

# CS102L Data Structures & Algorithms

## Lab 2



DEPARTMENT OF COMPUTER SCIENCE  
DHANANI SCHOOL OF SCIENCE AND ENGINEERING  
HABIB UNIVERSITY  
SPRING 2024

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Restriction . . . . .	2
1.2	Instructions . . . . .	2
1.3	Marking scheme . . . . .	2
1.4	Lab Objectives . . . . .	2
1.5	Late submission policy . . . . .	3
1.6	Use of AI . . . . .	3
1.7	Viva . . . . .	3
<b>2</b>	<b>Prerequisites</b>	<b>4</b>
2.1	Array Initialization in python . . . . .	4
2.2	Assert Statements . . . . .	4
2.3	Pytest . . . . .	5
<b>3</b>	<b>Lab Exercises</b>	<b>6</b>
3.1	Matrix Initialization . . . . .	6
3.1.1	Function Description . . . . .	6
3.1.2	Sample . . . . .	6
3.1.3	Testing . . . . .	6
3.2	Matrix Subtraction . . . . .	6
3.2.1	Function Description . . . . .	6
3.2.2	Sample . . . . .	7
3.2.3	Testing . . . . .	7
3.3	Transpose of a Matrix . . . . .	7
3.3.1	Function Description . . . . .	7
3.3.2	Sample . . . . .	8
3.3.3	Testing . . . . .	8
3.4	Matrix Multiplication . . . . .	8
3.4.1	Function Description . . . . .	8
3.4.2	Sample . . . . .	9
3.4.3	Testing . . . . .	9
3.5	Image Reduction . . . . .	10
3.5.1	Function Description . . . . .	10
3.5.2	Sample . . . . .	10
3.5.3	Testing . . . . .	11

# Introduction

## 1.1 Restriction

You are not allowed to use `append`, `remove`, `pop`, or any similar function that changes the size of the list for this lab.

## 1.2 Instructions

- This lab will contribute 1% towards your final grade.
- The deadline for submission of this lab is at the end of lab time.
- The lab must be submitted online via CANVAS. You are required to submit a zip file that contains all the *.py* files.
- The zip file should be named as *Lab\_02\_aa1234.zip* where *aa1234* will be replaced with your student id.
- **Files that don't follow the appropriate naming convention will not be graded.**
- **Your final grade will comprise of both your submission and your lab performance.**

## 1.3 Marking scheme

This lab will be marked out of 100.

- 50 Marks are for the completion of the lab.
- 50 Marks are for progress and attendance during the lab.

## 1.4 Lab Objectives

- To understand what nested lists are

- To understand how to represent matrices in Python using nested lists
- To use nested lists to perform image operations on different images

## **1.5 Late submission policy**

There is no late submission policy for the lab.

## **1.6 Use of AI**

Taking help from any AI-based tools such as ChatGPT is strictly prohibited and will be considered plagiarism.

## **1.7 Viva**

Course staff may call any student for Viva to provide an explanation for their submission.

# Prerequisites

## 2.1 Array Initialization in python

Arrays are not built-in data structure inside python. However, lists can be simulated as arrays.

For example:

Initializing 1D Arrays in Python: To initialize a 1D array of 5 elements, we can do this:

```
arr = [0 for i in range(5)]
```

This initialized an array with 0's. Similarly, to initialize 2D arrays, we can do this:

```
arr = [[0 for i in range(5)] for j in range(5)]
```

Please visit any of the following links to understand more about this problem:

- [https://en.wikibooks.org/wiki/Python\\_Programming/Lists#List\\_creation\\_shortcuts](https://en.wikibooks.org/wiki/Python_Programming/Lists#List_creation_shortcuts)
- <https://www.geeksforgeeks.org/python-using-2d-arrays-lists-the-right-way/>

## 2.2 Assert Statements

Assert statements are crucial tools in programming used to verify assumptions and test conditions within code. For instance, in Python, an assert statement is typically as follows:

```
assert condition, message
```

For example, in a function calculating the square root, in order to ensure that the input value is non-negative, one might add the following assert statement:

```
assert x >= 0, "x should be a non-negative number"
```

If the condition evaluates to False, an AssertionError is raised, highlighting the issue and its corresponding message.

