Python Libraries for Data Science

By Vaishali KUNJIR

Python Libraries for Data Science

- NumPy(Numerical Python)
- Pandas
- Matplotlib
- SciPy (Scientific Python)
- Scikit-learn

Data Science Python Libraries Applications

There are multiple stages in the data science process, and different libraries are used to help with each stage. Following are the statges of data science process:

- Extract, transform, load (ETL)
- Data exploration
- Data evaluation
- Data modeling
- Data presentation (or visualization)

NumPy
(https://numpy.org/devdocs/user/index.html#user)

What is Numpy?

1

Efficient scientific computing

Numpy is an open-source numerical computing library for Python. It enables fast and efficient computation with arrays and matrices.

2

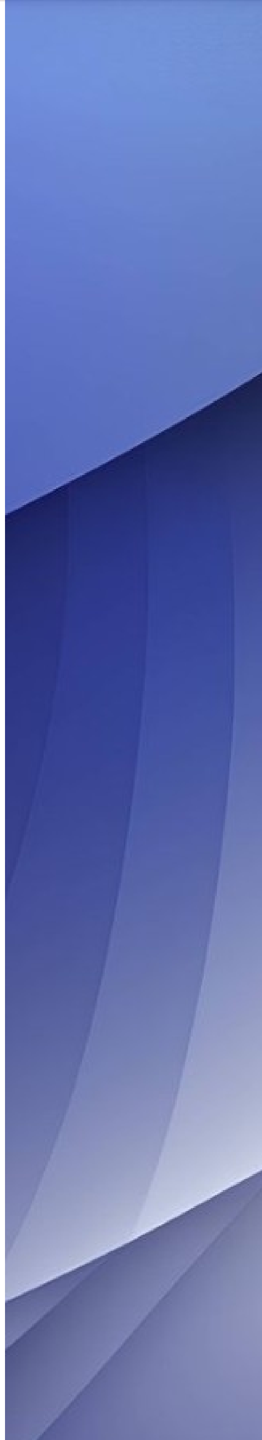
Central to data analysis

Numpy provides the backbone for many data analysis libraries, making it the foundation for several machine learning models and algorithms.

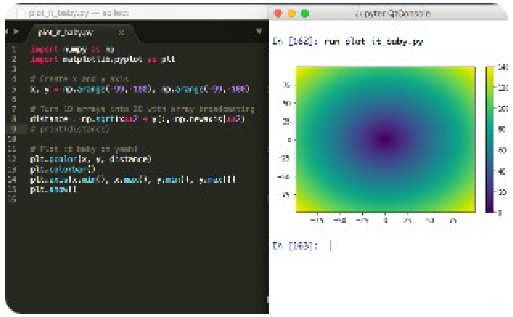
3

Trusted by the scientific community

Numpy has been actively developed and refined for over 15 years, making it a powerful tool for scientific computing and research.



Key Features of Numpy



```
import numpy as np
import matplotlib.pyplot as plt

# Create x and y axis
x, y = np.arange(-99, 100), np.arange(-99, 100)

# Turn 1D arrays into 2D with array broadcasting
distance = np.sqrt(x**2 + y[:, np.newaxis]**2)
# print(distance)

# Plot it baby in yeah!
plt.pcolor(x, y, distance)
plt.colorbar()
plt.axis([x.min(), x.max(), y.min(), y.max()])
plt.show()
```

```
In [162]: run plot_it_baby.py

In [163]: |
```

Arrays and Matrices

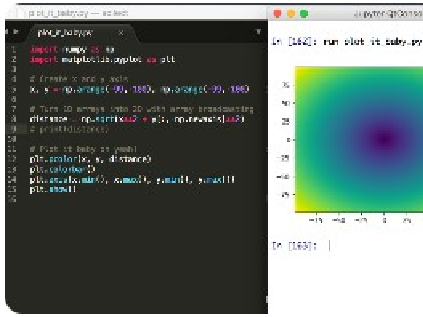
Numpy provides powerful data structures for performing mathematical operations on arrays

Mathematical Operations

Numpy offers a comprehensive set of mathematical functions for performing complex operations on arrays and matrices.

