Name : Dhaval Gogri

SMU ID : 47444609

Course : Advanced Application Programming

Quest : Quest 8

In [65]:
```
import numpy
import random
import timeit
from matplotlib import pyplot as plt
```

Martix Creation

In [66]:
```
# Create matrix of element
def createMatrixOfElements(number, shouldPrint = False):
    matrixElement = 0
    list = []
    for x in range(number):
        tempList = []
        for y in range(number):
            tempList.append(matrixElement)
            matrixElement = matrixElement + 1
        list.append(tempList)
    if(shouldPrint):
        print(list)
    return list
```

Simple Method

In [67]:
```
def getTransposeSimpleMethod(m, shouldPrint = False):
    rez = [[m[j][i] for j in range(len(m))] for i in range(len(m[0]))]
    if(shouldPrint):
        print("\n Transposed List Below - Simple Method")
        for row in rez:
            print(row)
```

Zip Method

In [68]:
```
def getTransposeZipMethod(matrix, shouldPrint = False):
    t_matrix = zip(*matrix)
    if(shouldPrint):
        print("\n Transposed List Below - Zip Method")
        for row in t_matrix:
            print (row )
```

Numpy Method

In [69]:
```
def getTransposeNumpyMethod(m, shouldPrint = False):
    m2 = numpy.transpose(m)
    if(shouldPrint):
        print("\n Transposed List Below - Numpy Method")
        print(m2)
```

Get Matrix Transpose

In [70]:
```
def getMatrixTranspose(matrixName, transposeMethodName, shouldPrint, numberOfLoopForTimeIt = 1):
    timeList = []
    for eachTransposeMethodName in transposeMethodName:
        timeitParameter1 = '' + eachTransposeMethodName + '(' + matrixName + ', ' + str(shouldPrint) + ')'
        timeitParameter2 = 'from __main__ import ' + eachTransposeMethodName + ', ' + matrixName
        time = timeit.timeit(timeitParameter1, timeitParameter2, number=numberOfLoopForTimeIt)
        timeList.append(time)
    return timeList
```

In [71]:
```
def printTime(timeList):
    for eachTime, eachTranspose in zip(timeList, transposeList):
        printString = "Time for " + eachTranspose + "  =  " + str(eachTime)
        print(printString)
```

In [72]:
```
def plotLineGraph(listOfMatrixSize, listOfTime):
    simpleMethod = []
    zipMethod = []
    numpyMethod = []
    i = 0
    for eachTime in listOfTime:
        i = 0
        for eachTimeData in eachTime:
            if(i == 0):
                simpleMethod.append(eachTimeData)
            elif (i == 1):
                zipMethod.append(eachTimeData)
            elif (i == 2):
                numpyMethod.append(eachTimeData)
            i = i + 1
    plt.figure(figsize=(10,10))
    plt.plot(listOfMatrixSize, simpleMethod, color='red', label = "Simple Method")
    plt.plot(listOfMatrixSize, zipMethod, color='green', label = "Zip Method")
    plt.plot(listOfMatrixSize, numpyMethod, color='blue', label = "Numpy Method")
    plt.xlabel('Matrix Size')
    plt.ylabel('Time in seconds')
    plt.title('Matrix Size vs Time for all Methods')
    plt.legend(numpoints=1)
    plt.show()
```

In [73]:
```
transposeMethodNameList = ['getTransposeSimpleMethod', 'getTransposeZipMethod', 'getTransposeNumpyMethod']
transposeList = ['Simple Method', 'Zip Method', 'Numpy Method']
```

5 x 5

In [74]:
```
matrix5 = createMatrixOfElements(5, True)
```

```
[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14], [15, 16, 17, 18, 19], [20, 21, 22, 23, 24]]
```

In [75]:
```
timeList5 = getMatrixTranspose('matrix5', transposeMethodNameList, True, 1)
```

```
 Transposed List Below - Simple Method
[0, 5, 10, 15, 20]
[1, 6, 11, 16, 21]
[2, 7, 12, 17, 22]
[3, 8, 13, 18, 23]
[4, 9, 14, 19, 24]

 Transposed List Below - Zip Method
(0, 5, 10, 15, 20)
(1, 6, 11, 16, 21)
(2, 7, 12, 17, 22)
(3, 8, 13, 18, 23)
(4, 9, 14, 19, 24)

 Transposed List Below - Numpy Method
[[ 0  5 10 15 20]
 [ 1  6 11 16 21]
 [ 2  7 12 17 22]
 [ 3  8 13 18 23]
 [ 4  9 14 19 24]]
```

In [76]:
```
printTime(timeList5)
```

```
Time for Simple Method  =  0.00035598999966168776
Time for Zip Method  =  0.00029699500009883195
Time for Numpy Method  =  0.00034561400025268085
```