## 2.3 Pytest

Pytest is a popular testing framework in Python known for its simplicity and power in facilitating efficient and scalable testing for applications and software. In order to use Pytest, one first needs to install it and create test functions prefixed with *test\_* in a Python file. To install Pytest, you can write the following command on the terminal:

```
pip install pytest
```

After installing Pytest, one can execute all the tests by typing the following command on the terminal:

```
pytest
```

If you want to test any particular file, then you will have to specify the filename as follows:

```
pytest <filename>.py
```

If all the tests pass successfully, then you should receive a similar output.

```
PS C:\Work\Spring2024\DSA\Lab1> pytest test_Question1.py
===== test session starts =====
platform win32 -- Python 3.11.2, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Work\Spring2024\DSA\Lab1
collected 10 items

test_Question1.py .....
===== 10 passed in 0.06s =====
```

However, if all the tests fail, then you should receive a similar output.

```
PS C:\Work\Spring2024\DSA\Lab1> pytest test_Question1.py
===== test session starts =====
platform win32 -- Python 3.11.2, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Work\Spring2024\DSA\Lab1
collected 10 items

test_Question1.py FFFFFFFF
===== FAILURES =====
----- test_question1[1st10-1st20-e54dc15ccafa608142ffa6340b035c327f972e24882052cfc3345f240f4ee237] -----
```

# Lab Exercises

## 3.1 Matrix Initialization

### 3.1.1 Function Description

Write a function named `initialize_matrix` that takes 2 parameters `row` and `column`, and returns a *matrix* (nested list) with all values *initialized with 0*. You can refer to your lecture notes and prerequisites in this PDF for the implementation of this.

### 3.1.2 Sample

```
>>> initialize_matrix(4, 5)
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0,
  0, 0, 0]]
```

### 3.1.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question1.py
```

## 3.2 Matrix Subtraction

### 3.2.1 Function Description

Write a function, `matrix_subtraction`, that takes two integer matrices, `A` and `B`, and returns a matrix whose entries are the *subtraction* of the corresponding entries in matrix `B` from `A`. Utilize your `initialize_matrix` function for initializing the matrix.

Function should be implemented in the file `Question2.py`, accessible within the Lab 02 module on CANVAS.

$$\begin{bmatrix} 8 & 5 \\ 2 & 3 \end{bmatrix} - \begin{bmatrix} 9 & 5 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix}$$

### 3.2.2 Sample

```
>>> matrix_subtraction(
    [[1,2,3],[4,5,6],[7,8,9]], [[9,8,7],[6,5,4],[3,2,1]] )
[[-8, -6, -4],[-2, 0, 2],[4, 6, 8]]

>>> matrix_subtraction
    ([[12,7,3],[4,5,6],[7,8,9]], [[5,8,1],[6,7,3],[4,5,9]] )
[[7,-1,2],[-2,-2,3],[3,3,0]]

>>> matrix_subtraction([[1],[1],[1]],[[2],[2],[4]])
[[-1],[-1],[-3]]

>> matrix_subtraction([[1],[2]],[[3,5],[4,6]])
"Matrices A and B don't have the same dimension required for
matrix subtraction."
```

### 3.2.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question2.py
```

## 3.3 Transpose of a Matrix

The transpose of a matrix is a new matrix whose rows are the columns of the original.

### 3.3.1 Function Description

Write a function, `matrix.transpose`, that takes an integer matrix `A`, and returns its *transpose*. Utilize your `initialize_matrix` function from *Question1.py* for initializing the matrix.

This function should be implemented in the file `Question3.py`, accessible within the Lab 02 module on CANVAS.



