|  |  |
| --- | --- |
| **NAME** | **DUE** |
| **TOMOKI KOIKE** | **NOV. 17 2019** |

**Every Boilermaker Engineering Code – Intermediate Level Programming**

**Week 8 – Programming Exercises**

|  |  |
| --- | --- |
| **QUESTION** | **SCORE** |
| **#1** | **/20** |
| **#2** | **/20** |
| **#3** | **/30** |
| **TOTAL** | **/70** |

1. **(20 points, Coordinate Base)** When drawing a 2D filled contour plot, a pair of 2D coordinate base arrays are required. You can use the contourf function in the module matplotlib.pyplot to draw the 2D filled contour plot. The contour function needs 3 arguments, i.e.s

where xx and yy are the pair of coordinate base arrays.

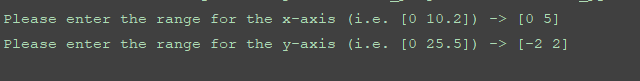
In the xx array, every row should be the same and every column should be the value for x-axis. In the yy array, every column should be the same and every row should be the value for y-axis. For example, a pair of 3 by 3 coordinate base arrays can be:

In this question, you are given the data of 2D velocity field of a linear shear flow in the file ‘linear\_shear\_flow\_u.txt’, which includes a 63 by 63 matrix. Write a program to create the coordinate base arrays and plot the filled contour plot. The x axis ranges from 0 to 5 and the y axis ranges from -2 to 2.

**>>CODE**

### QUESTION #1 ###  
# AUTHOR : TOMOKI KOIKE  
# DATE : NOV. 17 2019  
# DESCRIPTION: THIS PROGRAM IS DESIGNED TO PLOT A 2D FILED CONTOUR GRAPH FOR A PROVIDED DATA OF  
# A VELOCITY FIELD OF A LINEAR SHEAR FLOW.  
###################  
  
# Main function  
**def main():** # Obtaining the ranges for the axes from the user  
 **[**xlow, xhigh, ylow, yhigh**] =** enterInput**()  
 [**xmat, ymat**] =** createAxisArray**(**xlow, xhigh, ylow, yhigh**)** plotContour**(**xmat, ymat, readData**('linear\_shear\_flow\_u.txt'))**# Import modules  
**import** re  
**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
  
# Functions  
# Function for the user to enter inputs  
**def enterInput():** xrange **=** getValid**('Please enter the range for the x-axis (i.e. [0 10.2]) -> ')** xlow **=** xrange**[**0**]** xhigh **=** xrange**[**1**]** yrange **=** getValid**('Please enter the range for the y-axis (i.e. [0 25.5]) -> ')** ylow **=** yrange**[**0**]** yhigh **=** yrange**[**1**]  
 return** xlow, xhigh, ylow, yhigh  
  
# Function that extracts the numbers from the input string  
**def extractNum(**astring**):** astring **=** astring.replace**('['**, **'')** astring **=** astring.replace**(']'**, **'')** astring **=** astring.split**()  
 return** astring  
  
# Function to validate input  
**def getValid(**prompt**):** base **=** re.compile**('[^0-9.-]')  
 while True:  
 try:** this **=** input**(**prompt**)** num\_list **=** extractNum**(**this**)** low **=** num\_list**[**0**]** high **=** num\_list**[**1**]  
 except** ValueError**:** # Invalid input  
 print**('Sorry, could not understand input. Please try again.')  
 continue  
 if** base.search**(**low**) or** base.search**(**high**):** # Invalid input  
 print**('Inappropriate input. Please try again.')  
 continue  
 else:** # Valid input  
 **break  
 return** list**(**map**(**float, num\_list**))**# Function to read the text file  
**def readData(**file**):** afile **=** open**(**file, **'r')** # Opening the file in read mode  
 aline **=** afile.readline**()** # Read the first line  
 data\_matrix **=** np.empty**([**63, 63**]**, dtype**=**float**)** # Initialize an empty numpy array  
 n **=** 0 # Initialize the counter for the index  
 **while** aline **!= '':** aline **=** aline.replace**('\t'**, **'')** # Replacing the newline character  
 aline **=** aline.split**()** # Splitting the string into a list  
 aline **=** list**(**map**(**float, aline**))** # Changing the data types into floats  
 data\_matrix**[**n, **:] =** aline # Replacing the initialized numpy matrix with the data  
 aline **=** afile.readline**()** # Reading a new line  
 n **+=** 1 # Incrementing the counter  
 **return** data\_matrix  
  