200 x 200

In [77]:
```
matrix200 = createMatrixOfElements(200, False)
```

In [78]:
```
timeList200 = getMatrixTranspose('matrix200', transposeMethodNameList, False, 1)
```

In [79]:
```
printTime(timeList200)
```

```
Time for Simple Method  =  0.013150711998605402
Time for Zip Method  =  1.61239986482542e-05
Time for Numpy Method  =  0.002564808000897756
```

2000 x 2000

In [80]:
```
matrix2000 = createMatrixOfElements(2000, False)
```

In [81]:
```
timeList2000 = getMatrixTranspose('matrix2000', transposeMethodNameList, False, 1)
```

In [82]:
```
printTime(timeList2000)
```

```
Time for Simple Method  =  1.3633359109990124
Time for Zip Method  =  0.00011200699918845203
Time for Numpy Method  =  0.22538940499907767
```

20000 x 20000

In [83]:
```
matrix20000 = createMatrixOfElements(20000)
```

In [84]:
```
timeList20000 = getMatrixTranspose('matrix20000', transposeMethodNameList, False, 1)
```

In [85]:
```
printTime(timeList20000)
```

```
Time for Simple Method  =  325.5487300620007
Time for Zip Method  =  0.004506115999902249
Time for Numpy Method  =  68.24860644
```

30000 x 30000

In [86]:
```
matrix30000 = createMatrixOfElements(30000)
```

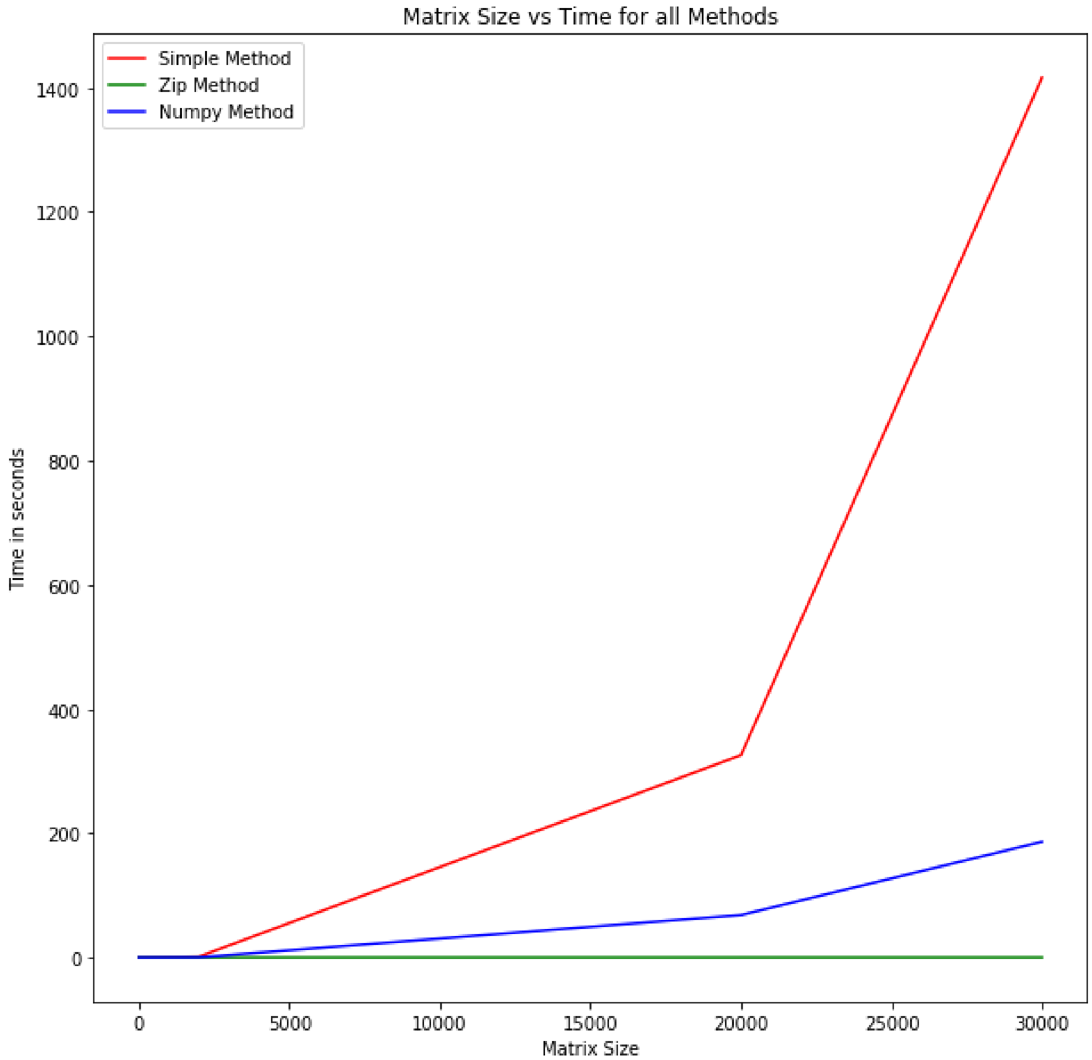
In [87]:
```
timeList30000 = getMatrixTranspose('matrix30000', transposeMethodNameList, False, 1)
```

