Inspiring Excellence

Course Title: Programming Language I

Course Code: CSE 110

Assignment no: 7

Total tasks	11
Total marks	11 x 5 = 55

Before starting the lab tasks, hopefully you have the idea that Python does not have built-in support for Arrays. That means unlike Lists or Strings, Arrays are not predefined in Python. For this reason, we are using the NumPy module in this lab to perform the Array tasks.

To import NumPy module, write the following line at the top of your code cell:

```
import numpy as np # you can also give any name other than np
```

In Google Colab, the NumPy module is already installed. If you are using any other platform and the above line gives an error, then go through this link and install the module following the instructions.

Now solve the following tasks:

Part 1: Linear Array

Task 1

Take an integer N as an input from the user and create an 1D array with size N by taking each integer element as user input. Then, print the array. Next, take another integer input from the user. Resize the array and add the new integer value in the new array. Lastly, print the new array.

Hint: As you know, the size of an array is fixed and can not be changed. So, to resize an array, copy the elements of the old array to a new array of length N+1 ***using a loop*** and then add the new value to the new array.

Sample Input	Sample Output	Explanation
3 5 8 2 9	Original Array: [5 8 2] Resized Array: [5 8 2 9]	Here, 3 is the length of the first array. Next 3 inputs (5, 8, 2) denote the elements of the array. Last input 9 is the new element which we want to add in the array.
4 7 1 3 9 5	Original Array: [7 1 3 9] Resized Array: [7 1 3 9 5]	Here, 4 is the length of the first array. Next 4 inputs (7, 1, 3, 9) denote the elements of the array. Last input 5 is the new element which we want to add in the array.

Task 2

Take an integer N input from the user and create a 1D integer array of N numbers. Then, print the array. Next, modify the array by changing the positive numbers by 1 and the negative numbers by 0. If the element is zero, then it will be unchanged. Lastly, print the modified array.

Sample Input	Sample Output	Explanation
4 3 4 -2 1	Original array: [3 4 -2 1] After modifying: [1 1 0 1]	Here, the first input 4 denotes the size of the array. Then, using the next 4 inputs an array is created and printed. Then the elements of the array are modified using the above mentioned criteria.
3 -4 0 2	Original array: [-4 0 2] After modifying: [0 0 1]	Here, the first input 3 denotes the size of the array. Then, using the next 3 inputs an array is created and printed. Then the elements of the array are modified using the above mentioned criteria.

Task 3

Write a Python function named **reverseArray** that takes a NumPy Array as parameter and ***using loop*** modifies & returns the reversed array **within the given array**. That means inside the function you can not create a new array.

Function Call	Sample Output
arr1 = np.array([10, 12, 20, 5, 7]) arr1 = reverseArray(arr1) print(arr1)	[7 5 20 12 10]
arr2 = np.array([4, 2, 6, 5]) arr2 = reverseArray(arr2) print(arr2)	[5 6 2 4]

Task 4

Write a Python function named **printPairs** that takes an array of numbers and a single integer value as parameters and prints the pairs of two array elements ***inside the function*** whose sum is equal to that integer value. If no such pair is found, then print "No Pair Found".

Function Call	Sample Output	Explanation
arr1 = np.array([7,8,10,5,3,4,2]) printPairs(arr1, 15)	7, 8 10, 5	If we take summation of all possible pairs of two numbers from the array, we can see only two pairs make a sum that is 15. 7 + 8 = 15 10 + 5 = 15
arr2 = np.array([2,-3,1,9,4,5]) printPairs(arr2, 6)	2, 4 -3, 9 1, 5	If we take summation of all possible pairs of two numbers from the array, we can see three pairs make a sum that is 6. 2 + 4 = 6 -3 + 9 = 6 1 + 5 = 6
arr3 = np.array([5, 9, 7, 6, 10]) printPairs(arr3, 18)	No Pair Found	If we take summation of all the possible pairs of two numbers, then we get no pair where the sum is 18.

Task 5

Suppose, you are *given* two 1D NumPy arrays (vectors). Now, ***using a loop***, find the dot product of these two arrays and print the result. If the result is even, then swap the elements in the even index of the arrays. If the result is odd, then swap the elements in the odd index of the arrays. [You may assume that the size of both the arrays will be the same. Your code should work for 1D arrays of any size. You can not use in-built dot(), sum() or @ to solve this problem.]