# Function that creates the array with x- and y-axis indices for the contour plot  
**def createAxisArray(**xlow, xhigh, ylow, yhigh**):** size **=** 63 # Size of the data matrix  
 # Intialize a numpy array for the x and y axis matrices  
 xmat **=** np.random.random\_sample**([**63, 63**])** xline **=** np.linspace**(**start**=**0, stop**=**5, num**=**63**)** ymat **=** np.random.random\_sample**([**63, 63**])** yline **=** np.linspace**(**start**=-**2, stop**=**2, num**=**63**)  
 for** n **in** range**(**size**):** xmat**[**n, **:] =** xline  
 ymat**[**n, **:] =** yline  
 **return** xmat, ymat.T  
  
# Function to plot the contour graph  
**def plotContour(**xmat, ymat, data**):** plt.contourf**(**xmat, ymat, data**)** plt.title**('Velocity Field of Linear Shear Flow')** plt.xlabel**('x positions')** plt.ylabel**('y positions')** plt.savefig**('shear\_flow\_u\_contour.png')** plt.show**()  
 return**# Executing main function  
**if** \_\_name\_\_ **== '\_\_main\_\_':** main**()**

**>>OUTPUT**



A close up of a logo

Description automatically generated

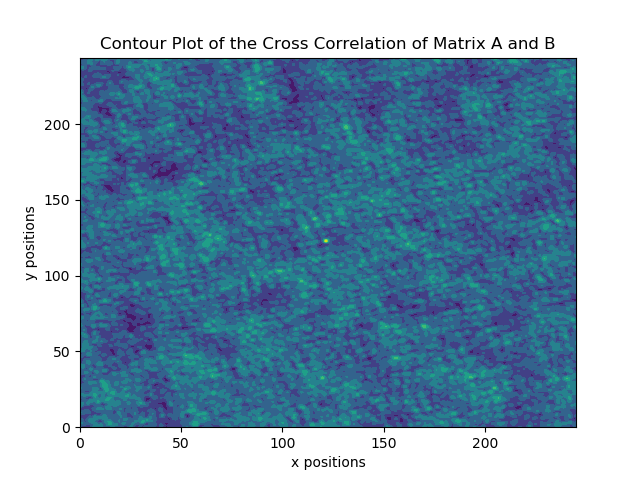
1. **(20 points, Cross Correlation)** The cross correlation can be determined between a large matrix A (m by n) and a small matrix B (x by x) only when they satisfy: x<=m and x<=n. The cross correlation between A and B is another matrix C whose size is (m-x+1) by (n-x+1).

In this question, you are given the two matrix A (1024\*1024) and matrix B (64\*64), which are in ‘matrix\_A’ and ‘matrix\_B’, respectively. Please write a program to calculate the cross correlation between these two matrices and draw the filled contour as discussed in the previous question. You can set your own scales of x and y axes.

**>>CODE**

### QUESTION #2 ###  
# AUTHOR : TOMOKI KOIKE  
# DATE : NOV. 17 2019  
# DESCRIPTION: THIS PROGRAM IS DESIGNED TO COMPUTE THE CROSS CORRELATION OF PROVIDED DATA OF TWO FILES AND  
# PLOT THE 2D CONTOUR OF THE RESULT.  
###################  
  
# Main function  
**def main():** matA **=** readData**('matrix\_A.txt')** # Reading matrix A  
 matB **=** readData**('matrix\_B.txt')** # Reading matrix B  
 matC **=** crossCorr**(**matA, matB**)** # Computing the cross correlation  
 l **=** list**(**np.shape**(**matC**))** # Obtaining the shape of matC as a list and not tuple  
 size **=** l**[**0**]** # Computing the x- and y- axis matrices  
 xmat, ymat **=** createAxisArray**(**np.min**(**matA**)**, np.max**(**matA**)**, np.min**(**matB**)**, np.max**(**matB**)**, size**)** # Plotting the contour  
 plotContour**(**xmat, ymat, matC**)**# Import modules  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** scipy **import** signal  
  
# Functions  
# Function to read the text file  
**def readData(**file**):** afile **=** open**(**file, **'r')** # Opening the file in read mode  
 aline **=** afile.readline**()** # Read the first line  
 length **=** measureLen**(**aline**)** # Get the length of the matrix rows and columns  
 data\_matrix **=** np.empty**([**length, length**]**, dtype**=**float**)** # Initialize an empty numpy array  
 n **=** 0 # Initialize the counter for the index  
 **while** aline **!= '':** aline **=** aline.replace**('\t'**, **' ')** # Replacing the newline character  
 aline **=** aline.replace**('\n'**, **' ')** # Replacing \n with space  
 aline **=** aline.split**()** # Splitting the string into a list  
 aline **=** list**(**map**(**float, aline**))** # Changing the data types into floats  
 data\_matrix**[**n, **:] =** aline # Replacing the initialized numpy matrix with the data  
 aline **=** afile.readline**()** # Reading a new line  
 n **+=** 1 # Incrementing the counter  
 **return** data\_matrix  
  
