Richard Hayes Crowley

03/22/2021

CSC\_157\_Lab\_013\_QA

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]]**

3 rows, four columns (3,4)

1. **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 ] ]

1. **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")** |

Exception handling is implemented using a try / except statement, and looks for and catches an IndexError ( which would be thrown by the second print statement, as a[4] does not exist).

1. **What is meant by scalar multiplication? 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 )**

Scalar multiplication is multiplying each item in a matrix by a certain number.

1. **What have you learned from performing and coding this lab assignment?**

Well, I learned a good bit about matrix operations. I began to implement my own formulas for finding the determinant and the trace of a matrix, but was pressed for time and reached for numpy to get the job done with` numpy.linalg.det(matrix)`. Finding the trace of a matrix seemed easy enough, but I went ahead and used numpy for that, as well. I did write my own formulas for transposing matrices and the AnalyzeTranspose function and SubtractMatrices functions however.  
  
As for exception handling, I already understood the concept and implementation, but it was good to get some practice, and gratifying to see it working in this moderately complex CLI tool.   
  
One thing I did enjoy doing, was using a dict to store key/value pairs with option keys [“a”,”b”,”c”, …etc.] with function values that were imported as matrix\_utils from a file module I wrote (operations.py). I’m familiar with this modular pattern of importing functions and mapping them with key/value pairs in JavaScript / React frontend development, and I’m glad we can write similarly modular code in Python.