The dot product of two vectors is defined by the summation of element wise multiplication of those two vectors.

Given Arrays	Output	Explanation
vec1 = np.array([1,2,3]) vec2 = np.array([4,5,6])	Dot Product: 32 After Swapping: [4 2 6] [1 5 3]	The dot product will be: (1*4) + (2*5) + (3*6) = 32 Since the total is even, we have to swap the elements in the even index. So. 1 and 4 from index 0 of the arrays are swapped. Similarly, 3 and 6 are swapped in index 2.
vec1 = np.array([1,2,3,7]) vec2 = np.array([4,5,6,9])	Dot Product: 95 After Swapping: [1 5 3 9] [4 2 6 7]	The dot product will be: (1*4) + (2*5) + (3*6) + (7*9) = 95 Since the total is odd, we have to swap the elements in the odd index. So. 2 and 5 from index 1 of the arrays are swapped. Similarly, 7 and 9 from index 3 are swapped.

Task 6

Take an integer N input from the user and create an 1D array of N numbers taking the elements as user input too. Then, print the array. Next, sort the array in the **ascending** order using **Selection Sort** technique. Lastly, print the sorted array.

[You are not allowed to use any built in functions]

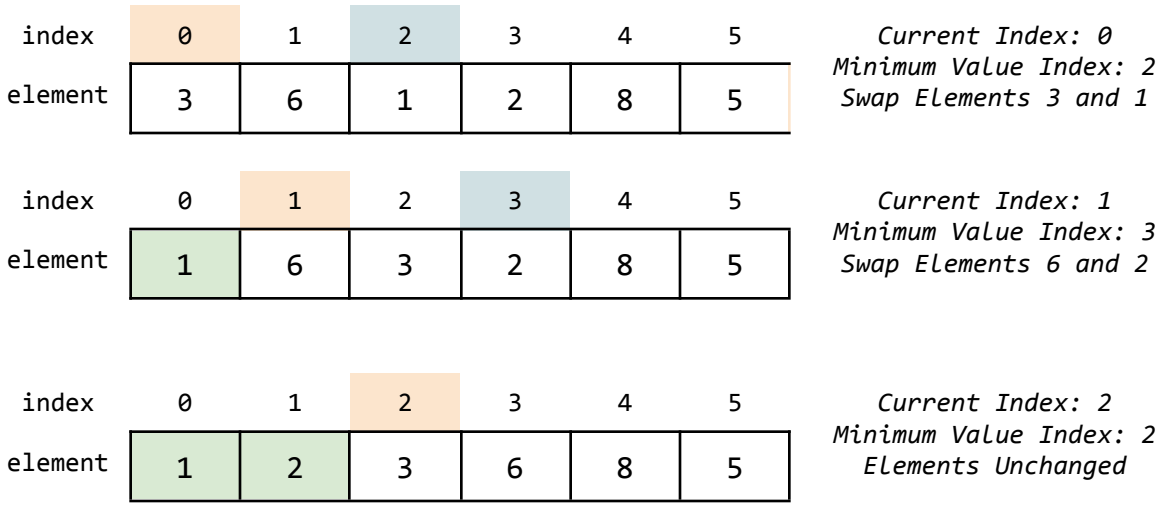
The steps of Selection Sort are given below:

