CSC 157

Name \_James Aniciete\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date \_\_4/30/2020\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lab No. \_\_\_13\_\_\_\_

Purpose of the Lab Activity

**The purpose of this lab is to examine matrix operations involving arrays using exception handling.**

Source Code

# Programmer: James Aniciete

# Course No.: CSC 157

# Lab No.: 13

# Date: 4/30/2020

# program to add two 3 by 3 matrices using nested loops

MatX = [[9, 11, 4], [3, 10, 4], [6, 7, 11]]

MatY = [[3, 9, 2], [1, 9, 6], [10, 2, 4]]

SumXY = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

# iterate through all the rows

for i in range(len(MatX)) :

# iterate through all the columns

for j in range(len(MatX[0])) :

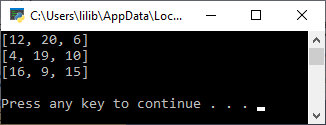
SumXY[i][j] = MatX[i][j] + MatY[i][j]

for elements in SumXY :

print(elements)

print(" ")

Snippet(s) of Output(s)



Modified Source Code

# Programmer: James Aniciete

# Course No.: CSC 157

# Lab No.: 13

# Date: 4/30/2020

### program to add two 3 by 3 matrices using nested loops

import numpy as np # for matrix algebra

try:

# original matrices

matx = [[9, 11, 4], [3, 10, 4], [6, 7, 11]]

maty = [[3, 9, 2], [1, 9, 6], [10, 2, 4]]

# raise an IndexError if a matrix isn't 3x3

if len(matx) != 3 or len(maty) !=3:

raise(IndexError)

for i in range(3):

if len(matx[i]) !=3 or len(maty[i]) != 3:

raise(IndexError)

# print the matrices

# Matrix X

print(" ----------------------")

print("Matrix X")

print(" ----------------------")

for i in matx:

print(i)

print("")

# Matrix Y

print(" ----------------------")

print("Matrix Y")

print(" ----------------------")

for i in maty:

print(i)

print("")

# scalar multiplication

def ScalarMult(ScaM, scalar):

for i in range(len(ScaM)):

for j in range(len(ScaM[0])):

ScaM[i][j] = scalar \* ScaM[i][j]

return ScaM

# matrix addition (includes option for subtraction)

def MatrixAdd(m1, m2, add = True):

# initialize variable for the sum

sum = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

# to subtract, use ScalarMult() to reverse the sign of the 2nd matrix

if add == False:

m2 = ScalarMult(m2, -1)

for i in range(len(m1)):

for j in range(len(m1[0])):

sum[i][j] += m1[i][j] + m2[i][j]

return sum

# matrix multiplication

def MatrixMult(m1, m2):

product = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

m1 = np.array(m1)

m2 = np.array(m2)

product = np.dot(m1,m2)

return product

# transpose

def MatrixTranspose(m):

transpose = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

for j in range(len(m[0])):

for i in range(len(m)):

transpose[i][j] = m[j][i]

return transpose

# Analyze Matrix & its Transpose, i.e. Matrix - Transpose

def TransposeAnalysis(m):

difference = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

transpose = MatrixTranspose(m)

difference = MatrixAdd(m, transpose, False)

return difference

# Matrix Determinant

def Determinant(m):

# return the determinant

return int(round(np.linalg.det(m))) # have to round with numpy.linalg.det()

# int() for formatting

# Trace of a Square Matrix

def Trace(m):

trace = 0

for i in range(len(m)):

for j in range(len(m[0])):

if i == j:

trace += m[i][j]

return trace

close = "N"

while close != "Y":

matrix = input("Which of the matrices would you like to analyze? (X, Y, or Both)\n")

print("Analysis Options:")

if matrix == "X" or matrix == "Y":

data = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

answer = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

# assign correct matrix to data

if matrix == "X":

data = matx

else:

data = maty

# display options

print("\t1. Scalar Multiplication")

print("\t2. Transpose")

print("\t3. Difference of Matrix and its Transpose")

print("\t4. Determinant")

print("\t5. Trace")

option = int(input("Enter an analysis option: "))

# based on option, call corresponding function

if option == 1:

scalar = int(input("Enter a scalar: "))

result = ScalarMult(data, scalar)

elif option == 2:

result = MatrixTranspose(data)

elif option == 3:

result = TransposeAnalysis(data)

elif option == 4:

print(Determinant(data))

elif option == 5:

print(Trace(data))

if option in [1,2,3]:

for i in result:

print(i)

elif matrix == "Both":

print("\t1. Matrix Addition")

print("\t2. Matrix Multiplication")

print("\t3. Transpose of Matrix Sum and Difference Between Originals & Transpose") # answers Steps 5 & 6

print("\tNote: Order does not matter for matrix addition and multiplication.")

