14 December 2018
20:07

Python Modules

Suppose, we created a function as below

```
def check():
        a=int(input('Enter a number'))
        if a%2==0: print("Even")
        else: print("Odd")
```

A module is a Python file containing Python statements and definitions. For example, a file evenodd.py is a module, and we call it ‘evenodd’. We put similar code together in one module. This helps us modularize our code, and make it much easier to deal with. And not only that, a module grants us reusability. With a module, we don’t need to write the same code again for a new project that we take up.

So, as you can see, a module simply contains Python code. Consequently, we can import it, like a package.
>>> import one.two.evenodd

To call function check(), we do the following:
>>> from one.two.evenodd import check
>>> check()

Another example would be the constants ‘pi’ and ‘e’ from the ‘math’ module.
>>> import math
>>> from math import pi
>>> math.pi

Let’s update evenodd.py to have two functions- check and evenodd.

```
def check():
        a=int(input('Enter a number'))
        if a%2==0: print("Even")
        else: print("Odd")
def add(a,b):
        return a+b
```

Now, if we want to import all functions from module evenodd, we can just use the wildcard *:
>>> from one.two.evenodd import *

Python Libraries

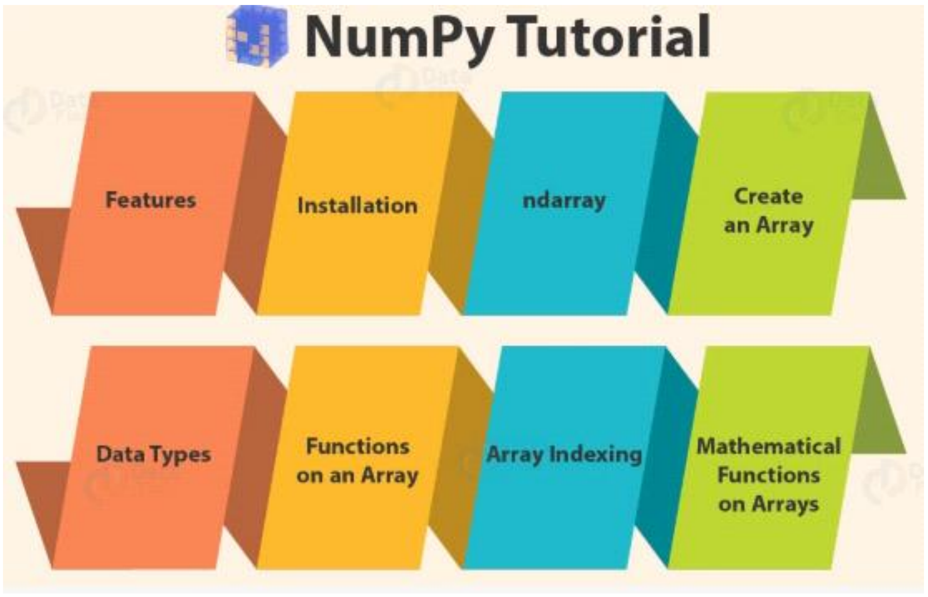
Lets take one step ahead in our journey to learn Python by getting acquainted with some useful libraries. The first step is obviously to learn to import them into our environment. There are several ways of doing so in Python:

```
import math as m
```

```
from math import *
```

Following are a list of libraries, you will need for any scientific computations and data analysis:

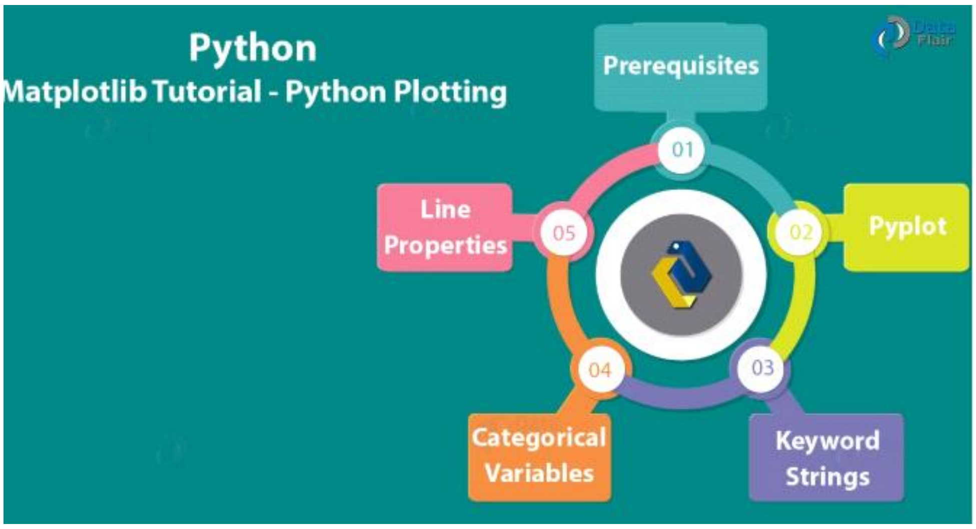
- **NumPy** stands for Numerical Python. The most powerful feature of NumPy is n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools for integration with other low level languages like Fortran, C and C++



