various matrix operations, such as dot product, matrix multiplication, matrix transposition, and determinant calculation. The system uses NumPy arrays for efficient computation, and it is built to streamline mathematical operations involving matrices. The application ensures that all operations are performed correctly, validating matrix dimensions where necessary to prevent errors. This system is essential for users working with large data sets, scientific computations, or linear algebra problems, providing a simple and reliable interface for matrix-based computations.

Business Requirements:

- **Provide functionality for users to input two matrices and perform common operations like dot product, multiplication, transposition, and determinant calculation.**
- **Handle invalid operations, such as matrix dimension mismatches, by raising appropriate exceptions.**
- **Support matrix operations in a variety of use cases including academic, scientific, and engineering applications.**
- **Offer a user-friendly interface with clear error messages for invalid inputs.**

Constraints
Input Requirements
Matrix A:

- **Must be a 2D list or array, such as a list of lists.**
- **Example: `[[1, 2], [3, 4]]`**

Matrix B:

- **Must be a 2D list or array, such as a list of lists.**
- **Example: `[[5, 6], [7, 8]]`**

Matrix Operations Format:

- **Matrix A and Matrix B must have compatible dimensions for operations like multiplication and dot product.**
- **Example: If Matrix A is of size 2x3, Matrix B must be of size 3xN for multiplication to be valid.**

Operation Constraints
Dot Product:

- **The dot product is only valid when the number of columns of Matrix A matches the number of rows of Matrix B.**
- **Formula: `Dot Product = A . B`**

Matrix Multiplication:

- **Matrix multiplication is valid if the number of columns of Matrix A equals the number of rows of Matrix B.**
- **If this condition is violated, raise a `ValueError`.**
- **Formula: `Matrix C = A × B`**

Transpose:

- **Can transpose either Matrix A or Matrix B.**
- **Transposition is performed by swapping rows with columns.**

Determinant:

- **The determinant is calculated for square matrices only.**
- **If the matrix is not square, raise a `ValueError`.**

Output Constraints
1. Display Format:

- **For each operation, display the result in a clean format.**
- **Ensure that the output of matrix operations is properly formatted for readability.**

Required Output Format:

- **Dot Product: Display the result of the dot product operation.**

- **Matrix Multiplication: Show the resultant matrix after multiplication.**
- **Transpose: Show the transposed matrix A or B.**
- **Determinant: Display the calculated determinant for the selected matrix.**

Template Code Structure:
1. Matrix Operations Functions:

- `dot_product()`
- `matrix_multiplication()`
- `transpose_matrix()`
- `determinant()`

2. Input Section:

- **Get Matrix A as a list of lists (or 2D array).**
- **Get Matrix B as a list of lists (or 2D array).**

3. Operation Section:

- **Perform the selected matrix operation (dot product, multiplication, transpose, or determinant).**

4. Output Section:

- **Display the results of the operations in a readable format.**

Execution Steps to Follow:

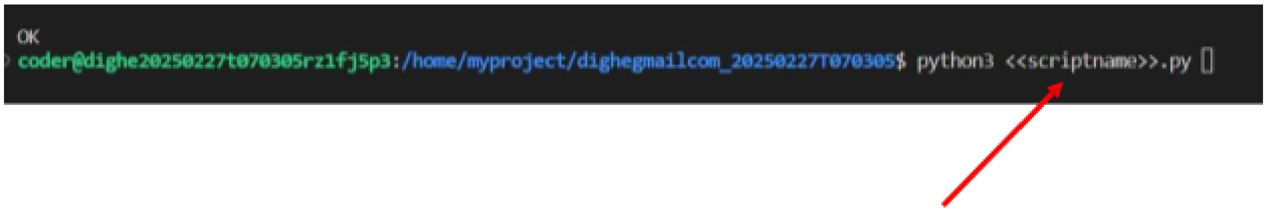
- All actions like build, compile, running application, running test cases will be through Command Terminal.
- To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
- This editor Auto Saves the code
- If you want 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.

