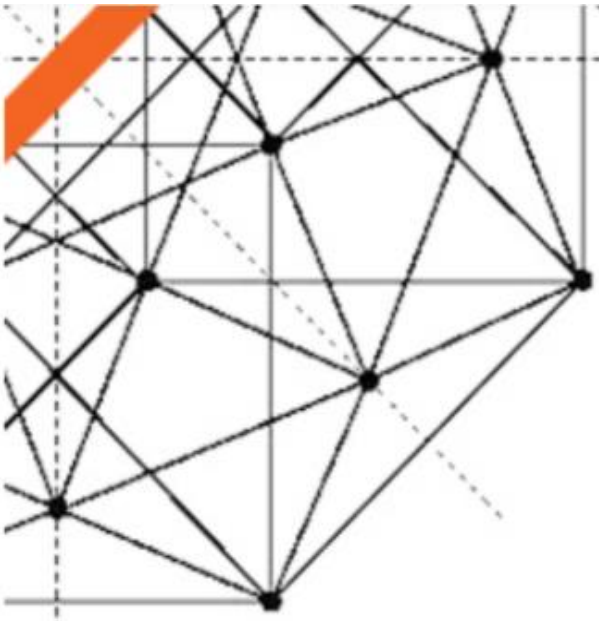

Matrices in Python


Coding and programming are around us in all aspects of our day to day lives. We are surrounded by electronic devices each running several numbers of software that run on a code of some kind. Learning to code, or having a basic understanding of it, will be vital in years to come in many different sectors of industry.

Python is one such programming language which is used by people every day. It is a powerful, multi-purpose programming language which has a simple easy-to-use syntax, making it perfect for beginner programmers. The number of uses Python has is nearly endless, there are so many different applications which can be carried out by the programming language.

One such application of python is associated with matrices. A matrix is a two-dimensional data structure where numbers are arranged into rows and columns. Python doesn't have a built-in type for matrices; however, you can treat list of a list as a matrix. For example:

```
A = [[1, 4, 5],
     [-5, 8, 9]]
```

However, Python also has a package for scientific computing which has support for a powerful N-dimensional array object – this package is known as NumPy. One of the key aspects of this package is that NumPY arrays have a fixed size at creation, and therefore (unlike Python lists), cannot grow dynamically without creating a new array and deleting the old one. An example can be seen below:

```
import numpy as np
a = np.array([1, 2, 3])
print(a)                    # Output: [1, 2, 3]
print(type(a))              # Output: <class 'numpy.ndarray'>
```

You can also use NumPy arrays to create matrices of more then one row:

Input:

```
8 import numpy as np
9 A = np.array([[1, 2, 3], [3, 4, 5]])
10 print(A)
```

Output:

```
In [1]: runfile('C:/Users/Olivia/Documents/3rd Year University
of Exeter/Maths Educational Project/NumPy.py', wdir='C:/Users/
Olivia/Documents/3rd Year University of Exeter/Maths
Educational Project')
[[1 2 3]
 [3 4 5]]
```

Matrix Addition and Subtraction

One of the most basic applications of Python for matrices is the addition and subtraction of them. This uses the concept of 'for' loops. A for loop is used when you have a block of code you want to repeat a fixed number of times. The statement iterates over the members of a sequence in order, executing the block each time.

The code for adding or subtracting 2 matrices looks like the following:

Input:

```
#addidng 2 by 2 matrix
A = [[1,2],
    [3,4]]

B = [[1,2],
    [3,4]]

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

for i in range(len(A)):
    for j in range(len(A[0])):
        answer[i][j] = A[i][j] + B[i][j]

print('A + B =')
for a in answer:
    print(a)

#subtracting 3 by 3 matrix
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]

Y = [[5,8,1],
    [6,7,3],
    [4,5,9]]

result = [[0,0,0],
          [0,0,0],
          [0,0,0]]

for i in range(len(X)):
   for j in range(len(X[0])):
       result[i][j] = X[i][j] - Y[i][j]

print('X - Y =')
for r in result:
   print(r)
```

Output:

```
A + B =
[2, 4]
[6, 8]
X - Y =
[7, -1, 2]
[-2, -2, 3]
[3, 3, 0]
```

Matrix Multiplication

As we saw earlier in this resource, you can use NumPy arrays to create matrices. NumPy has many inbuilt functions which allows the user to manipulate matrices. One such function is 'numpy.dot', which multiplies two matrices together using the dot product.

Input:

```
import numpy

X = numpy.array([[1, 2], [4, 5]])
Y = numpy.array([[7, 8], [9, 10]])

print ("XY : ")
print (numpy.dot(X,Y))
```

Output:

```
XY :
[[25 28]
 [73 82]]
```

Transposing a Matrix

Another application of Python is transposing a matrix. The transpose of a matrix is a new matrix whose rows are the columns of the original, and therefore making the columns of the new matrix the rows of the original.

Input:

```
#TRANSPOSING A 3x2 MATRIX

X = [[12,7],
     [4 ,5],
     [3 ,8]]

result = [[0,0,0],
          [0,0,0]]

# iterate through rows
for i in range(len(X)):
   # iterate through columns
   for j in range(len(X[0])):
       result[j][i] = X[i][j]

print('The transpose of X is:')

for r in result:
   print(r)


#transposing a 3x3 matrix

Y = [[1,5,7],
     [3 ,1 ,6],
     [8 ,9 ,5]]

result = [[0,0,0],
          [0,0,0],
          [0,0,0]]

for i in range(len(Y)):
    for j in range(len(Y[0])):
        result[j][i] = Y[i][j]

print('The transpose of Y is:')

for r in result:
    print(r)
```

Output:

```
The transpose of X is:
[12, 4, 3]
[7, 5, 8]
The transpose of Y is:
[1, 3, 8]
[5, 1, 9]
[7, 6, 5]
```

Rotation Matrix

To create and apply a rotation matrix using python, one solution is to use numpy. In linear algebra, a rotation matrix is a matrix that is used to perform a rotation in Euclidean space. Python allows us to apply a rotation matrix to a set of coordinates and determine the new points. These new points can subsequently be plotted onto a graph to establish what effect the rotation matrix has.

Input:

```
import numpy as np

m = np.array(( (1, 0),
               (0, -1) ))
print('rotation matrix:')
print(m)

x1 = np.array((0,1))
print('old coordinate1:')
print(x1)
print('new cooridnate1:')
print(m.dot(x1))

x2 = np.array((0,0))
print('old coordinate2:')
print(x2)
print('new cooridnate2:')
print(m.dot(x2))

x3 = np.array((1,0))
print('old coordinate3:')
print(x3)
print('new cooridnate3:')
print(m.dot(x3))

x4 = np.array((1,1))
print('old coordinate4:')
print(x4)
print('new cooridnate4:')
print(m.dot(x4))
```

Output:

```
rotation matrix:
[[ 1  0]
 [ 0 -1]]
old coordinate1:
[0 1]
new cooridnate1:
[ 0 -1]
old coordinate2:
[0 0]
new cooridnate2:
[0 0]
old coordinate3:
[1 0]
new cooridnate3:
[1 0]
old coordinate4:
[1 1]
new cooridnate4:
[ 1 -1]
```