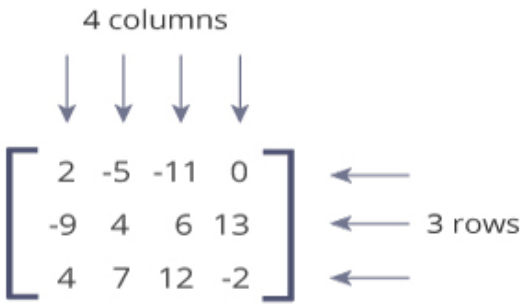
Use of Numpy For Matrix Operations

- **Matrix-** A matrix is a two-dimensional data structure where numbers are arranged into rows and columns.
  For example:



This is a 3x4 (pronounced "three by four") matrix because it has 3 rows and 4 columns.

- **Python Matrix -** Python doesn't have a built-in type for matrices. However, we can treat list of a list as a matrix.
  For example:

```
In [1]: A = [[1, 4, 5],
            [-5, 8, 9]]
        print(A)
```

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

- **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

1. Creating a matrix in NumPy-

- **Creating a matrix using lists:**

```
In [2]: import numpy as np

        A = np.array([[1, 2, 3], [3, 4, 5]])
        print(A)

        A = np.array([[1.1, 2, 3], [3, 4, 5]]) # Array of floats
        print(A)

        A = np.array([[1, 2, 3], [3, 4, 5]], dtype = complex) # Array of complex numbers
        print(A)
```

```
[[1 2 3]
 [3 4 5]]
[[1.1 2.  3. ]
 [3.  4.  5. ]]
[[1.+0.j 2.+0.j 3.+0.j]
 [3.+0.j 4.+0.j 5.+0.j]]
```

Use of Numpy For Matrix Operations

- **Creating matrix using arange() and shape():**

```
In [3]: import numpy as np

        A = np.arange(4)
        print('A =', A)

        B = np.arange(12).reshape(2, 6)
        print('B =', B)
```

```
A = [0 1 2 3]
B = [[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]
```

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

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

        zeors_array = np.zeros( (2, 3) )
        print(zeors_array)
```

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

```
In [7]: ones_array = np.ones( (1, 5), dtype=np.int16 )
        print(ones_array)
```

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

2. Matrix Operations – It consists of Addition of two matrices, Multiplication of two matrices and Transpose of a matrix.

- **Addition of Two Matrices :**

```
In [8]: import numpy as np

        A = np.array([[2, 4], [5, -6]])
        B = np.array([[9, -3], [3, 6]])
        C = A + B      # element wise addition
        print(C)
```

```
[[11  1]
 [ 8  0]]
```

- **Multiplication of Two Matrices :**

```
In [9]: import numpy as np

        A = np.array([[3, 6, 7], [5, -3, 0]])
        B = np.array([[1, 1], [2, 1], [3, -3]])
        C = A.dot(B)
        print(C)
```

```
[[ 36 -12]
 [ -1   2]]
```

Use of Numpy For Matrix Operations

- **Transpose of a Matrix :**

```
In [10]: import numpy as np

A = np.array([[1, 1], [2, 1], [3, -3]])
print(A.transpose())
```

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

3. Access matrix elements, rows and columns- Similar like lists, we can access matrix elements using index.

- **Access matrix elements:**
  This is an example of accessing elements in 1D matrix.

```
In [11]: import numpy as np
A = np.array([2, 4, 6, 8, 10])

print("A[0] =", A[0])     # First element
print("A[2] =", A[2])     # Third element
print("A[-1] =", A[-1])   # Last element
```

```
A[0] = 2
A[2] = 6
A[-1] = 10
```

This is an example of accessing elements in 2D matrix.

```
In [12]: import numpy as np

A = np.array([[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]])

#  First element of first row
print("A[0][0] =", A[0][0])

# Third element of second row
print("A[1][2] =", A[1][2])

# Last element of last row
print("A[-1][-1] =", A[-1][-1])
```

```
A[0][0] = 1
A[1][2] = 9
A[-1][-1] = 19
```

- **Access rows of a Matrix:**

```
In [13]: import numpy as np

A = np.array([[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]])

print("A[0] =", A[0]) # First Row
print("A[2] =", A[2]) # Third Row
print("A[-1] =", A[-1]) # Last Row (3rd row in this case)
```

```
A[0] = [ 1  4  5 12]
A[2] = [-6  7 11 19]
A[-1] = [-6  7 11 19]
```

Use of Numpy For Matrix Operations

- **Access columns of a Matrix:**

In [14]:
```
import numpy as np

A = np.array([[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]])

print("A[:,0] =",A[:,0]) # First Column
print("A[:,3] =", A[:,3]) # Fourth Column
print("A[:,-1] =", A[:,-1]) # Last Column (4th column in this case)
```

```
A[:,0] = [ 1 -5 -6]
A[:,3] = [12  0 19]
A[:,-1] = [12  0 19]
```

4. Slicing of a Matrix- Slicing of a one-dimensional NumPy array is similar to a list.

- **Slicing of 1-D Matrix**

In [16]:
```
import numpy as np
letters = np.array([1, 3, 5, 7, 9, 7, 5])

# 3rd to 5th elements
print(letters[2:5])

# 1st to 4th elements
print(letters[:-5])

# 6th to last elements
print(letters[5:])

# 1st to last elements
print(letters[:])

# reversing a list
print(letters[::-1])
```

```
[5 7 9]
[1 3]
[7 5]
[1 3 5 7 9 7 5]
[5 7 9 7 5 3 1]
```

- **Slicing of 2-D Matrix**

In [17]:
```
import numpy as np

A = np.array([[1, 4, 5, 12, 14],
    [-5, 8, 9, 0, 17],
    [-6, 7, 11, 19, 21]])

print(A[:2, :4])
```

```
[[ 1  4  5 12]
 [-5  8  9  0]]
```