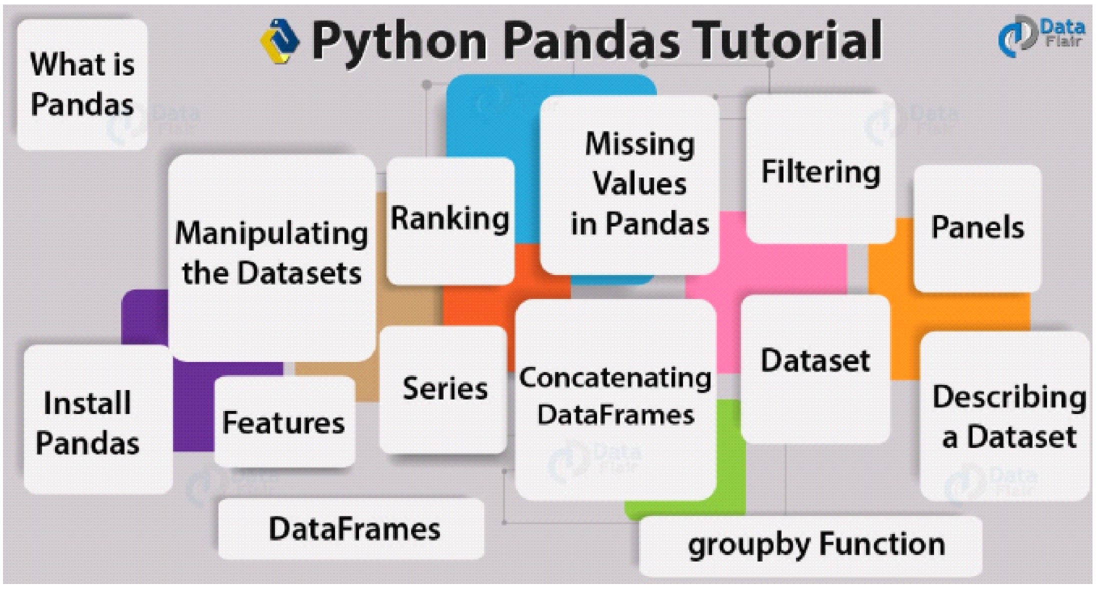
- **SciPy** stands for Scientific Python. SciPy is built on NumPy. It is one of the most useful library for variety of high level science and engineering modules like discrete Fourier transform, Linear Algebra, Optimization and Sparse matrices.

SciPy Tutorial
Introducing SciPy
Optimizing
Installation
SciPy Subpackages
Working with Statistics
Benefits of SciPy
Processing Images with SciPy
Working with Polynomials
Linear Algebra
Integration
Fast Fourier Transforms
Processing Signals with SciPy
Vectorizin Functions
Special Functions

- **Matplotlib** is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source

- 

- **Pandas** for structured data operations and manipulations. It is extensively used for data munging and preparation. Pandas were added relatively recently to Python and have been instrumental in boosting Python's usage in data scientist community. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, organize, manipulate, model, and analyse the data.

- 

- **Scikit Learn** for machine learning. Built on NumPy, SciPy and matplotlib, this library contains a lot of effiecient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

- **Statsmodels** for statistical modeling. Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.

- **Seaborn** for statistical data visualization. Seaborn is a library for making attractive and informative statistical graphics in Python. It is based on matplotlib. Seaborn aims to make visualization a central part of exploring and understanding data.

- **Bokeh** for creating interactive plots, dashboards and data applications on modern web-browsers. It empowers the user to generate elegant and concise graphics in the style of D3.js. Moreover, it has the capability of high-performance interactivity over very large or streaming datasets.

- **Blaze** for extending the capability of Numpy and Pandas to distributed and streaming datasets. It can be used to access data from a multitude of sources including Bcolz, MongoDB, SQLAlchemy, Apache Spark, PyTables, etc. Together with Bokeh, Blaze can act as a very powerful tool for creating effective visualizations and dashboards on huge chunks of data.

- **Scrapy** for web crawling. It is a very useful framework for getting specific patterns of data. It has the capability to start at a website home url and then dig through web-pages within the website to gather information.

- **SymPy** for symbolic computation. It has wide-ranging capabilities from basic symbolic arithmetic to calculus, algebra, discrete mathematics and quantum physics. Another useful feature is the capability of formatting the result of the computations as LaTeX code.

- **Requests** for accessing the web. It works similar to the the standard python library urllib2 but is much easier to code. You will find subtle differences with urllib2 but for beginners, Requests might be more convenient.