NumPy is the foundation of the Python machine learning stack.

NumPy allows for efficient operations on the data structures often used in machine learning.

The data strucutures are vectors, matrices, arrays.

In [20]:

```
#You need to create a vector.
#Use NumPy to create a one-dimensional array
import numpy as np

vector_row = np.array([[1,2,3]])
print(vector_row)
vector_col =np.array([[1],[2],[3]])
print(vector_col)
vector_col


```

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

Out[20]:

```
array([[1],
       [2],
       [3]])
```

In [10]:

```python
#Creating a two dimensional Matrix
import numpy as np

matrix = np.array([[1,2],[3,4],[5,6]])
print(matrix)

# other way with mat() method
matrix = np.mat([[1,2],[3,4],[5,6]])
matrix
```

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

Out[10]:

```
matrix([[1, 2],
        [3, 4],
        [5, 6]])
```

In [38]:

```
#Selecting elements
import numpy as np

vector_row = np.array([1,2,3])
#select specific element
print(vector_row[2])

#select all elements upto index 2
print(vector_row[:2])

#select all elements
print(vector_row[:])

matrix = np.array([[1,2,3],
                   [4,5,6],
                   [7,8,9]])
#select element of third row and second column
print(matrix[2,1])
#Select first two rows and all columns
print(matrix[:2,:])
#Select all rows and the second column
print(matrix[:,1:2])


```

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

In [42]:

```
#Describing a Matrix in terms of shape,size and dimensions

import numpy as np
#create matrix
matrix = np.array([[1,2,3],
                  [4,5,6],
                  [7,8,9]])

# View number of rows and columns
print(matrix.shape)

# View number of elements (rows * columns)
print(matrix.size)

# View number of dimensions
print(matrix.ndim)

```

```
(3, 3)
9
2
```

NumPy’s vectorize class converts a function into a function that can apply to all elements in an array or slice of an array

In [6]:

```
#You want to apply some function to multiple elements in an array.
#Use NumPy's vectorize

# Load library
import numpy as np

# Create matrix
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])

# Create function that adds 100 to something
add_100 = lambda i: i + 100

# Create vectorized function
vectorized_add_100 = np.vectorize(add_100)

# Apply function to all elements in matrix
print(vectorized_add_100(matrix))
print()

#One more simple way
new_matrix = matrix+100
print(new_matrix)
```

```
[[101 102 103]
 [104 105 106]
 [107 108 109]]

[[101 102 103]
 [104 105 106]
 [107 108 109]]
```

In [10]:

```python
# Finding the Maximum and Minimum Values

# Load library
import numpy as np
# Create matrix
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])

# Return maximum element
print(np.max(matrix))

# Return minimum element
print(np.min(matrix))

# Find maximum element in each row axis = 0 is for row
print(np.max(matrix, axis=0))

# Find maximum element in each column axis = 1 is for column
print(np.max(matrix, axis=1))
```

```
9
1
[7 8 9]
[3 6 9]
```

In [15]:

```python
# Calculating the Average, Variance, and Standard Deviation

# Load library
import numpy as np

# Create matrix
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])


#Return Average
print("Average:",np.average(matrix))

# Return mean
print("Mean:",np.mean(matrix))

# Return variance
print("Variance:",np.var(matrix))

# Return standard deviation
print("Standard Deviation:",np.std(matrix))

```

```
Average: 5.0
Mean: 5.0
Variance: 6.666666666666667
Standard Deviation: 2.581988897471611
```

In [22]:

```python
#Transposing a Vector or Matrix by using T method

# Load library
import numpy as np

# Create matrix
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
#Create Vector
vector = np.array([[1,2,3,4,5]])

# Transpose matrix
print(matrix.T)
print("\n")
#Transpose Vector
print(vector.T)

```

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


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

In [23]:
```python
#Flattening a Matrix means transform a matrix into a one-dimensional array by using flatten().

# Load library
import numpy as np
# Create matrix
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
# Flatten matrix
matrix.flatten()

```

Out[23]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [29]:
```python
#Finding the Rank of a Matrix.
#Use NumPy's linear algebra method matrix_rank.


# Load library
import numpy as np
# Create matrix
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])

#Rank of a matrix
print("Rank of Matrix:",np.linalg.matrix_rank(matrix))


```

```
Rank of Matrix: 2
```

In [31]:

```python
# Getting the Diagonal elements of a Matrix using diagonal method
#Getting trace of the matrix. The trace of a matrix is the sum of the diagonal elements

# Load library
import numpy as np

# Create matrix
matrix = np.array([[1, 2, 3],
                    [2, 4, 6],
                    [3, 8, 9]])

# Return diagonal elements
print(matrix.diagonal())

#Return the trace
print(matrix.trace())
```

```
[1 4 9]
14
```

In [32]:

```
#operations of two matrices. Addition, Subtraction, Multiplication.

# Load library
import numpy as np

# Create matrix
matrix_a = np.array([[1, 1, 1],
                     [1, 1, 1],
                     [1, 1, 2]])

# Create matrix
matrix_b = np.array([[1, 3, 1],
                     [1, 3, 1],
                     [1, 3, 8]])

# Add two matrices
print(np.add(matrix_a, matrix_b))

# Subtract two matrices
print(np.subtract(matrix_a, matrix_b))

# Multiply two matrices
print(np.dot(matrix_a, matrix_b))

```

```
[[ 2  4  2]
 [ 2  4  2]
 [ 2  4 10]]
[[ 0 -2  0]
 [ 0 -2  0]
 [ 0 -2 -6]]
[[ 3  9 10]
 [ 3  9 10]
 [ 4 12 18]]
```

In [33]:

```python
# Calculating inverse of a matrix using NumPy's linear algebra inv method.

# Load library
import numpy as np

# Create matrix
matrix = np.array([[1, 4],
                   [2, 5]])

# Calculate inverse of matrix
print(np.linalg.inv(matrix))


```

```
[[-1.66666667  1.33333333]
 [ 0.66666667 -0.33333333]]
```

In [38]:

```python
#Generating Random Values

# Load library
import numpy as np

# Generate three random floats between 0.0 and 1.0
print(np.random.random(10))

# Generate three random integers between 1 and 10
print(np.random.randint(0, 11, 3))

# Draw three numbers greater than or equal to 1.0 and less than 2.0
print(np.random.uniform(1.0, 2.0, 3))
```

```
[0.71998631 0.74220332 0.91882157 0.99343941 0.02097137 0.42885785
 0.71807372 0.82008064 0.78739958 0.3929567 ]
[10  8  2]
[1.02039526 1.74755115 1.30519769]
```