Random Number Generation

Numpy provides several functions for generating arrays with random numbers, which is useful in many applications such as



```
import numpy as np
import matplotlib.pyplot as plt

# Create x and y axis
x, y = np.arange(-99, 100), np.arange(-99, 100)

# Turn 1D arrays into 2D with array broadcasting
distance = np.sqrt(x**2 + y[:, np.newaxis]**2)
# print(distance)

# Plot it baby in yeah!
plt.pcolor(x, y, distance)
plt.colorbar()
plt.axis([x.min(), x.max(), y.min(), y.max()])
plt.show()
```

```
In [162]: run plot_it_baby.py

In [163]: |
```

Broadcasting

Numpy's broadcasti feature improves co readability and enab us to perform matrix operations without actually creating a second matrix.

Common Numpy Functions and Examples

Creating Arrays and Matrices

Numpy makes creating and initializing arrays and matrices a breeze. For example, you can create an array filled with zeros with np.zeros((3,3)) or an identity matrix with np.eye(3).

Indexing and Slicing Arrays

Numpy offers advanced indexing and slicing features, enabling us to extract subsets of data based on specific criteria.

Statistical Operations with Numpy

Numpy offers several statistical functions, including mean, median, variance, and standard

Practical Applications of Numpy

1

Using Numpy for Data Analysis

Numpy is an essential tool for data analysis in fields ranging from finance to scientific research. You can use Numpy to perform computations on large datasets and perform advanced statistical analysis quickly and efficiently.

2

Implementing Machine Learning Algorithms with Numpy

Numpy's powerful array and matrix operations make it a robust tool for implementing machine learning models, such as linear regression, logistic regression, and neural networks.

