GITAA
Transforming careers

Introduction to Python

Contents

- Numpy library
- Numpy operations
- Matrix
- Matrix operations
- Linear algebra

GITAA
Transforming careers

Numpy

- Numpy is a Python package
- It stands for 'Numerical Python'
- It is a library consisting of multidimensional array objects and a collection of routines for processing of array

Using Numpy, a developer can perform the following operations:

- Mathematical and logical operations on arrays
- Fourier transforms
- Operations related to linear algebra

Numpy array

An ordered collection of basic data types of given length

Code

```
import numpy as np
x = np.array([2,3,4,5])
print(type(x))
print(x)
```

Console Output

```
In [5]: import numpy as np

In [6]: x = np.array([2,3,4,5])

In [7]: print(type(x))
<class 'numpy.ndarray'>

In [8]: print(x)
[2 3 4 5]
```

Trying coercions on arrays

Code

```
print ('numpy can handle different catagorical entities ... \n')
import numpy as np
x = np.array([2,3,'n',5])
print(type(x))
print(x)
```

Console Output

```
In [9]: x = np.array([2,3,'n',5])

In [10]: print(type(x))
<class 'numpy.ndarray'>

In [11]: print(x)
['2' '3' 'n' '5']
```

All elements are coerced to same data type

Numpy – Statistical functions

Code

```
import numpy as np
a = np.array([[1,2,3],[3,4,5],[4,5,6]])
print('Our array is:\n',a)
```

Console Output

```
Our array is:
 [[1 2 3]
 [3 4 5]
 [4 5 6]]
```

Finding the sum:

```
a.sum()
np.sum(a)
33
```

Entire array is summed

Finding the sum: (along axis = 0)

```
np.sum(a, axis = 0)
array([ 8, 11, 14])
```

sum of the entries across the rows

Finding the sum: (along axis = 1)

```
np.sum(a, axis = 1)
array([ 6, 12, 15])
```

sum of the entries across the columns

Other functions

Please try other statistical functions such as:

- Mean – *mean()*
- Variance – *var()*
- Standard deviation – *std()*

Matrices

- Rectangular arrangement of numbers in rows and columns
- Rows run horizontally and columns run vertically

$$\begin{pmatrix} 1 & 5 & 3 \\ 4 & 9 & 2 \\ 5 & 6 & 7 \end{pmatrix} \qquad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 4 & 5 \end{bmatrix}$$

Matrices

Creating a matrix

Code

```
import numpy as np
m1 = np.matrix("1,2,3;4,5,6;7,8,9")
print(m1)
```

Console Output

```
In [1]: import numpy as np

In [2]: m1 = np.matrix("1,2,3;4,5,6;7,8,9")

In [3]: print(m1)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Matrices

Size of Matrix

IN THE EXAMPLE : CONTINUE FROM PREVIOUS CODE

Code

```
# Information from matrix
# Dimension of matrix
m1.shape
# No. of elements in matrix
m1.size
# To find the rows and columns
m1.shape[0]
m1.shape[1]
```

Console Output

```
m1.shape
(3, 3)
```

```
m1.size
9
```

```
m1.shape[0]
3

m1.shape[1]
3
```

Inserting row/column to a matrix

Code

```
import numpy as np
T = np.matrix(np.arange(0,20)).reshape(5,4)
print(T)
```

Console Output

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

Inserting row/column to a matrix

Creating values for a new column

Code

```
c = np.matrix("3,9,0,8,8")
print(c)
```

Console Output

```
[[3 9 0 8 8]]
```

Creating values for a new row

Code

```
r = np.matrix("7,9,8,9")
print(r)
```

Console Output

```
[[7 9 8 9]]
```

Inserting row/column to a matrix

Adding a new column

Code

```
np.insert(T,0,c,axis = 1)
```

Console Output

```
matrix([[ 3,  0,  1,  2,  3],
        [ 9,  4,  5,  6,  7],
        [ 0,  8,  9, 10, 11],
        [ 8, 12, 13, 14, 15],
        [ 8, 16, 17, 18, 19]])
```

Adding a new row

Code

```
np.insert(T,0,r,axis = 0)
```

Console Output

```
matrix([[ 7,  9,  8,  9],
        [ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11],
        [12, 13, 14, 15],
        [16, 17, 18, 19]])
```

Matrix operations

Matrix operations in python

- Addition
- subtraction
- Matrix Multiplication
- Matrix Division
- Matrix operations

Matrices – Arithmetic operations - I

Create matrix n1 & n2

$$n1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 8 & 9 & 1 \end{bmatrix}_{3\times 3}$$

$$n2 = \begin{bmatrix} 3 & 1 & 3 \\ 4 & 2 & 1 \\ 5 & 1 & 2 \end{bmatrix}_{3\times 3}$$

Matrix addition

```
print(np.add(n1,n2))
```

```
[[ 4  3  6]
 [ 8  7  7]
 [13 10  3]]
```

Matrix subtraction

```
print(np.subtract(n1,n2))
```

```
[[-2  1  0]
 [ 0  3  5]
 [ 3  8 -1]]
```

Matrix – Arithmetic operations - II

$$n1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 8 & 9 & 1 \end{bmatrix}_{3\times 3}$$

$$n2 = \begin{bmatrix} 3 & 1 & 3 \\ 4 & 2 & 1 \\ 5 & 1 & 2 \end{bmatrix}_{3\times 3}$$

Matrix multiplication

