PLOT IN PYTHON

Ritik Bilala @ 2020

RITIK BILALA

Hi, I am mechanical engineering undergraduate from IITB,

I am learning data analysis, machine learning and deep learning.

python visualisation tools are building block for any analysis and Here are some tutorials to learn

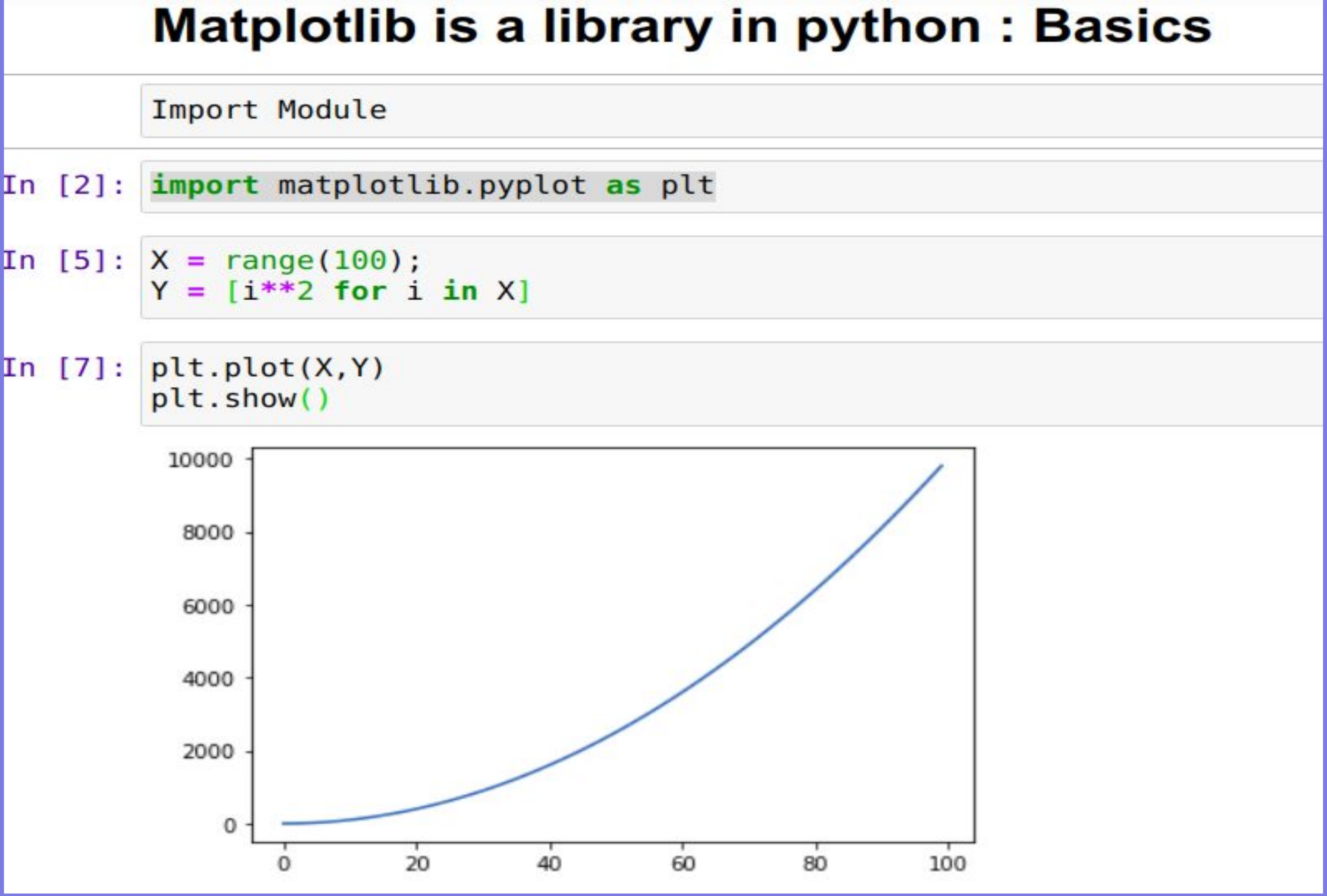
Matplotlib produces publishable quality plots are are directly usable for our projects