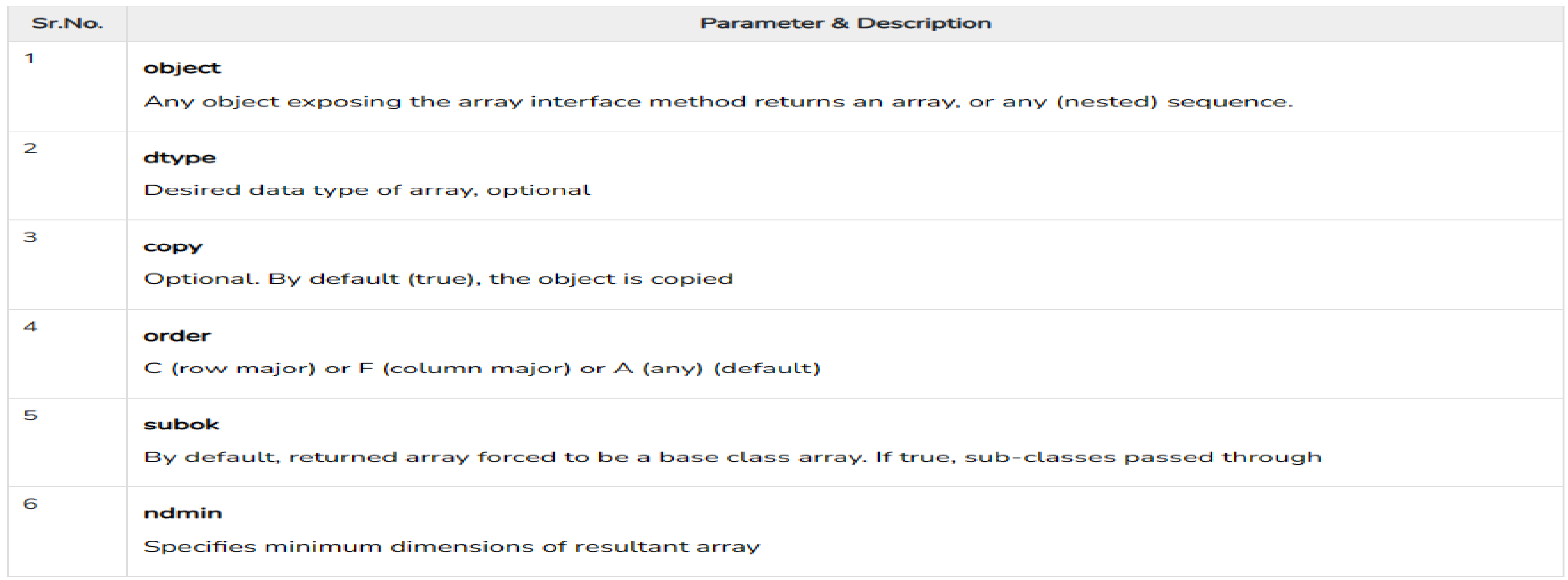
3

Optimizing Numpy Performance

With careful implementation, numpy can offer blazing-fast performance, particularly

NumPy (Ndarray object)

- Ndarray is a collection of same data type elements
- Every element in a ndarray takes the same size of block in the memory
- Elements of array accessed using zero-based index
- Syntax: numpy.array(object*, dtype = None, copy = True, order = None, subok = False, ndmin = 0)



Sr.No.	Parameter & Description
1	**object** Any object exposing the array interface method returns an array, or any (nested) sequence.
2	**dtype** Desired data type of array, optional
3	**copy** Optional. By default (true), the object is copied
4	**order** C (row major) or F (column major) or A (any) (default)
5	**subok** By default, returned array forced to be a base class array. If true, sub-classes passed through
6	**ndmin** Specifies minimum dimensions of resultant array

More on Numpy Array operations

- numpy.sort() ==> Sorts elements
- numpy.concatenate() ==> Joins 2 arrays
- numpy.reshape(number of rows, number of columns) ==> Reshape the array
- ndarray.ndim ==> will tell you, number of axes/dimension of the array
- ndarray.size==> will tell you the total number of elements of the array
- ndarray.shape ==> will display a tuple of integers that indicate the number of elements stored along each dimension of the array. If, for example, you have a 2-D array with 2 rows and 3 columns, the shape of your array is (2, 3)

Creation of numpy array using existing data

Syntax: ***numpy.asarray(data, dtype=None, order=None)***

data = Input data in the form of list, list of tuples, tuples,
Dtype = Default data type of input data is applied
Order = C(row major and its default) or F(column major)

Example 1:

```
# convert list to ndarray
import numpy as np
x = [1,2,3]
a = np.asarray(x)
print a  #output [1 2 3]
```

Example 2:

```
# ndarray from tuple
import numpy as np
x = (1,2,3)
a = np.asarray(x)
print a # output [1 2 3]
```

What is difference between python list and numpy array?

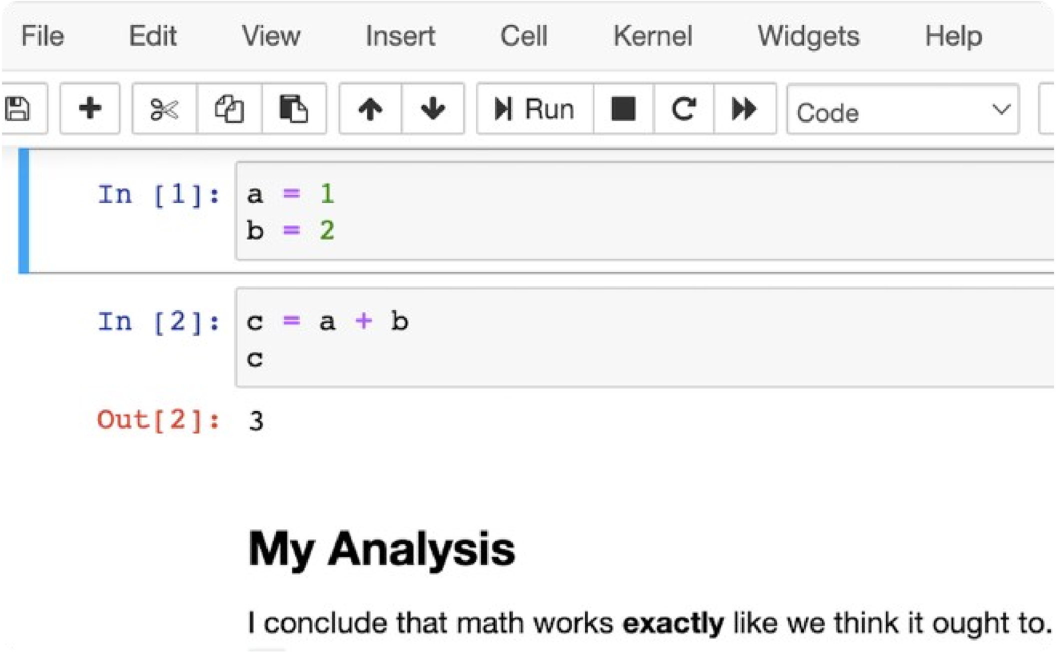
Numpy Array (https://numpy.org/doc/stable/reference/generated/numpy.array.html)