In [88]:
```
printTime(timeList30000)
```

```
Time for Simple Method  =  1416.1061392029987
Time for Zip Method  =  0.012699921997409547
Time for Numpy Method  =  186.29550401499728
```

Comparision between different Methods

In [89]:
```
matrixSizeList = [5, 200, 2000, 20000, 30000]
timeList = [timeList5, timeList200, timeList2000, timeList20000, timeList30000]
plotLineGraph(matrixSizeList, timeList)
```



In [120]:
```
print("Size 200 and 2000")
simpleTimeDiff = timeList2000[0]/timeList200[0]
simpleSizeDiff = (2000*2000)/(200*200)
print("Simple Method Size Diff = " + str(simpleSizeDiff))
print("Simple Method Time Diff = " + str(simpleTimeDiff))


zipTimeDiff = timeList2000[1]/timeList200[1]
zipSizeDiff = (2000*2000)/(200*200)
print("Zip Method Size Diff = " + str(zipSizeDiff))
print("Zip Method Time Diff = " + str(zipTimeDiff))


numpyTimeDiff = timeList2000[2]/timeList200[2]
numpySizeDiff = (2000*2000)/(200*200)
print("Numpy Method Size Diff = " + str(numpySizeDiff))
print("Numpy Method Time Diff = " + str(numpyTimeDiff))




print("\n\nSize 2000 and 20000")
simpleTimeDiff2 = timeList20000[0]/timeList2000[0]
simpleSizeDiff2 = (20000*20000)/(2000*2000)
print("Simple Method Size Diff = " + str(simpleSizeDiff2))
print("Simple Method Time Diff = " + str(simpleTimeDiff2))


zipTimeDiff2 = timeList20000[1]/timeList2000[1]
zipSizeDiff2 = (20000*20000)/(2000*2000)
print("Zip Method Size Diff = " + str(zipSizeDiff2))
print("Zip Method Time Diff = " + str(zipTimeDiff2))


numpyTimeDiff2 = timeList20000[2]/timeList2000[2]
numpySizeDiff2 = (20000*20000)/(2000*2000)
print("Numpy Method Size Diff = " + str(numpySizeDiff2))
print("Numpy Method Time Diff = " + str(numpyTimeDiff2))
```

```
Size 200 and 2000
Simple Method Size Diff = 100.0
Simple Method Time Diff = 103.6701215222104
Zip Method Size Diff = 100.0
Zip Method Time Diff = 6.946601871650454
Numpy Method Size Diff = 100.0
Numpy Method Time Diff = 87.87769100852186
```

```
Size 2000 and 20000
Simple Method Size Diff = 100.0
Simple Method Time Diff = 238.7883480773628
Zip Method Size Diff = 100.0
Zip Method Time Diff = 40.2306644455379
Numpy Method Size Diff = 100.0
Numpy Method Time Diff = 302.80308180537276
```

In [*]:

```
    plt.figure(figsize=(10,10))
    plt.plot([200, 2000, 20000], [timeList200[0], timeList2000[0], timeList20000[0]], color='red', label = "Simple Meth
    plt.plot([200, 2000, 20000], [timeList200[1], timeList2000[1], timeList20000[1]],  color='green', label = "Zip Meth
    plt.plot([200, 2000, 20000], [timeList200[2], timeList2000[2], timeList20000[2]], color='blue', label = "Numpy Meth
    plt.xlabel('Matrix Size')
    plt.ylabel('Time in seconds')
    plt.title('Matrix Size vs Time for all Methods')
    plt.legend(numpoints=1)
    plt.show()
```

In [ ]:

In [ ]:

Question and Answers

Write up your results. What are the implications?
From the results that I have got, we can see that the zip method is the fastest when transposing a matrix
All the times for each method and transpose method are shown above

What is the time complexity of matrix transposition?(When memory and space is available)

- The simple method takes a lot of time. As the size of the matrix increases by the factor Xn the time increases by a factor of almost n. This tells us that simple method take O(n)
- For Zip method it is O(nlogn)
- Numpy method also increases by almost be a factor of Xn. It might be O(n)

Do your results confirm the theory?
Yes my results confirm the theory. Zip is the fastest with the lowest time complexity

In [ ]:

Present Slides Themes Help