option = int(input("Enter an analysis option: "))

if option == 1:

result = MatrixAdd(matx, maty)

for i in result:

print(i)

elif option == 2:

result = MatrixMult(matx, maty)

for i in result:

print(i)

elif option == 3:

# Steps 5 - Transpose of Matrix Sum

transpose5 = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

sumxy = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

diffx = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

diffy = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

sumxy = MatrixAdd(matx, maty)

transpose5 = MatrixTranspose(sumxy)

print("Tranpose of the Sum of Matrix X & Matrix Y i.e. (X + Y) ^ T")

for i in transpose5:

print(i)

print("")

# Step 6 - Difference between original matrices and transpose

diffx = MatrixAdd(matx, transpose5, False)

# have to multiply transpose by -1 since False case of MatrixAdd

# unintentionally stores the opposite of transpose5

transpose5 = ScalarMult(transpose5, -1)

diffy = MatrixAdd(maty, transpose5, False)

print("X - (X + Y) ^ T")

for i in diffx:

print(i)

print("")

print("Y - (X + Y) ^ T")

for i in diffy:

print(i)

else:

print("Invalid entry.")

close = input("Would you like to close the program? (Y/N)\n")

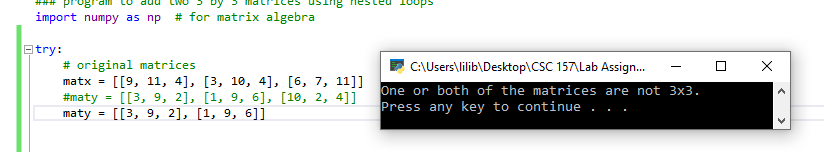
except IndexError:

print("One or both of the matrices are not 3x3.")

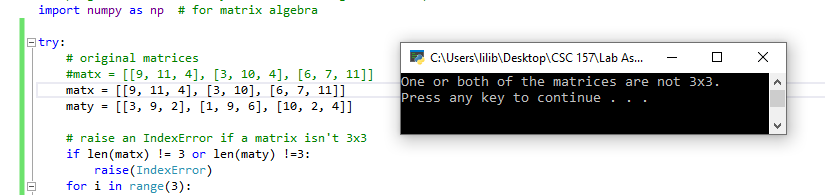
Snippet(s) of Output(s) from execution of modified Code

