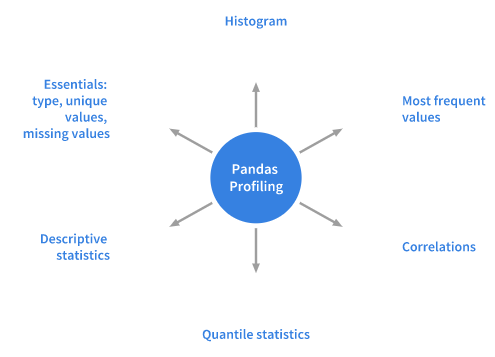
**Exploratory Data Analysis**

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

**1. Profiling the pandas dataframe**

Profiling is a process that helps us in understanding our data and [Pandas Profiling](https://github.com/pandas-profiling/pandas-profiling) is python package which does exactly that. It is a simple and fast way to perform exploratory data analysis of a Pandas Dataframe. The pandas df.describe() and df.info()functions are normally used as a first step in the EDA process. However, it only gives a very basic overview of the data and doesn’t help much in the case of large data sets. The Pandas Profiling function, on the other hand, extends the pandas DataFrame with df.profile\_report() for quick data analysis. It displays a lot of information with a single line of code and that too in an interactive HTML report.

For a given dataset the pandas profiling package computes the following statistics:



Statistics computer by Pandas Profiling package.

**Installation**

pip install pandas-profiling  
or  
conda install -c anaconda pandas-profiling

**2.What is Tableau?**

Tableau is a powerful and fastest growing data visualization tool used in the Business Intelligence Industry. It helps in simplifying raw data into the very easily understandable format.

Data analysis is very fast with Tableau and the visualizations created are in the form of dashboards and worksheets. The data that is created using Tableau can be understood by professional at any level in an organization. It even allows a non-technical user to create a customized dashboard.

The best features Tableau are

Data Blending

Real time analysis

Collaboration of data

The great thing about Tableau software is that it doesn't require any technical or any kind of programming skills to operate. The tool has garnered interest among the people from all sectors such as business, researchers, different industries, etc.

**Fast Text Extraction with Python and Tika**

Recently I have been doing a lot of work using text data in my machine learning models and have found extracting text from documents an incredibly slow and frustrating process.

Having tried a range of libraries I finally came across an [Apache Tika](https://tika.apache.org/) port for Python which extracts text quickly, accurately and is simpler to use than the other libraries I have come across.

Tika works on .pdf, the most recent OOXML Microsoft Office file types and older binary file formats such as .doc, .ppt and .xls. For a full list of supported file types [see here](https://tika.apache.org/1.4/formats.html).

**Requirements**

First we need to install the [tika-python](https://github.com/chrismattmann/tika-python) library written by Chris Mattmann, this can be done via pip in the command line:

pip install tika

To allow the library to launch the Tika REST server in the background [Java 7](https://java.com/en/download/) or higher also needs to be installed.

**The Code**

Below is the code to extract the contents of a file, note how simple it is with the library handling all communication with the REST server and returning a dictionary containing the parsed data.

from tika import parser

file = 'path/to/file'

# Parse data from file

file\_data =parser.from\_file(file)

# Get files text content

text = file\_data['content']

print(text)

The Tika parser can also be combined with Pythons multiprocessing module, in the below example the code fetches some file paths from a MySQL database, then parses the files in parallel and writes the extracted text back to the database.

**Introduction to Dash**

Dash is a productive Python framework for building web applications.

Written on top of Flask, Plotly.js, and React.js, Dash is ideal for building data visualization apps with highly custom user interfaces in pure Python. It's particularly suited for anyone who works with data in Python.

Through a couple of simple patterns, Dash abstracts away all of the technologies and protocols that are required to build an interactive web-based application. Dash is simple enough that you can bind a user interface around your Python code in an afternoon.

Dash apps are rendered in the web browser. You can deploy your apps to servers and then share them through URLs. Since Dash apps are viewed in the web browser, Dash is inherently cross-platform and mobile ready.

**What is Matplotlib**

To make necessary statistical inferences, it becomes necessary to visualize your data and Matplotlib is one such solution for the Python users. It is a very powerful plotting library useful for those working with Python and NumPy. The most used module of Matplotib is Pyplot which provides an interface like MATLAB but instead, it uses Python and it is open source.

