Richard Hayes Crowley

03/22/2021

CSC\_157\_Lab\_013

**SOURCE CODE:**

**(operations.py)**

*import* numpy *as* np

def ScalarMult(*ScaM*, *num*):

*for* i *in* range(len(ScaM)):

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

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

*return* ScaM

def MatrixAdd(*Matrix1*, *Matrix2*):

*for* i *in* range(len(Matrix1)):

*for* j *in* range(len(Matrix1[0])):

Matrix1[i][j] = Matrix1[i][j] + Matrix2[i][j]

*return* Matrix1

def MatrixSubtract(*Matrix1*, *Matrix2*):

*for* i *in* range(len(Matrix1)):

*for* j *in* range(len(Matrix1[0])):

Matrix1[i][j] = Matrix1[i][j] - Matrix2[i][j]

*return* Matrix1

def MatrixMultiply(*Matrix1*, *Matrix2*):

*for* i *in* range(len(Matrix1)):

*for* j *in* range(len(Matrix1[0])):

Matrix1[i][j] = Matrix1[i][j] \* Matrix2[i][j]

*return* Matrix1

def MatrixTranspose(*Matrix*):

transposedMatrix = [[] *for* i *in* range(len(Matrix))]

*for* row *in* Matrix:

*for* idx, item *in* enumerate(row):

transposedMatrix[idx].append(item)

*return* transposedMatrix

def AnalyzeTranspose(*Matrix*):

transposedMatrix = MatrixTranspose(Matrix)

*return* MatrixSubtract(Matrix, transposedMatrix)

def MatrixDeterminant(*Matrix*):

*return* np.linalg.det(Matrix)

def MatrixTrace(*Matrix*):

*return* np.trace(Matrix)

**(lab\_13.py)**

*# program to add manipulate 3 by 3 matrices using nested loops*

*# Richard Hayes Crowley, CSC\_157, 03-22-2021*

*import* operations *as* matrix\_utils

global continue\_program

def main():

class OptionException(Exception):

"""Custom exception for not choosing correct option"""

*pass*

continue\_program = True

*# original matrices*

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

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

def showMatrix(*MatA*, *m*, *n*):

*for* i *in* range(m):

*for* j *in* range(n):

*try*:

print(MatA[i][j], "\t", *end*="")

*except* IndexError:

print(

"Out of range, are you sure you inputted the correct rows / columns length?")

print("\n")

*# Matrix X*

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

print("Matrix X")

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

showMatrix(MatX, 3, 3)

print(" ")

*# Matrix Y*

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

print("Matrix Y")

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

showMatrix(MatY, 3, 3)

optionsDict = {

"test": print,

"a": matrix\_utils.ScalarMult,

"b": matrix\_utils.MatrixAdd,

"c": matrix\_utils.MatrixMultiply,

"d": matrix\_utils.MatrixTranspose,

"e": matrix\_utils.AnalyzeTranspose,

"f": matrix\_utils.MatrixDeterminant,

"g": matrix\_utils.MatrixTrace

}

*while* continue\_program:

*# Scalar Multiplication*

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

print("Please choose an operation to perform on a matrix or matrices (or type Q to quit): ")

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

option = ""

*while* True:

*try*:

option = input(

"A) Scalar Multiply\nB) Matrix Add \nC) Matrix Multiply \nD) Matrix Tranpose \nE) Analyze Transpose \nF) Matrix Determinant \nG) Matrix Trace\n\rChoice: ").lower()

*if* option == "q":

*break*

*elif* option not in ["a", "b", "c", "d", "e", "f", "g"]:

*raise* OptionException

*else*:

*break*

*except* OptionException:

print(

"Please input one of the following or else type 'Q' to quit: A, B, C, D, E, F, G")

*if* option == "q":

print("Goodbye!")

continue\_program = False

*elif* option.lower() == "a":

*while* True:

*try*:

mChoice = input(

"Which matrix would you like to scale? A) MatrixX or B) MatrixY. Type Q to quit.\nChoice: ").lower()

*if* mChoice == "q":

print("goodbye!")

continue\_program = False

*break*

*elif* mChoice not in ["a", "b"]:

*raise* OptionException

scale = int(

input("Please input an integer to scale this matrix by.\nInteger: "))

print(optionsDict["a"](

MatX *if* mChoice == "a" *else* MatY, scale))

cont = input(

"Would you like to continue? Enter Y if so.").lower()

*if* cont != "y":

continue\_program = False

*break*

*except* OptionException:

print("Please select A or B (or type Q to quit).")

*except* ValueError:

print("Please enter an integer.")

*elif* option.lower() in ["b", "c"]:

print(optionsDict[option](MatX, MatY))

cont = input("Would you like to continue? Enter Y if so: ").lower()

*if* cont != "y":

continue\_program = False

*elif* option.lower() in ["d", "e", "f", "g"]:

*while* True:

*try*:

mChoice = input(

"Which matrix would you like to perform this operation on? A) MatrixX or B) MatrixY. Type Q to quit.\nChoice: ").lower()

*if* mChoice == "q":

print("goodbye!")

continue\_program = False

*break*

*elif* mChoice not in ["a", "b"]:

*raise* OptionException

*else*:

print(optionsDict[option](

MatX *if* mChoice == "a" *else* MatY))

cont = input(

"Would you like to continue? Enter Y if so.")

*if* cont.lower() != "y":

continue\_program = False

*break*

*except* OptionException:

print("Please select A or B (or type Q to quit).")

main()

**Output on next page…**

**OUTPUT (3 screenshots):  
  
Text

Description automatically generated**Text

Description automatically generated

Text

Description automatically generated