**Exception Handling:**

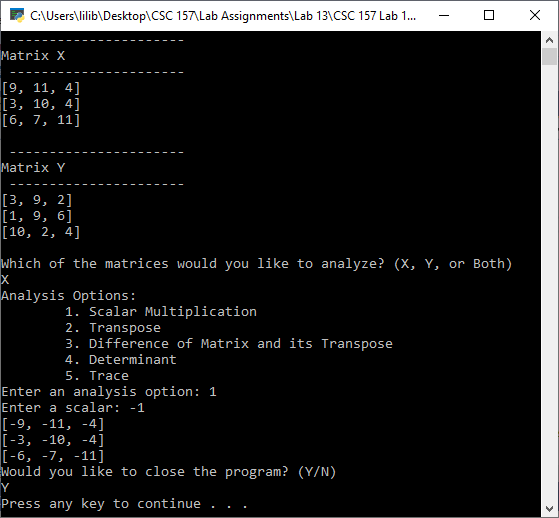
**Incorrect Row Length in Y:**



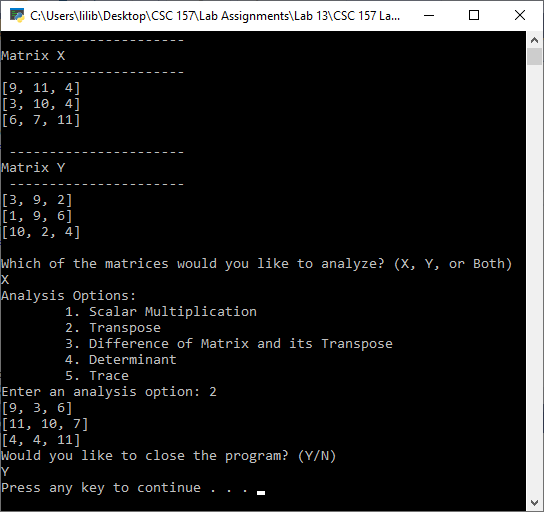
**Incorrect Column Length in X:**



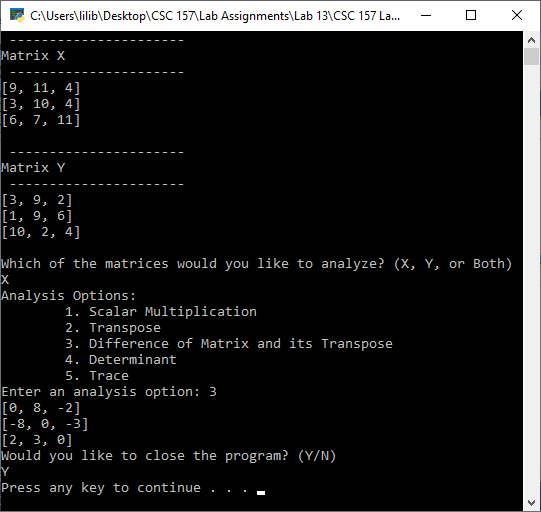
**Scalar Multiplication – Scalar Multiplied X by -1:**



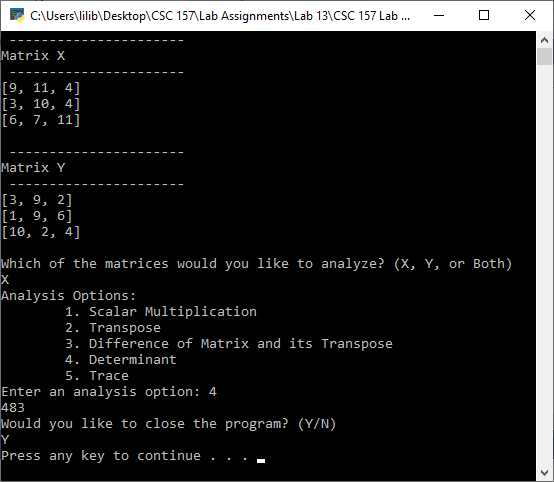
**Transpose of X, i.e. X ^ T:**



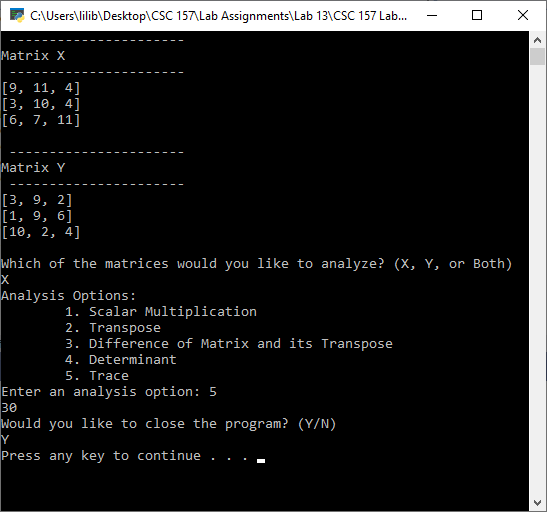
**Analyze a Matrix X and its Transpose, i.e. X – X^T:**



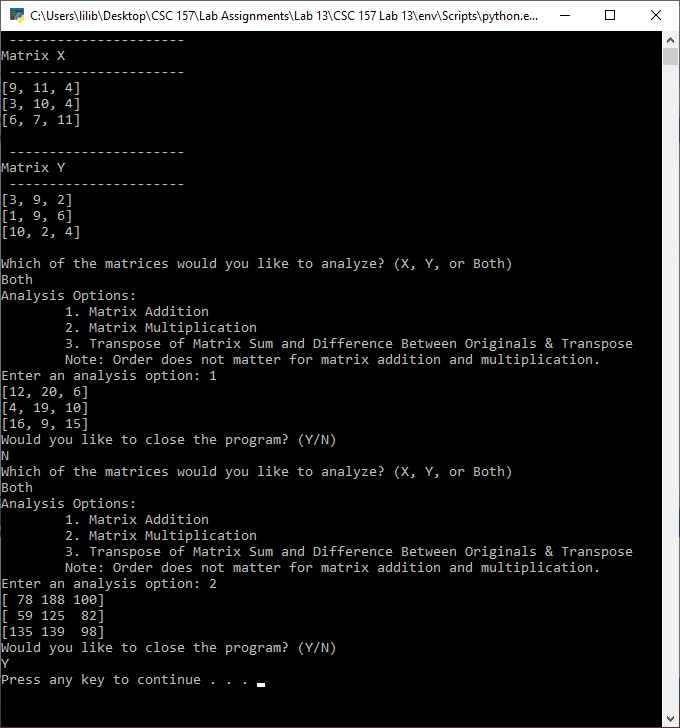
**Determinant of Matrix X, i.e. det(X):**