- NumPy array gives you an enormous range of fast and efficient ways of creating arrays and manipulating numerical data inside them.
- All the elements of numpy array are **homogeneous**.
  Example: [1,2,3,4,5]
- NumPy arrays are faster and more compact than Python lists
- An array consumes less memory and is convenient to use. NumPy uses much less memory to store data and it provides a mechanism of specifying the data types. This allows the code to be optimized even further.

Python List

- While a Python list can contain different data types within a single list.
- All the elements of python list are **heterogeneous**.
  Example: [1, 2.8, 3, ’hello’]

Tips and Tricks for Using Numpy Effectively



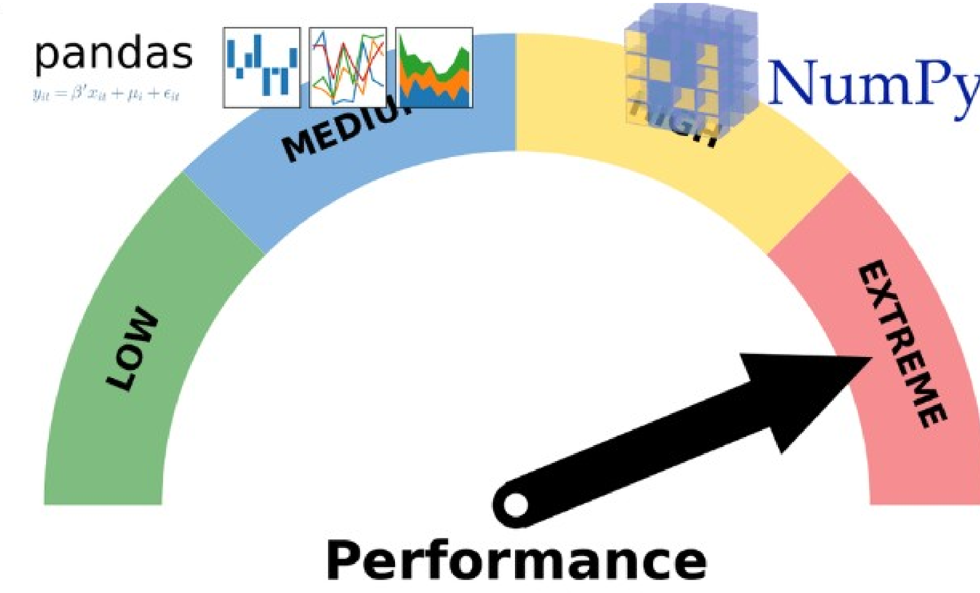
```
In [1]: a = 1
        b = 2

In [2]: c = a + b
        c

Out[2]: 3
```

My Analysis

I conclude that math works **exactly** like we think it ought to.



Avoiding Common Pitfalls and Errors

One of the most significant pitfalls is improper array size and shape, leading to errors. One workaround

Optimizing Numpy Performance

Numpy provides many built-in functions and optimized algorithms to improve performance