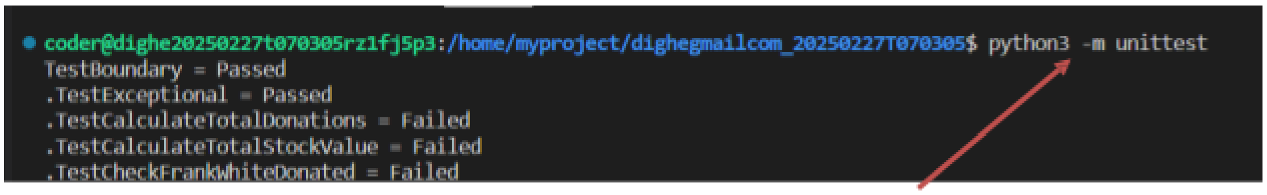
<u>**Screen shot to run the program**</u>

To run the application



```
OK
coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 <<scriptname>>.py
```

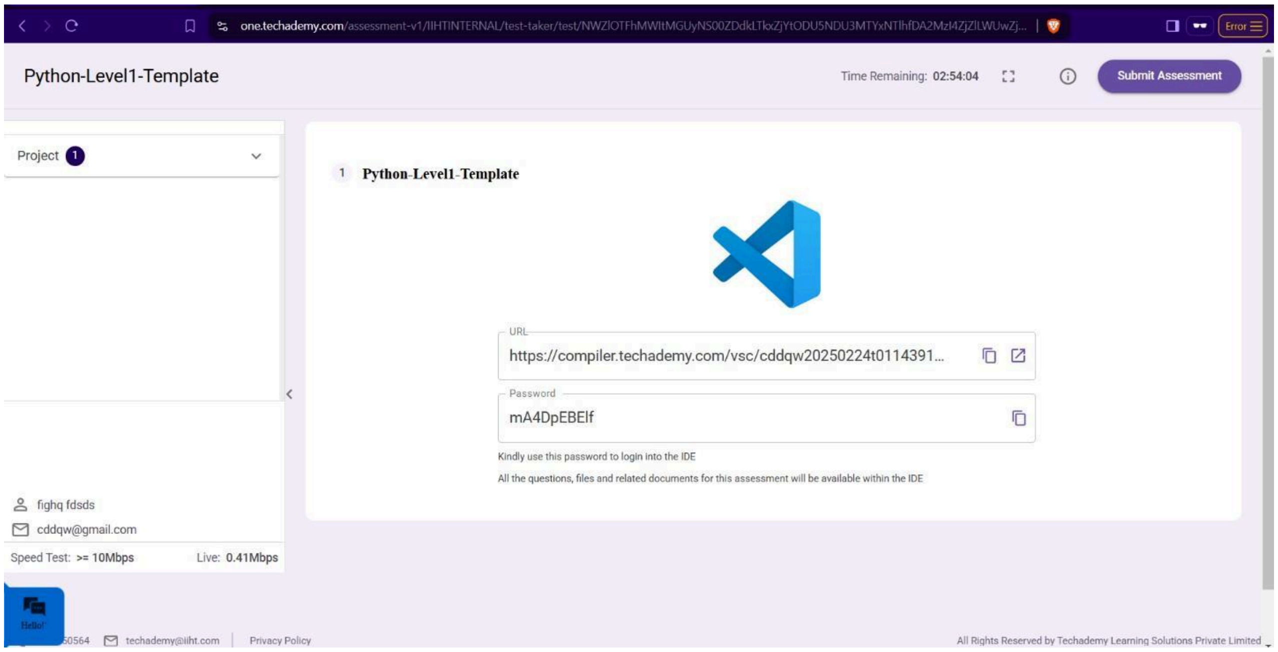
python3 mainclass.py python3



```
coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 -m unittest
  TestBoundary = Passed
  .TestExceptional = Passed
  .TestCalculateTotalDonations = Failed
  .TestCalculateTotalStockValue = Failed
  .TestCheckFrankWhiteDonated = Failed
```

To run the testcase

python3 -m unittest7


Python-Level1-Template

Time Remaining: 02:54:04

Submit Assessment

Project 1

1 Python-Level1-Template

URL
https://compiler.techademy.com/vsc/cddqw20250224t0114391...

Password
mA4DpEBElf

Kindly use this password to login into the IDE

All the questions, files and related documents for this assessment will be available within the IDE

fighq fdsds
cddqw@gmail.com
Speed Test: >= 10Mbps Live: 0.41Mbps

- **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.**