```
print(np.dot(n1,n2))
```

```
[[26  8 11]
 [62 20 29]
 [65 27 35]]
```

Matrix division

```
print(np.divide(n1,n2))
```

```
[[ 0.33333333  2.          1.        ]
 [ 1.          2.5         6.        ]
 [ 1.6         9.          0.5       ]]
```

Matrix operations

Accessing and editing Matrices

Convention:

- **Array/value before "," for accessing rows**
- **Array/value after "," for accessing columns**
- **Use of numpy's delete command for removing rows/columns**

$$m1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}_{3\times3}$$

Accessing data in matrices

$$m1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}_{3\times 3}$$

Code	Console Output
`print(m1[1,:])`	`[[4 5 6]]`
`print(m1[1,2])`	`6`

`print(m1[:,2])`	`[[3]`
	` [6]`
	` [9]]`

Editing matrices

$$m1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}_{3\times 3}$$

Code

```
m1[1,1] = -3

m1[0,2] = 16

print(m1)
```

Console Output

```
[[ 1  2 16]
 [ 4 -3  6]
 [ 7  8  9]]
```

Matrix – Linear algebra

GITAA
Transforming careers

Linear algebra operations in Python : -

- Determinant of matrix
- Rank of matrix
- Eigen values & vectors
- Solving system of equations
- Inverse of matrix

Matrix – Linear algebra

Code

```
import numpy as np
m1 = np.matrix("0,1,2;3,4,5;6,7,8")
print(m1)
```

Console Output

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Determinant of matrix

```
det_matrix = np.linalg.det(m1)
print('\n',"Determinent of matrix m1:",det_matrix)
```

```
Determinent of matrix m1: 0.0
```

Rank of matrix

```
rank_matrix = np.linalg.matrix_rank(m1)
print("Rank of the matrix m1:", rank_matrix)
```

```
Rank of the matrix m1: 2
```

Matrix – Linear algebra

Eigen values and vectors of matrix

```
In [6]: eig_val, eig_vect = np.linalg.eig(m1)  # Gives
eigen values and vectors

In [7]: print(eig_val,"\n",eig_vect)
[  1.33484692e+01  -1.34846923e+00  -1.15433316e-15]
 [[ 0.16476382  0.79969966  0.40824829]
 [ 0.50577448  0.10420579 -0.81649658]
 [ 0.84678513 -0.59128809  0.40824829]]
```

Eigen values

Eigen vectors

Eigen values only

```
In [4]: Value1 = np.linalg.eigvals(m1) # to find the
values only

In [5]: print(Value1)
[  1.33484692e+01  -1.34846923e+00  -1.15433316e-15]
```

Matrix – Linear algebra

Solving a system of equations

- $3x + y + 2z = 2$
- $3x + 2y + 5z = -1$
- $6x + 7y + 8z = 3$

- $$\begin{pmatrix} 3 & 1 & 2 \\ 3 & 2 & 5 \\ 6 & 7 & 8 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{matrix} 2 \\ -1 \\ 3 \end{matrix}$$

Matrix – Linear algebra

Code

```
import numpy as np
M = np.matrix("3,1,2;3,2,5;6,7,8")
print(M)
```

```
B = np.matrix("2,-1,3").transpose()
print(B)
```

```
solve = np.linalg.solve(M,B)
print(solve)
```

Console Output

```
[[3 1 2]
 [3 2 5]
 [6 7 8]]
```

```
[[ 2]
 [-1]
 [ 3]]
```

```
[[ 1.24242424]
 [ 0.81818182]
 [-1.27272727]]
```

Matrix – Linear algebra

Code

```
import numpy as np
M = np.matrix("3,1,2;3,2,5;6,7,8")
print(M)
```

Console Output

```
[[3 1 2]
 [3 2 5]
 [6 7 8]]
```

Inverse of matrix

```
# Inverse of a matrix
Inv = np.linalg.inv(M)
print(Inv)
```

```
[[ 0.57575758 -0.18181818 -0.03030303]
 [-0.18181818 -0.36363636  0.27272727]
 [-0.27272727  0.45454545 -0.09090909]]
```

Eigenvalue decomposition

- Matrix decompositions are a useful tool for reducing a range of complex operations.
- Commonly used decomposition method is the eigen decomposition that decomposes a matrix into eigenvectors and eigenvalues.
- This decomposition plays a role in machine learning such as in the Principal Component Analysis(PCA).

Eigenvalue decomposition - I

Code

```
import numpy as np
A = np.matrix([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])
print(A)
```

Console Output

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
values, vectors = np.linalg.eig(A)
```

```
print(values)
```

```
[ 1.61168440e+01  -1.11684397e+00  -9.75918483e-16]
```

```
print(vectors)
```

```
[[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
```

Eigenvalue decomposition - II

Code

Console Output

```
Q = vectors
print(Q)
```

```
[[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
```

```
R = np.linalg.inv(Q)
print(R)
```

```
[[-0.48295226 -0.59340999 -0.70386772]
 [-0.91788599 -0.24901003  0.41986593]
 [ 0.40824829 -0.81649658  0.40824829]]
```

Eigenvalue decomposition - III

Code

Console Output

```
L = np.diag(values)
print(L)
```

```
[[  1.61168440e+01   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00  -1.11684397e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00  -9.75918483e-16]]
```

```
B = Q.dot(L).dot(R)
print(B)
```

```
[[ 1.   2.   3.]
 [ 4.   5.   6.]
 [ 7.   8.   9.]]
```

THANK YOU

GITAA
Transforming careers