$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}_{2 \times 3} \quad A^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}_{3 \times 2}$$

### 3.3.2 Sample

```
>>>matrix_transpose([[12,7],[4 ,5],[3 ,8]])
[[12, 4, 3],[7, 5, 8]]

>>>matrix_transpose([[12, 4, 3],[7, 5, 8]])
[[12,7],[4 ,5],[3 ,8]]
```

### 3.3.3 Testing

In order to test your function, type the following command on the terminal:

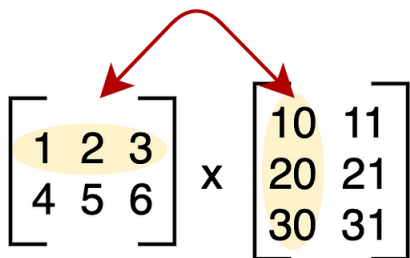
```
pytest test_Question3.py
```

## 3.4 Matrix Multiplication

### 3.4.1 Function Description

Write a function, `matrix_multiplication`, that takes two integer matrices, `A` and `B`, and returns their *dot product*. Utilize your `initialize_matrix` function from `Question1.py` for initializing the matrix.

This function should be implemented in the file `Question4.py`, accessible within the Lab 02 module on CANVAS.



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 10 & 11 \\ 20 & 21 \\ 30 & 31 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \times 10 + 2 \times 20 + 3 \times 30 & 1 \times 11 + 2 \times 21 + 3 \times 31 \\ 4 \times 10 + 5 \times 20 + 6 \times 30 & 4 \times 11 + 5 \times 21 + 6 \times 31 \end{bmatrix}$$

$$= \begin{bmatrix} 10+40+90 & 11+42+93 \\ 40+100+180 & 44+105+186 \end{bmatrix} = \begin{bmatrix} 140 & 146 \\ 320 & 335 \end{bmatrix}$$

Reference: <https://www.mathsisfun.com/algebra/matrix-multiplying.html>

### 3.4.2 Sample

```

>>> matrix_multiplication([[12,7,3],[4 ,5,6],[7 ,8,9]],
    [[5,8,1,2],[6,7,3,0], [4,5,9,1]])
[[114, 160, 60, 27],[74, 97, 73, 14],[119, 157, 112, 23]]

>>> matrix_multiplication([[34,1,77],[2,14,8],[3 ,17,11]],
    [[6,8,1],[9,27,5],[2,43,31]])
[[367, 3610, 2426], [154, 738, 320], [193, 956, 429]]

>>> matrix_multiplication([[1,2,3],[4,5,6]],
    [[7,8],[9,10],[11,12]])
[[58, 64], [139, 154]]

>>> matrix_multiplication([[7,3], [2,5], [6,8], [9,0]],
    [[8,14,0,3,-1], [7,11,5,91,3], [8,-4,19,5, 57]])
"The number of columns in Matrix A does not equal the number
of rows in Matrix B required for Matrix Multiplication."

```

### 3.4.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question4.py
```

## 3.5 Image Reduction

An image can be reduced by taking the **sum** of the neighborhood (**up, down, left, right, top-left, top-right, bottom-left, bottom-right**) and **multiplying it with pixel value** and then taking its **cube root**. The resultant pixel value would be the new value of the pixel.

### 3.5.1 Function Description

Write a function **reduce\_image()** that takes as parameter an image in the form of a nested list **A** and **reduces** it. Utilize your **initialize\_matrix** function from **Question1.py** for initializing the matrix.

This function should be implemented in the file **Question5.py**, accessible within the Lab 02 module on CANVAS.

### 3.5.2 Sample

**Input:**

<b>10</b>	<b>20</b>	<b>30</b>
<b>30</b>	<b>10</b>	<b>20</b>
<b>20</b>	<b>0</b>	<b>30</b>

**Output:**

8.434	12.599	11.447
12.164	11.696	12.164
9.283	0	9.655

**Explanation:**

$$Result_{00} = \text{cube\_root} ( (A_{10} + A_{01} + A_{11}) * A_{00} )$$

$$Result_{00} = \text{cube\_root} ( 60 * 10 )$$

$$Result_{00} = 8.434$$

$$Result_{11} = \text{cube\_root} ( (A_{10} + A_{01} + A_{12} + A_{21} + A_{00} + A_{02} + A_{20} + A_{22}) * A_{11} )$$

$$Result_{11} = \text{cube\_root} ( 160 * 10 )$$

$$Result_{11} = 11.696$$

**Note:** Not all of the neighbors are available in boundary cases. You have to write suitable conditions accordingly. All pixel values are rounded off to three decimal places.

### 3.5.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question5.py
```