Useful links

- https://numpy.org/devdocs/user/absolute_beginners.html#adding-removing-and-sorting-elements
- https://numpy.org/devdocs/user/absolute_beginners.html#how-do-you-know-the-shape-and-size-of-an-array
- https://numpy.org/devdocs/user/absolute_beginners.html#indexing-and-slicing
- https://numpy.org/devdocs/user/absolute_beginners.html#how-to-create-an-array-from-existing-data
- https://numpy.org/devdocs/user/absolute_beginners.html#broadcasting
- https://numpy.org/devdocs/user/absolute_beginners.html#basic-array-operations
- https://numpy.org/devdocs/user/absolute_beginners.html#more-useful-array-operations
- https://numpy.org/devdocs/user/absolute_beginners.html#creating-matrices
- https://numpy.org/devdocs/user/absolute_beginners.html#transposing-and-reshaping-a-matrix
- https://numpy.org/devdocs/user/absolute_beginners.html#how-to-reverse-an-array
- https://numpy.org/devdocs/user/absolute_beginners.html#working-with-mathematical-formulas

Pandas

(https://pandas.pydata.org/docs/getting_started/intro_tutorials/01_table_oriented.html)

What are Pandas?

- Pandas is an open-source library in Python used for data manipulation and analysis. It provides data structures and functions to efficiently manipulate numerical tables and time series data.
- Pandas is an open-source Python Library providing high-performance data manipulation ar analysis tool using its powerful data structures
- The name Pandas is derived from the word Panel Data – an Econometrics from **Multidimensional** data.
- Useful in data preparation
- Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

Data Structures

Pandas offers powerful data structures like DataFrames and Series to handle structured data effectively.

Data Manipulation

Pandas supports various operations like filtering, merging, grouping, reshaping, and more to manipulate data quickly.

Missing Data Handling

Pandas provides flexible tools to handle missing or incomplete data, preventing disruptions in your analysis.

Time Series Analysis

Pandas includes functionalities to work with time series data, making it easier to analyze trends and patterns over time.

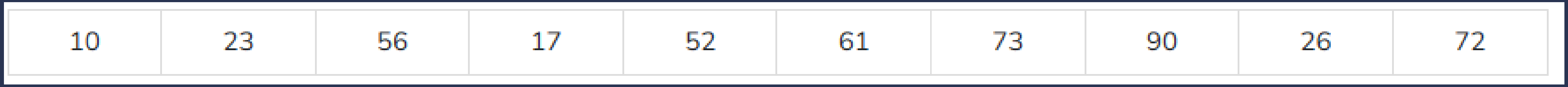
Data structures of Pandas

1. Series ==> 1D labeled homogeneous array, size immutable.

2. DataFrame ==> 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns

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

- Key Points

  - Homogeneous data

  - Size Immutable

  - Values of Data Mutable

- Syntax

  pd.Series(data, index=index)

  data ==> Python dict/ndarray/scalar value like 6

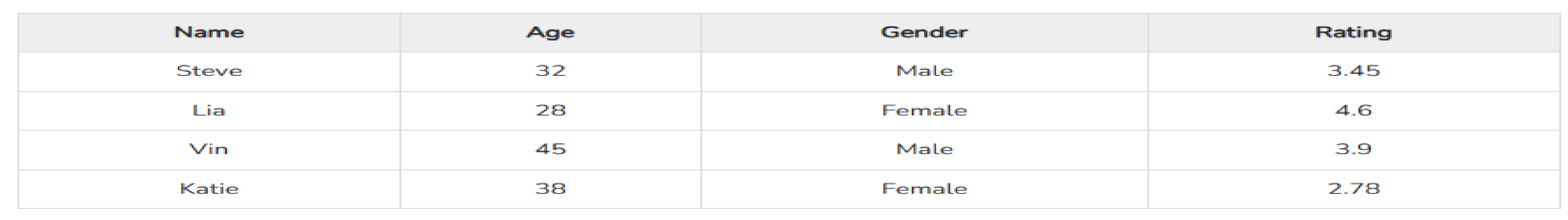
  index ==> list of axis labels depending on what data is (https://pandas.pydata.org/docs/user_guide/dsintro.html#series)

- Example

```
import pandas as pd
s = pd.Series([1,2,3,4])
print(s)
```

Data Frame

- DataFrame is a two-dimensional array with heterogeneous data



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

- **Key Points**

- Heterogeneous data

- Size Mutable

- Data Mutable

- **Example**

```
import pandas as pd
d=pd.DataFrame({ 'A': 1.0,
   'name': ['vaishali'],
   'class': 12,
})
print(d)
```

