Numpy in Python

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

It is also useful in linear algebra, random number capability etc. NumPy array can also be used as an efficient multi-dimensional container for generic data.

NumPy Array: Numpy array is a powerful N-dimensional array object which is in the form of rows and columns. We can initialize numpy arrays from nested Python lists and access it elements. In order to perform these numpy operations.

Example: Creating a single and a two dimensional numpy array

```
import numpy as np

a=np.array([1,2,3]) #single dimensional array

b = np.array([(4,5,6),(7,8,9)])

print(a)

print(b)
```

O/P:

```
[1 2 3]


[ [4 5 6]
 [7 8 9] ]
```

Python NumPy Operations

ndim:

it is used for finding whether the array is one dimensional or two domensional.

```
import numpy as np

a = np.array([(1,2,3),(4,5,6)])
```

```
print(a.ndim)

O/P : 2
```

Example : Demonstrating various numpy functions and operations

```
import numpy as np

a = np.array([1, 2, 5, 3])

# add 1 to every element

print ("Adding 1 to every element:", a+1)

# subtract 3 from each element

print ("Subtracting 3 from each element:", a-3)

# multiply each element by 10

print ("Multiplying each element by 10:", a*10)

# square each element

print ("Squaring each element:", a**2)

# modify existing array

a *= 2

print ("Doubled each element of original array:", a)

# transpose of array

a = np.array([[1, 2, 3], [3, 4, 5], [9, 6, 0]])

print ("\nOriginal array:\n", a)

print ("Transpose of array:\n", a.T)
```

Example : Sorting a numpy array

```
import numpy as np

a = np.array([[1, 4, 2],

              [3, 4, 6],

              [0, -1, 5]])

# sorted array
```

```
print ("Array elements in sorted order:\n",
                np.sort(a, axis = None))
# sort array row-wise
print ("Row-wise sorted array:\n",
            np.sort(a, axis = 1))

# specify sort algorithm
print ("Column wise sort by applying merge-sort:\n",
         np.sort(a, axis = 0, kind = 'mergesort'))

# Example to show sorting of structured array
# set alias names for dtypes
dtypes = [('name', 'S10'), ('grad_year', int), ('cgpa', float)]

# Values to be put in array
values = [('Hrithik', 2009, 8.5), ('Ajay', 2008, 8.7),
        ('Pankaj', 2008, 7.9), ('Aakash', 2009, 9.0)]

# Creating array
arr = np.array(values, dtype = dtypes)
print ("\nArray sorted by names:\n",
         np.sort(arr, order = 'name'))

print ("Array sorted by grauation year and then cgpa:\n",
            np.sort(arr, order = ['grad_year', 'cgpa']))
```

O/P:

```
Array elements in sorted order:
```

```
[-1  0  1  2  3  4  4  5  6]

Row-wise sorted array:

[[ 1  2  4]

 [ 3  4  6]

 [-1  0  5]]

Column wise sort by applying merge-sort:

[[ 0 -1  2]

 [ 1  4  5]

 [ 3  4  6]]


Array sorted by names:

[('Aakash', 2009, 9. ) ('Ajay', 2008, 8.7) ('Hrithik', 2009, 8.5)

 ('Pankaj', 2008, 7.9)]

Array sorted by grauation year and then cgpa:

[('Pankaj', 2008, 7.9) ('Ajay', 2008, 8.7) ('Hrithik', 2009, 8.5)

 ('Aakash', 2009, 9. )]
```

Pandas in Python

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Pandas deals with the following three data structures −

- Series
- DataFrame
- Panel

Series

Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56, …

10	**23**	**56**	**17**	**52**	**61**	**73**	**90**	**26**	**72**

DataFrame

DataFrame is a two-dimensional array with heterogeneous data. For example,

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

Panel

Panel is a three-dimensional data structure with heterogeneous data. It is hard to represent the panel in graphical representation. But a panel can be illustrated as a container of DataFrame.

Scikit in Python

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis

The library is focused on modeling data. It is not focused on loading, manipulating and summarizing data. For these features, refer to NumPy and Pandas.

Some popular groups of models provided by scikit-learn include:

- **Clustering**: for grouping unlabeled data such as KMeans.
- **Cross Validation**: for estimating the performance of supervised models on unseen data.
- **Datasets**: for test datasets and for generating datasets with specific properties for investigating model behavior.
- **Dimensionality Reduction**: for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
- **Ensemble methods**: for combining the predictions of multiple supervised models.
- **Feature extraction**: for defining attributes in image and text data.
- **Feature selection**: for identifying meaningful attributes from which to create supervised models.
- **Parameter Tuning**: for getting the most out of supervised models.
- **Manifold Learning**: For summarizing and depicting complex multi-dimensional data.
- **Supervised Models**: a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.