**Installing Matplotlib**

To install Matplotlib on your local machine, open Python command prompt and type following commands:

python -m pip install -U pip  
python -m pip install -U matplotlib

**General Concepts**

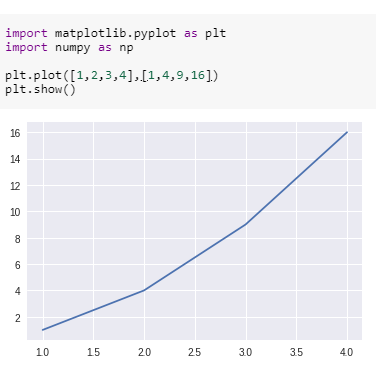
A Matplotlib figure can be categorized into several parts as below:

**Figure:** It is a whole figure which may contain one or more than one axes (plots). You can think of a **Figure** as a canvas which contains plots.

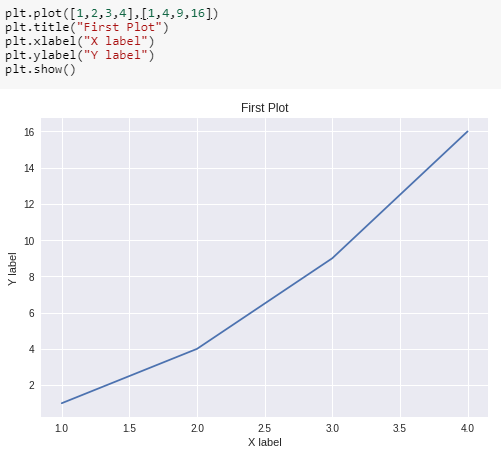
**Axes:** It is what we generally think of as a plot. A **Figure** can contain many Axes.It contains two or three (in the case of 3D) **Axis** objects. Each Axes has a title, an x-label and a y-label.

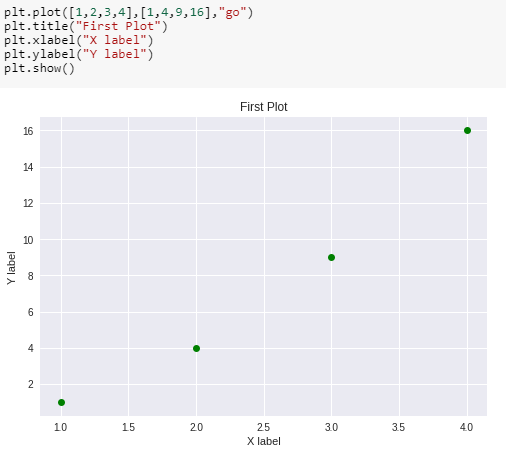
**Axis:** They are the number line like objects and take care of generating the graph limits.

Here we import Matplotlib’s Pyplot module and Numpy library as most of the data that we will be working with will be in the form of arrays only.

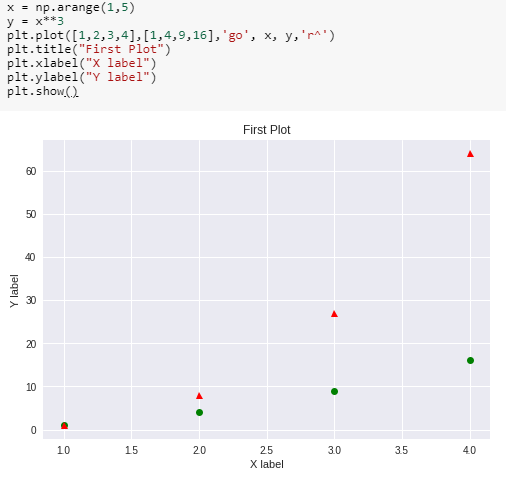


We pass two arrays as our input arguments to Pyplot’s plot() method and use show() method to invoke the required plot. Here note that the first array appears on the x-axis and second array appears on the y-axis of the plot. Now that our first plot is ready, let us add the title, and name x-axis and y-axis using methods title(), xlabel() and ylabel() respectively.