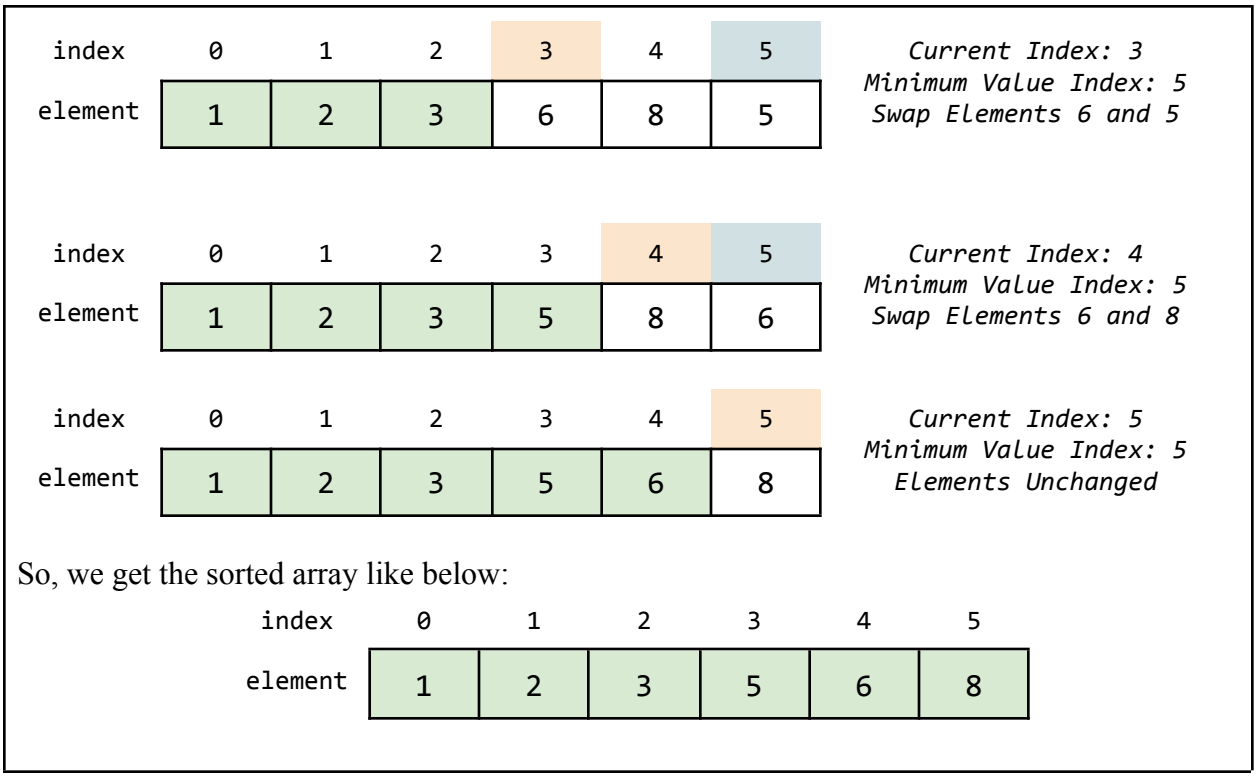
1. At the beginning of each iteration, assume that the current index of the array has the minimum value.
2. Search from (current index + 1) up to (N - 1) for the index with the minimum value.
3. Swap the value of the current index with the value of the index with the minimum value.
4. Repeat Steps 1-3 up to the end of the array. Therefore, starting from the left-most index, the values from each index get sorted one by one after each iteration.

Sample Input	Sample Output
6 3 6 1 2 8 5	Original Array: [3 6 1 2 8 5] Sorted Array: [1 2 3 5 6 8]

Explanation:

Here, N = 6. Then the next 6 values denote the elements of the Array. Now, let's see how Selection Sort works.





So, we get the sorted array like below:

Part 2: Multidimensional Array

Task 7

Write a Python function called **flatten** that takes a 2D array as parameter and returns an 1D array by flattening the 2D array.

Function Call	Sample Output
arr1 = np.array([[1, 2, 3], [3, 4, 5]]) arr2 = flatten(arr1) print(arr2)	[1 2 3 3 4 5]
arr3 = np.array([[1, 4], [5, 6], [8, 9]]) arr4 = flatten(arr3) print(arr4)	[1 4 5 6 8 9]

Task 8

You are given a square matrix **A** of size N×N. Check whether the given matrix is an Identity matrix or not. If it is, then print "Identity matrix" or otherwise print "Not an Identity matrix". Your program should work for any given 2D Arrays of size N×N.

[You may need to use the concept of flag and break to solve this problem.]

Identity Matrix is a square matrix with 1's along the diagonal from upper left to lower right and 0's in all other positions.

Given Array	Output
A = np.array([[1, 0, 0, 1], [0, 1, 0, 0], [1, 0, 1, 0], [0, 1, 0, 1]])	Not an Identity Matrix
A = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])	Identity Matrix

Task 9

You are given a 2D array **A** of size N×N. Print the absolute difference between the summation of its two diagonals (primary and secondary diagonal). Your program should work for any given 2D Arrays of size N×N.