Dataframe Iteration

To iterate over the rows of the DataFrame, we can use the following functions –

- **items()** – to iterate over the (key,value)/columns pairs
- **iterrows()** – iterate over the rows as (index,series) pairs

Data cleaning and preprocessing with Pandas

Pandas provides you with several fast, flexible, and intuitive ways to clean and prepare your data. By the end of this tutorial, you'll have learned all you need to know to get started with:

- Many data scientists estimate that they spend 80% of their time cleaning and preparing their datasets
- Working with missing data using methods such as .fillna()
- Working with duplicate data using methods such as the .drop_duplicates() method
- Cleaning string data using the .str accessor.

Matplotlib (https://matplotlib.org/stable/tutorials/introductory/quick_start.html)

Introduction to Matplotlib

What is Matplotlib?

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Why use Matplotlib?

Matplotlib provides a wide range of plotting options, making it suitable for various applications, from simple charts to complex data exploration.

Key features of Matplotlib

Matplotlib offers a rich set of features, including support for different plot types, customizable styling options, and interactive capabilities.

Matplotlib plot types (https://matplotlib.org/stable/plot_types/index.html)

1. Bar - Make a bar plot.
2. Barh - Make a horizontal bar plot.
3. Boxplot - Make a box and whisker plot.
4. Hist - Plot a histogram.
5. Hist2d - Make a 2D histogram plot.
6. Pie - Plot a pie chart.
7. Plot - Plot lines and/or markers to the Axes.
8. Polar - Make a polar plot.
9. Scatter - Make a scatter plot of x vs y.
10. Stackplot - Draws a stacked area plot.
11. Stem - Create a stem plot.
12. Step - Make a step plot.
13. Quiver - Plot a 2-D field of arrows.
14. Violinplot

Matplotlib Data Visualization

Data visualization with Matplotlib (check matplotlib.ipynb)

SciPy

(https://docs.scipy.org/doc/scipy/tutorial/index.html)

SciPy

- SciPy is a scientific computation library that uses NumPy underneath.
- SciPy stands for Scientific Python.
- It provides more utility functions for optimization, statistics and signal processing.
- Like NumPy, SciPy is open source so we can use it freely.
- SciPy was created by NumPy's creator Travis Olliphant.

Why Use SciPy?

If SciPy uses NumPy underneath, why can we not just use NumPy?

