Use Of Numpy For Matrix Operations

Matrices-

Matrices are used as a mathematical tool for a variety of purposes in the real world.

NumPy-

It is a Python library allowing easy numerical calculations involving single and multidimensional arrays and matrices. It excels in performing numerical calculations.

NumPy provides-

- a powerful N-dimensional array object called as ndarray
- Broadcasting functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Creating a matrix in NumPy-

→ **Creating a matrix using lists-**

```
## Import numpy
import numpy as np

## Create a 2D numpy array using python lists
arr = np.array([[ 1, 2, 3],[ 4, 5, 6]])
print(arr)
```

Output-

```
[[1 2 3]
 [4 5 6]]
```

It represents a 2D matrix where input to np.array() is a list of lists [[1, 2, 3],[4, 5, 6]] . Each list in the parent list forms a row in the matrix. np.array is used to create NumPy array from a list. NumPy arrays are of type ndarray.

→ **Creating matrix using ranges-**

np.arange() can generate a sequence of numbers given the start and end.

```
print(np.arange(0,5))
```

Output-

```
[0 1 2 3 4]
```

Use Of Numpy For Matrix Operations

To generate 2D matrix we can use np.arange() inside a list. We pass this list into np.array() which makes it a 2D NumPy array.

```
1 print(np.array([np.arange(0,5), np.arange(5,10)]))
```

Output-

```
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

→ **Shape of NumPy array-**

```
arr_2d = np.array([np.arange(0,5), np.arange(5,10)])
print(arr_2d.shape)
```

Output-

```
(2, 5)
```

(2, 5) means that the matrix has 2 rows and 5 columns.

→ **Matrix filled with zeros and ones-**

↵ Filling with zeros:

```
print(np.zeros((3, 4)))
```

Output-

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

↵ Filling with ones:

```
print(np.ones((2, 2), dtype=np.int16))
```

Output-

```
[[1 1]
 [1 1]]
```

Use Of Numpy For Matrix Operations

Matrix Operations –

→ Addition-

```
import numpy as np
## Generate two matrices
mat_2d_1 = np.array([np.arange(0,3), np.arange(3,6)])
mat_2d_2 = np.array([np.arange(6,9), np.arange(9,12)])</pre>
print("Matrix1: n ", mat_2d_1)
print("Matrix2: n ", mat_2d_2)

## Add 1 to each element in mat_2d_1 and print it
print("Scalar addition: n ", mat_2d_1 + 1)

## Add two matrices above elementwise
print("Element wise addition of two matrices of same size: n ", mat_2d_1 + mat
```

Output-

```
Matrix1:
   [[0 1 2]
  [3 4 5]]
Matrix2:
   [[ 6  7  8]
  [ 9 10 11]]
Scalar addition:
   [[1 2 3]
  [4 5 6]]
Element wise addition of two matrices of same size:
   [[ 6  8 10]
  [12 14 16]]
```

→ Subtraction-

```
import numpy as np

## Generate two matrices
mat_2d_1 = np.array([np.arange(0,3), np.arange(3,6)])
mat_2d_2 = np.array([np.arange(6,9), np.arange(9,12)])

print("Matrix1: n ", mat_2d_1)
print("Matrix2: n ", mat_2d_2)

## Subtract 1 from each element in mat_2d_1 and print it
print("Scalar addition: n ", mat_2d_1 - 1)

## Subtract two matrices above elementwise
print("Element wise subtraction of two matrices of same size: n ", mat 2d 1
```

Output-

```
Matrix1:
   [[0 1 2]
  [3 4 5]]
Matrix2:
   [[ 6  7  8]
  [ 9 10 11]]
Scalar addition:
   [[-1  0  1]
  [ 2  3  4]]
Element wise subtraction of two matrices of same size:
   [[-6 -6 -6]
  [-6 -6 -6]]
```

Use Of Numpy For Matrix Operations

→ **Product-**

```python
import numpy as np

## Generate two matrices of shape (2,3) and (3,2) so that we can find
## dot product
mat_2d_1 = np.array([np.arange(0,3), np.arange(3,6)])
mat_2d_2 = np.array([np.arange(0,2), np.arange(2,4), np.arange(4,6)])

## Print shapes and matrices
print("Matrix1: n ", mat_2d_1)
print("Matrix1 shape: n", mat_2d_1.shape)
print("Matrix2: n ", mat_2d_2)
print("Matrix2 shape: n", mat_2d_2.shape)

## Multiply each element by 2 in mat_2d_1 and print it
print("Scalar Product: n ", mat_2d_1 * 2)

## Find product of two matrices above using dot product
print("Dot Product: n ", np.dot(mat_2d_1, mat_2d_2))
```

Output-

```
Matrix1:
  [[0 1 2]
 [3 4 5]]
Matrix1 shape:
 (2, 3)
Matrix2:
  [[0 1]
 [2 3]
 [4 5]]
Matrix2 shape:
 (3, 2)
Scalar Product:
  [[ 0  2  4]
 [ 6  8 10]]
Dot Product:
  [[10 13]
 [28 40]]
```

→ **Division-**

```python
import numpy as np

## Generate a matrix of shape (2,3)
mat_2d = np.array([np.arange(0,3), np.arange(3,6)])

## Print the matrix
print("Matrix: n ", mat_2d)

## Element wise division by scalar
print("Scalar Division: n ", mat_2d / 2)
```

Output-

```
Matrix:
  [[0 1 2]
 [3 4 5]]
Scalar Division:
  [[0.  0.5 1. ]
 [1.5 2.  2.5]]
```

Use Of Numpy For Matrix Operations

→ **Exponent-**

```
import numpy as np

## Generate a matrix of shape (2,3)
mat_2d = np.array([np.arange(0,3), np.arange(3,6)])

## Print the matrix
print("Matrix: n ", mat_2d)

## Find exponent element wise i.e. raise each element in matrix to power 2
print("Matrix raised to power of 2: n ", mat_2d ** 2)
```

Output-

```
Matrix:
  [[0 1 2]
 [3 4 5]]
Matrix raised to power of 2:
  [[ 0  1  4]
 [ 9 16 25]]
```

→ **Transpose-**

```
import numpy as np

## Generate a matrix of shape (2,3)
mat_2d = np.array([np.arange(0,3), np.arange(3,6)])

## Print the matrix
print("Matrix: n ", mat_2d)

## Matrix Transpose
print("Transpose n ", mat_2d.T)
```

Output-

```
Matrix:
  [[0 1 2]
 [3 4 5]]
Transpose
  [[0 3]
 [1 4]
 [2 5]]
```

Use Of Numpy For Matrix Operations

→ **Slicing a Matrix-**

```
import numpy as np
# Create a matrix
mat_2d = np.array([np.arange(0,3), np.arange(3,6)])
print("Matrix: n", mat_2d)

# Slice to get second row in matrix
print("Sliced: n ", mat_2d[1:, :])
```

Output-

```
Matrix:
 [[0 1 2]
 [3 4 5]]
Sliced:
  [[3 4 5]]
```