# Function to split the read line to find the length of the row and column  
**def measureLen(**astring**):** astring **=** astring.replace**('\t'**, **' ')** # Replacing the newline character  
 astring **=** astring.replace**('\n'**, **' ')** astring **=** astring.split**()** # Splitting the string into individual elements as a list  
 **return** len**(**astring**)**# Function to compute the cross-correlation of 2 matrices  
**def crossCorr(**matA, matB**):** sizeA **=** np.array**(**list**(**np.shape**(**matA**))**, dtype**=**int**)** # Obtaining the size of matA in immutable form  
 sizeB **=** np.array**(**list**(**np.shape**(**matB**))**, dtype**=**int**)** # OBtaining the size of matB in immutable form  
 sizeC **=** sizeA **-** sizeB **+ [**1, 1**]** # Calculating the size of the matrix with the cross correlation results  
 corr **=** signal.correlate2d**(**matA, matB, mode**='valid'**, boundary**='fill')** # Using Scipy's 2D cross correlation  
  
 ### The generic way of cross correlation using 4 for loops ###  
 # matC = np.random.random\_sample(sizeC) # Initializing the matrix to store the resulting matrix  
 # for p in range(sizeC[0]):  
 # for q in range(sizeC[1]):  
 # point = 0 # Initialize temporary value holder  
 # for i in range(sizeB[0]-1):  
 # for j in range(sizeB[1]-1):  
 # point += (matA[p+i, q+j] - matB[i, j])\*\*2  
 # matC[p, q] = point  
 **return** corr  
  
# Function that creates the array with x- and y-axis indices for the contour plot  
**def createAxisArray(**xlow, xhigh, ylow, yhigh, size**):** # Intialize a numpy array for the x and y axis matrices  
 xmat **=** np.random.random\_sample**([**size, size**])** xline **=** np.linspace**(**start**=**xlow, stop**=**xhigh, num**=**size**)** ymat **=** np.random.random\_sample**([**size, size**])** yline **=** np.linspace**(**start**=**ylow, stop**=**yhigh, num**=**size**)  
 for** n **in** range**(**size**):** xmat**[**n, **:] =** xline  
 ymat**[**n, **:] =** yline  
 **return** xmat, ymat.T  
  
# Function to plot the contour graph  
**def plotContour(**xmat, ymat, data**):** plt.contourf**(**xmat, ymat, data**)** plt.title**('Contour Plot of the Cross Correlation of Matrix A and B')** plt.xlabel**('x positions')** plt.ylabel**('y positions')** plt.savefig**('xcorr\_contour\_plot.png')** plt.show**()  
 return**# Execute the main function  
**if** \_\_name\_\_ **== '\_\_main\_\_':** main**()**

**>>OUTPUT**

****

1. **(30 points, File Analysis)** In the attachments, there are two files named “xian\_1.txt” and “xian\_2.txt”. Write a program that reads the contents of two text files and compares then in the following ways:
2. Write the unique words contained in “xian\_1.txt” and the frequency (the number of times each word appears.) of each word to file “word\_frequency\_1.txt”. The format in the text file should be:

Word: Frequency

walls: 5

the: 30

…

1. Write the unique words contained in “xian\_2.txt” and the frequency (the number of times each word appears.) of each word to file “word\_frequency\_2.txt”. The format in the text file should be:

Word: Frequency

walls: 3

the: 33

…

1. Write a list of the words that appear in both files to file “common\_words.txt”, one word per line.
2. Write a list of the words that appear in either the first or second file, but not both to file “eitherbutnotboth.txt”

**(Note: ‘Xian’s’ counts as ‘xian’, ‘Xian’ also counts as ‘xian’. You will apply lists, sets, regular expression in this program.)**

**>>CODE**

### QUESTION #3 ###  
# AUTHOR : TOMOKI KOIKE  
# DATE : NOV. 17 2019  
# DESCRIPTION: THIS PROGRAM IS DESIGNED TO COMPARE THE UNIQUE WORDS IN TWO TEXT FILES AND FIND THE  
# WORDS UNIQUE TO EACH FILE OR COMMON TO BOTH FILES.  
###################  
  
# The main function  
**from** typing **import** Pattern  
  
  
**def main():** unqdict1, list1 **=** scanFile**('xian\_1.txt')** unqdict2, list2 **=** scanFile**('xian\_2.txt')** writeUnq**(**unqdict1, **'word\_frequency\_1.txt')** writeUnq**(**unqdict2, **'word\_frequency\_2.txt')** commonWords**(**list1, list2, **'common\_words.txt')** unq2either**(**list1, list2, **'eitherbutnotboth.txt')**# Importing modules  
**import** re  
  