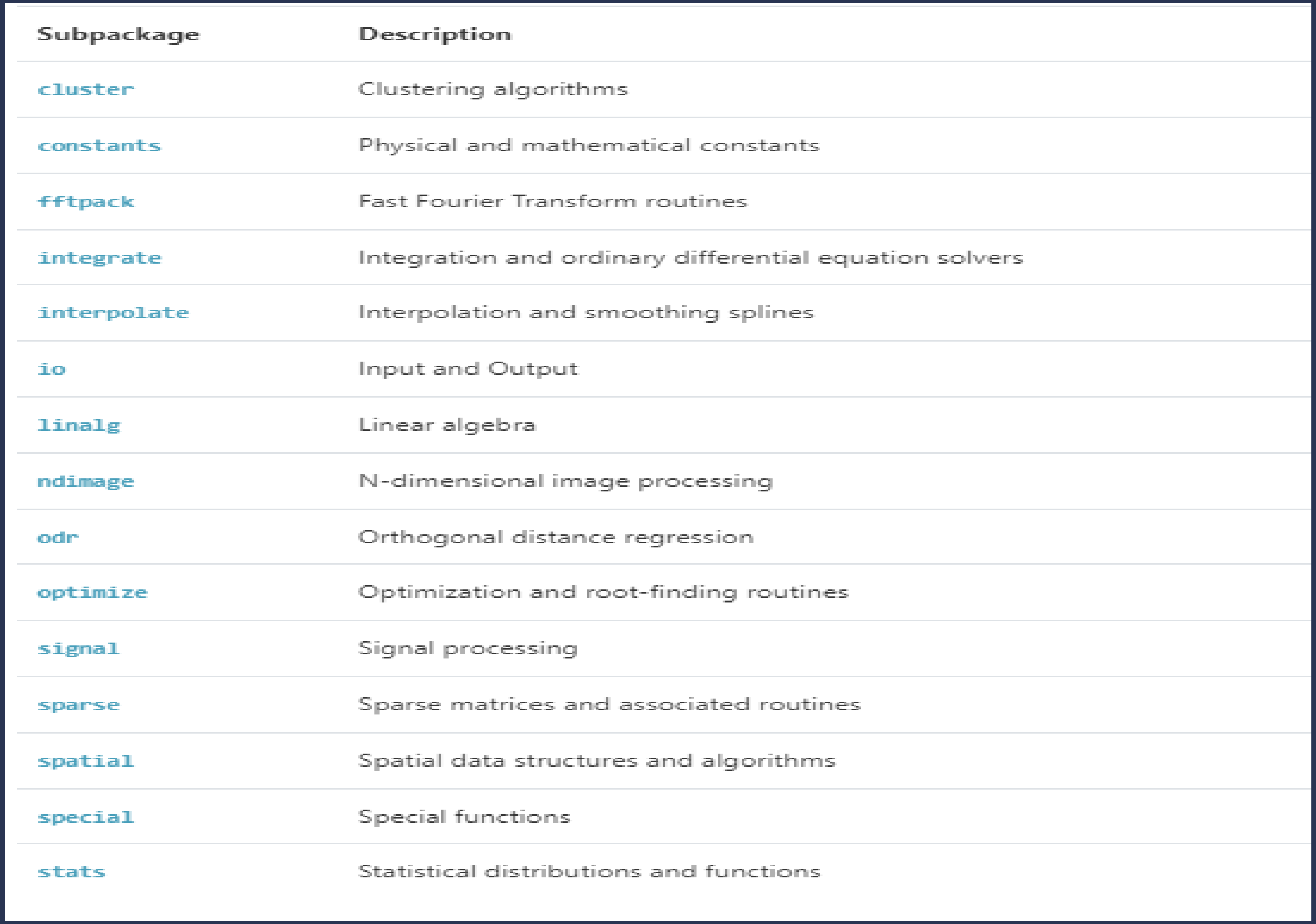
SciPy has optimized and added functions that are frequently used in NumPy and Data Science.

Which Language is SciPy Written in?

SciPy is predominantly written in Python, but a few segments are written in C.

SciPy Subpackages

https://docs.scipy.org/doc/scipy/tutorial/index.html#subpackages



Subpackage	Description
cluster	Clustering algorithms
constants	Physical and mathematical constants
fftpack	Fast Fourier Transform routines
integrate	Integration and ordinary differential equation solvers
interpolate	Interpolation and smoothing splines
io	Input and Output
linalg	Linear algebra
ndimage	N-dimensional image processing
odr	Orthogonal distance regression
optimize	Optimization and root-finding routines
signal	Signal processing
sparse	Sparse matrices and associated routines
spatial	Spatial data structures and algorithms
special	Special functions
stats	Statistical distributions and functions

Scikit-learn (https://scikit-learn.org/stable/tutorial/index.html)

Introduction to scikit-learn

What is scikit-learn?

- Discover how scikit-learn, a powerful Python library, makes machine learning accessible to everyone. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib

Why use scikit-learn?

Explore the advantages and benefits of utilizing scikit-learn for your machine learning projects.

Getting started with scikit-learn

Learn how to install and set up scikit-learn on your machine to start building machine learning models.

Key concepts in scikit-learn

Gain an understanding of the fundamental concepts and terminology used in scikit-learn.

Scikit-Learn Features

- Supervised Learning algorithms – Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.
- Unsupervised Learning algorithms – On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
- Clustering – This model is used for grouping unlabeled data.
- Cross Validation – It is used to check the accuracy of supervised models on unseen data.
- Dimensionality Reduction – It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.
- Ensemble methods – As name suggest, it is used for combining the predictions of multiple supervised models.
- Feature extraction – It is used to extract the features from data to define the attributes in image and text data.
- Feature selection – It is used to identify useful attributes to create supervised models.
- Open Source – It is open source library and also commercially usable under BSD license.