You can use abs() function to calculate the absolute difference.

Given Array	Output
A = np.array([[1, 5, 12, 1], [2, -4, 6, 7], [3, 8, 5, 9], [3, 5, 23, -6]])	22

Explanation:

1	**5**	**12**	**1**
2	**-4**	**6**	**7**
3	**8**	**5**	**9**
3	**5**	**23**	**-6**

Primary Diagonal (elements with orange color) elements are 1, -4, 5, -6 (sum = -4).

Secondary Diagonal (elements with green color) elements are 1, 6, 8, 3 (sum = 18).

The absolute difference between the summation of its two diagonals will be:

$$| -4 - 18 | = | -22 | = 22$$

Task 10

Subtask 10.1: Suppose, you are given a matrix of dimension M×N called **A**. You have to transpose the matrix in a new 2D array. Finally, print the new array. Your program should work for any given 2D Arrays of size M×N.

The transpose of a matrix is a new matrix that is obtained by exchanging the rows and columns of the original matrix. Given a matrix A with dimensions M×N, the transpose A^T will have dimensions N×M, where the rows of A become the columns of A^T and vice versa.

Given Array	Output
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])	[[1 4 7] [2 5 8] [3 6 9]]
A = np.array([[1,2,3,4], [1,4,9,16]])	[[1 1] [2 4] [3 9] [4 16]]

Subtask 10.2:
Using the result from Subtask 10.1, calculate the gram matrix for A. You have to use loops to solve this task.

When a matrix (A) is multiplied by its transpose (A^T), the resulting symmetric matrix (A^T×A) is referred to as a "Gram matrix".
Matrix multiplication pairs up elements from rows in the first matrix with elements from columns in the second matrix.
Let's say, you have two matrices, Matrix A and Matrix B:

Matrix A:

$$\begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix}$$

Matrix B:

$$\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

When you multiply these matrices, each element in the resulting matrix is calculated by taking a row from Matrix A and a column from Matrix B, then pairing and multiplying the corresponding elements.
For example, to calculate the element in the first row and first column of the resulting matrix:

- *Row from Matrix A (first row): 2 3*
- *Column from Matrix B (first column): 5 7*
- *Pairing and Multiplying: Pair up corresponding elements: 2×5 and 3×7*
- *Sum the Products: 2 × 5 + 3 × 7 = 10 + 21 = 31*

This result, 31, becomes the first element in the resulting matrix.

Given Array A From The Previous Task	A^T	Output ($A^T \times A$)
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])	[[1 4 7] [2 5 8] [3 6 9]]	[[66 78 90] [78 93 108] [90 108 126]]
A = np.array([[1,2,3,4], [1,4,9,16]])	[[1 1] [2 4] [3 9] [4 16]]	[[2 6 12 20] [6 20 42 72] [12 42 90 156] [20 72 156 272]]

Task 11

Suppose, you are *given* an M×N array called A. Your task is to subtract the row and column numbers and find the absolute difference. Next, find the remainder (rounded to 2 decimal places) by dividing the absolute value by M. Now,

- If remainder = 0, then find the average of the 1st row.
- If remainder = 1, then find the average of the 2nd row.
- If remainder = 2, then find the average of the 3rd row.
- Similarly, if remainder = m-1, then find the average of the mth row.

Now multiply each element of the original array with the average to find out the final array. Finally, display the new array.

Given Array	Sample Output	Explanation
A = np.array([[4,1,2], [9,3,7]])	[[25.32 6.33 12.66] [56.97 18.99 44.31]]	Here, row number = 2 and column number = 3. So, the absolute value will be |2-3| = 1. absolute value % row number = 1 % 2 = 1 The remainder is 1 and the 2nd row is [9 3 7]. So the average is (9+3+7) / 3 = 6.33 Next, we have to multiply each element of the array with 6.33 to get the final array.
A = np.array([[6,3], [1,5]])	[[27. 13.5] [4.5 22.5]]	Here, row number = 2 and column number = 2. So, the absolute value will be |2-2| = 0. absolute value % row number = 0 % 2 = 0 The reminder is 0. and the 1st row is [6 3]. The average is (6+3) / 2 = 4.50 Next, we have to multiply each element of the array with 4.5 to get the final array.