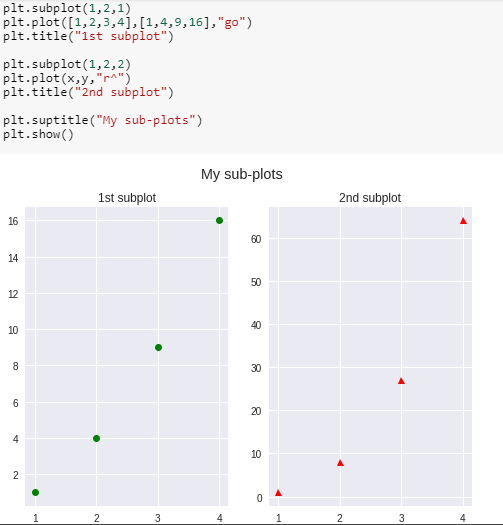


We can also plot multiple sets of data by passing in multiple sets of arguments of X and Y axis in the plot() method as shown.

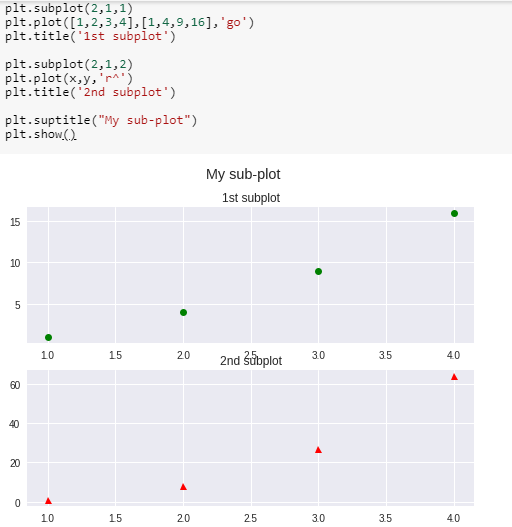


Multiple plots in one figure:

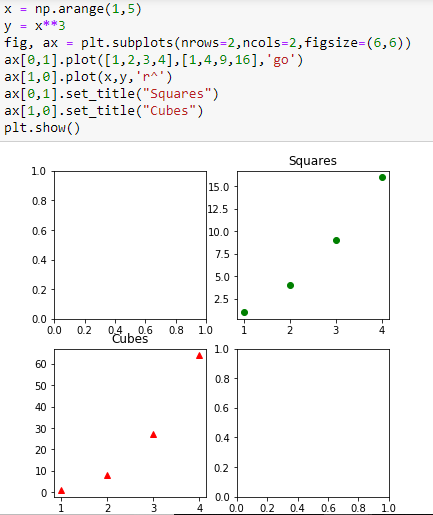
We can use subplot() method to add more than one plots in one figure. In the image below, we used this method to separate two graphs which we plotted on the same axes in the previous example. The subplot() method takes three arguments: they are nrows, ncols and index. They indicate the number of rows, number of columns and the index number of the sub-plot. For instance, in our example, we want to create two sub-plots in one figure such that it comes in one row and in two columns and hence we pass arguments (1,2,1) and (1,2,2) in the subplot() method. Note that we have separately used title()method for both the subplots. We use suptitle() method to make a centralized title for the figure.



If we want our sub-plots in two rows and single column, we can pass arguments (2,1,1) and (2,1,2)



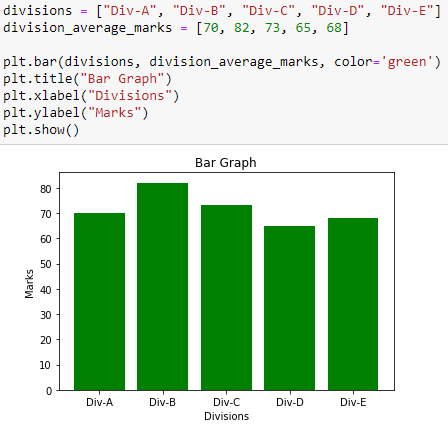
The above way of creating subplots becomes a bit tedious when we want many subplots in our figure. A more convenient way is to use subpltots() method. Notice the difference of **‘s’** in both the methods. This method takes two arguments nrows and ncols as number of rows and number of columns respectively. This method creates two objects:figure and axes which we store in variables fig and ax which can be used to change the figure and axes level attributes respectively. Note that these variable names are chosen arbitrarily.