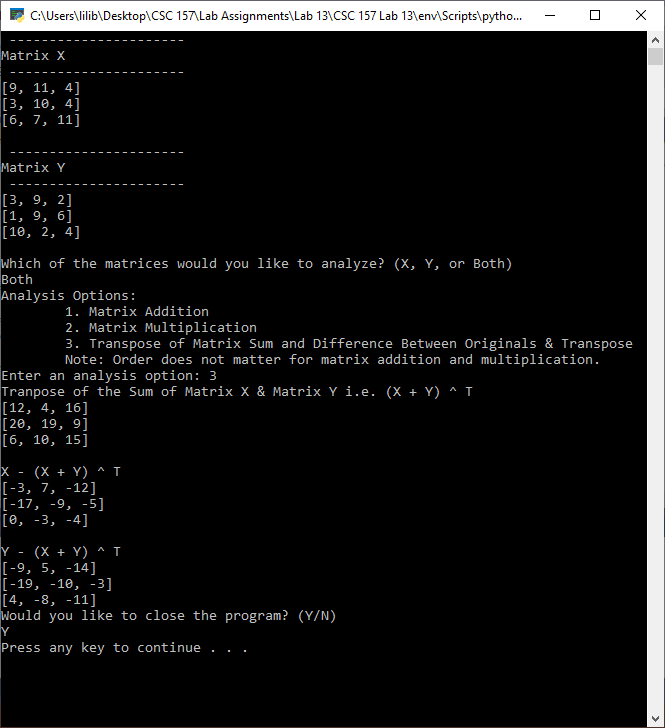
**Trace of Matrix X:**



**Matrix Addition and Matrix Multiplication:**



**Step 5 and Step 6 Answers – (X+Y)^T, X – (X+Y)^T, and Y – (X+Y)^T:**



Excel Spreadsheet (when Calculations are involved)

**N/A**

Answers to Questions (Be sure to copy the questios themselves!)

**(1)** Determine the row by column dimension of this list of elements.

**Mat\_A = [[7, 10, 4, 6], [2, 9, 3, 0], [0, 6, 1, 10]]**

**This is a 3x4 matrix.**

**(2)** Form the transpose of the given 3 × 3 matrix.

|  |  |  |
| --- | --- | --- |
| + 18 | + 33 | − 41 |
| − 41 | − 27 | + 73 |
| + 11 | + 69 | + 56 |

**Transposed Matrix:**

|  |  |  |
| --- | --- | --- |
| 18 | − 41 | 11 |
| 33 | − 27 | 69 |
| − 41- | 73 | 56+ |

**(3)** Explain, in detail, how exception handling is implemented with this block of code statements.

|  |
| --- |
| **a = [10, 20, 30, 40] # an array simulated as a list**  **try :**  **print("display an element in the array = %d" % (a[0]))**    **print("display and element in the array = %d" % (a[4]))**  **except IndexError :**  **print("an exception has occurred")** |

**This code tries to print the first and fifth elements in list a. If an element in the list does not exist for the given index, this means that an IndexError has occurred and thus the code will print that an exception has occurred. Here, the resulting output will be 10, the value of the first element of list a, and “an exception has occurred” for the fifth element in list a since the list only contains four elements.**

**(4)** What is meant by scalar multiplication?

**Scalar multiplication is multiplying each element in a matrix by the same number.**

Fill the blanks to perform scalar matrix multiplication, by 8 , on this list.

**matA = [2, -5, 6, 12]**

**print ("original array", matA)**

**for \_\_\_\_index\_\_\_\_\_\_ in range(len(matA)) :**

**matA[index] \*= \_\_\_8\_\_\_\_\_\_\_**

**print ("newly scaled array", \_\_\_matA\_\_\_\_\_\_\_ )**

**(5)** What have you learned from performing and coding this lab assignment?

**I have learned that using for loops to do calculations involving matrices are hard to troubleshoot. A lot of the errors that occurred dealt with the inability to convert integers to some data type, the existence of a variable with a value of None, and array or list variables not being iterable. Also, although numpy offers methods that calculate matrix operations faster, it is hard to format those results that sometimes are enclosed in two sets of brackets.**