# Functions  
# Function that read through the file  
**def scanFile(**file**):** afile **=** open**(**file, **'r'**, encoding**='utf-8')** # Open the file in read mode  
 lesen **=** afile.read**()** # Read the entire file  
 lesen **=** lesen.lower**()**.split**()** lesen **=** filterRegEx**(**lesen**)** tempList **=** lesen  
 unqdict **=** scanIt**(**lesen**)** # Creating a dictionary with all the unique words and count of them  
 afile.close**()** # Close the file  
 **return** unqdict, tempList  
  
# Function to filter with regEx  
**def filterRegEx(**stringList**):** base1 **=** re.compile**(r'.+(n\’t)+$')** base2 **=** re.compile**(r'[a-z](\’s)$')** base3**:** Pattern**[**str**] =** re.compile**(r'([a-z]\.[a-z]\.)')** base4 **=** re.compile**(r'.+\W$')** base5 **=** re.compile**(r'^\W+.')** base6 **=** re.compile**(r'^\w+\W+\w+$')** base7 **=** re.compile**(r'\W')  
 for** i, v **in** enumerate**(**stringList**):  
 if** base1.search**(**v**):** stringList**[**i**] =** re.sub**(**base1, **''**, v**)  
 elif** base2.search**(**v**):** stringList**[**i**] =** re.sub**(**base2, **''**, v**)  
 elif** base3.search**(**v**):** stringList**[**i**] =** v.upper**()  
 elif** base4.search**(**v**):** temp **=** re.findall**(**base4, v**)[**0**]** stringList**[**i**] =** v.rstrip**(**temp**[-**1**])  
 elif** base5.search**(**v**):** temp **=** re.findall**(**base5, v**)[**0**]** stringList**[**i**] =** v.lstrip**(**temp**[**0**])  
 elif** base6.search**(**v**):  
 pass  
 elif** base7.search**(**v**):** stringList**[**i**] =** re.sub**(**base7, **''**, v**)  
 if not** stringList**[**i**]:** stringList.pop**(**i**)  
 return** stringList  
  
# Function to scan through the read line of the file and find unique words  
**def scanIt(**strList**):** # Call the rmDuplicate function to get a dict of unique words being the keys and count as values  
 unqdict **=** rmDuplicate**(**strList**)** # For loop to count the number of each unique word in the line list and assign them to the value  
 # of the dictionary  
 **for** x **in** strList**:** unqdict**[**x**] +=** 1  
 **return** unqdict  
  
# Function to remove duplicate values or strings from a list and return as a dictionary  
**def rmDuplicate(**alist**):** adict **=** dict.fromkeys**(**alist**)** # Form dictionary from the list  
 # For loop to change the NoneType to 0  
 **for** x **in** adict**:  
 if** adict**[**x**] == None:** adict**[**x**] =** 0  
 **return** adict  
  
# Function to remove duplicate values of strings from a list  
**def rmDuplicateList(**alist**):** alist **= [**x **for** x **in** alist **if** x **!= 's']  
 return** list**(**dict.fromkeys**(**alist**))**# Function to create a new file with all the common words in both files  
**def commonWords(**list1, list2, filename**):** # Convert both list to tuple  
 set1 **=** set**(**rmDuplicateList**(**list1**))** set2 **=** set**(**rmDuplicateList**(**list2**))** # Find the common words  
 commonset **=** set1 **&** set2  
 commonlist **=** list**(**commonset**)** newfile **=** open**(**filename, **'w')** # Opening file  
 **for** n **in** commonlist**:** newfile.write**('{0}\n'**.format**(**n**))** newfile.close**()  
 return**# Function to create a new file with the words unique to either of the files  
**def unq2either(**list1, list2, filename**):** # Convert both list to tuple  
 set1 **=** set**(**rmDuplicateList**(**list1**))** set2 **=** set**(**rmDuplicateList**(**list2**))** # Find the common words  
 unq2Oneset **=** set1 **^** set2  
 unq2Onelist **=** list**(**unq2Oneset**)** newfile **=** open**(**filename, **'w')** # Opening file  
 **for** n **in** unq2Onelist**:** newfile.write**('{0}\n'**.format**(**n**))** newfile.close**()  
 return**# Writing the unique words in the file to a new text file as results  
**def writeUnq(**adict, name**):** newfile **=** open**(**name, **'w')** # Opening a new file write mode  
 # Looping through to write  
 **for** k, v **in** adict.items**():** newfile.write**('{0}: {1}\r'**.format**(**k, v**))** newfile.close**()** # Close written file  
 **return**# Executing the main function  
**if** \_\_name\_\_ **== '\_\_main\_\_':** main**()**

**>>OUTPUT**

**See the submitted .txt files**