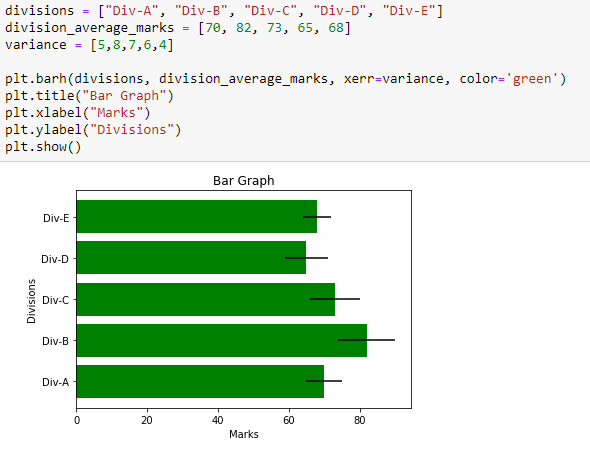
Creating different types of graphs with Pyplot

1) Bar Graphs

Bar graphs are one of the most common types of graphs and are used to show data associated with the categorical variables. Pyplot provides a method bar() to make bar graphs which take arguments: categorical variables, their values and color (if you want to specify any).



To make horizontal bar graphs use method barh() Also we can pass an argument (with its value)xerr oryerr (in case of the above vertical bar graphs) to depict the variance in our data as follows:



To create horizontally stacked bar graphs we use the bar() method twice and pass the arguments where we mention the index and width of our bar graphs in order to horizontally stack them together. Also, notice the use of two other methods legend() which is used to show the legend of the graph and xticks() to label our x-axis based on the position of our bars.

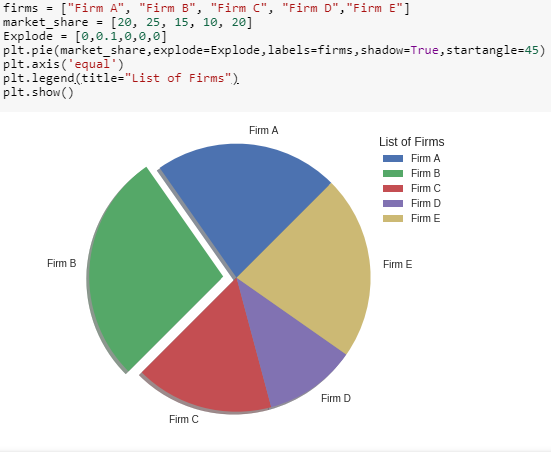


Similarly, to vertically stack the bar graphs together, we can use an argument bottom and mention the bar graph which we want to stack below as its value.



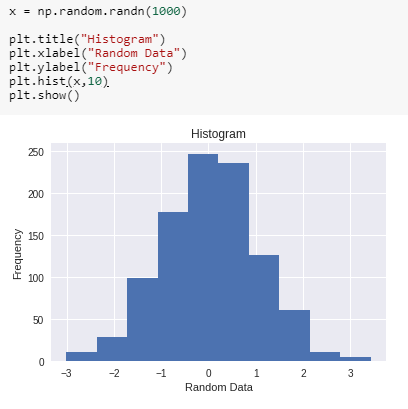
2) Pie Charts

One more basic type of chart is a Pie chart which can be made using the method pie() We can also pass in arguments to customize our Pie chart to show shadow, explode a part of it, tilt it at an angle as follows:



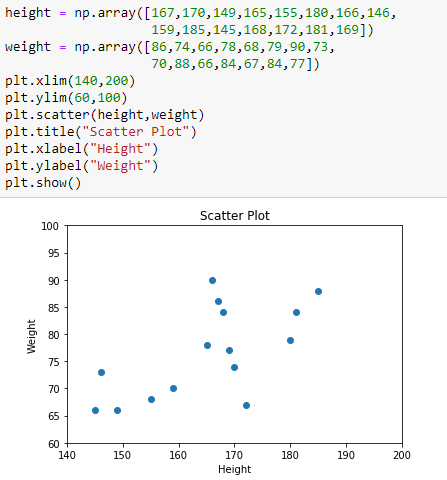
3) Histogram

Histograms are a very common type of plots when we are looking at data like height and weight, stock prices, waiting time for a customer, etc which are continuous in nature. Histogram’s data is plotted within a range against its frequency. Histograms are very commonly occurring graphs in probability and statistics and form the basis for various distributions like the normal -distribution, t-distribution, etc. In the following example, we generate a random continuous data of 1000 entries and plot it against its frequency with the data divided into 10 equal strata. We have used NumPy’s random.randn() method which generates data with the properties of a standard normal distribution i.e. mean = 0 and standard deviation = 1, and hence the histogram looks like a normal distribution curve.



4)Scatter Plots and 3-D plotting

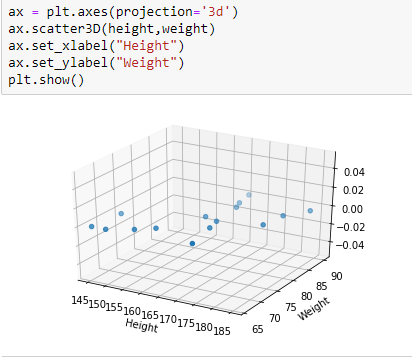
Scatter plots are widely used graphs, especially they come in handy in visualizing a problem of regression. In the following example, we feed in arbitrarily created data of height and weight and plot them against each other. We used xlim() and ylim() methods to set the limits of X-axis and Y-axis respectively.



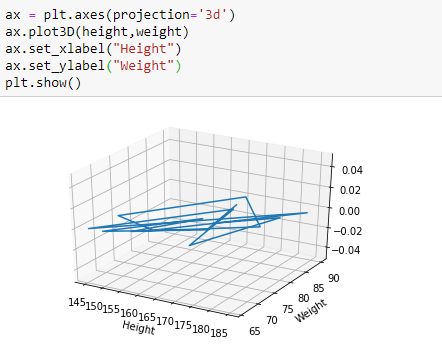
The above scatter can also be visualized in three dimensions. To use this functionality, we first import the module mplot3d as follows:

from mpl\_toolkits import mplot3d

Once the module is imported, a three-dimensional axes is created by passing the keyword projection='3d' to the axes() method of Pyplot module. Once the object instance is created, we pass our arguments height and weight to scatter3D() method.



We can also create 3-D graphs of other types like line graph, surface, wireframes, contours, etc. The above example in the form of a simple line graph is as follows: Here instead of scatter3D() we use method plot3D()



**chatbot**

A chatbot is an artificial intelligence-powered piece of software in a device (Siri, Alexa, Google Assistant etc), application, website or other networks that try to gauge consumer’s needs and then assist them to perform a particular task like a commercial transaction, hotel booking, form submission etc . Today almost every company has a chatbot deployed to engage with the users. Some of the ways in which companies are using chatbots are:

To deliver flight information

to connect customers and their finances

As customer support

The possibilities are (almost) limitless.



<http://bots.duolingo.com/>

How do Chatbots work?

There are broadly two variants of [chatbots](https://medium.com/botsupply/rule-based-bots-vs-ai-bots-b60cdb786ffa): Rule-Based and Self-learning.

In a Rule-based approach, a bot answers questions based on some rules on which it is trained on. The rules defined can be very simple to very complex. The bots can handle simple queries but fail to manage complex ones.

Self-learning bots are the ones that use some Machine Learning-based approaches and are definitely more efficient than rule-based bots. These bots can be of further two types: Retrieval Based or Generative

**i) In retrieval-based models**, a chatbot uses some heuristic to select a response from a library of predefined responses. The chatbot uses the message and context of the conversation for selecting the best response from a predefined list of bot messages. The context can include a current position in the dialogue tree, all previous messages in the conversation, previously saved variables (e.g. username). Heuristics for selecting a response can be engineered in many different ways, from rule-based if-else conditional logic to machine learning classifiers.

**ii) Generative bots** can generate the answers and not always replies with one of the answers from a set of answers. This makes them more intelligent as they take word by word from the query and generates the answers.