I have created this content for beginners

Matplotlib & Seaborn

ML ENGINEER
DATA SCIENTIST
DATA ENGINEERS

Limited Edition - FREE



Matplotlib is a library in python : Basics

Import Module

```
In [2]: import matplotlib.pyplot as plt
```

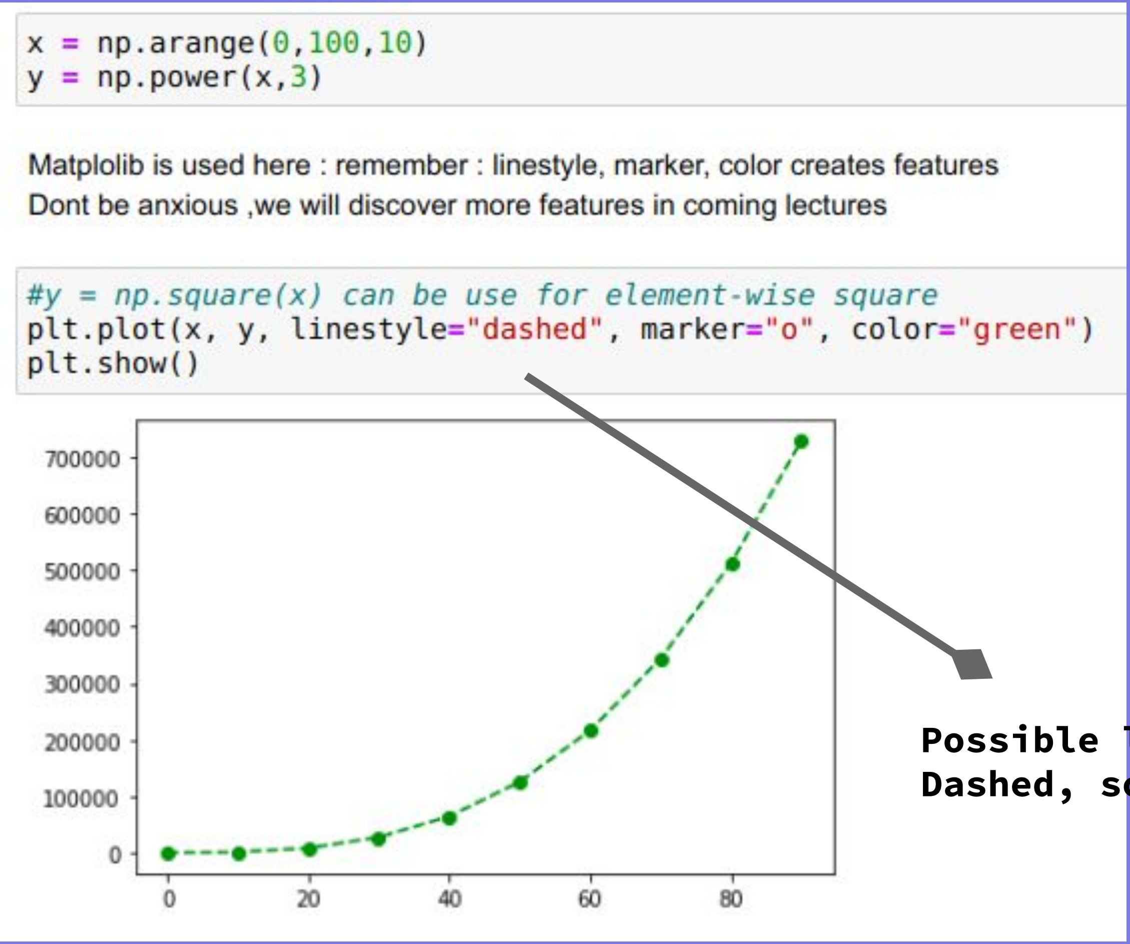
```
In [5]: X = range(100);
        Y = [i**2 for i in X]
```

```
In [7]: plt.plot(X,Y)
        plt.show()
```

Plt is alias and is equivalent to type matplotlib.pyplot
plt.show() is used to show plot which otherwise may or may not displayed

LET'S START THE FUN

LET'S START THE FUN



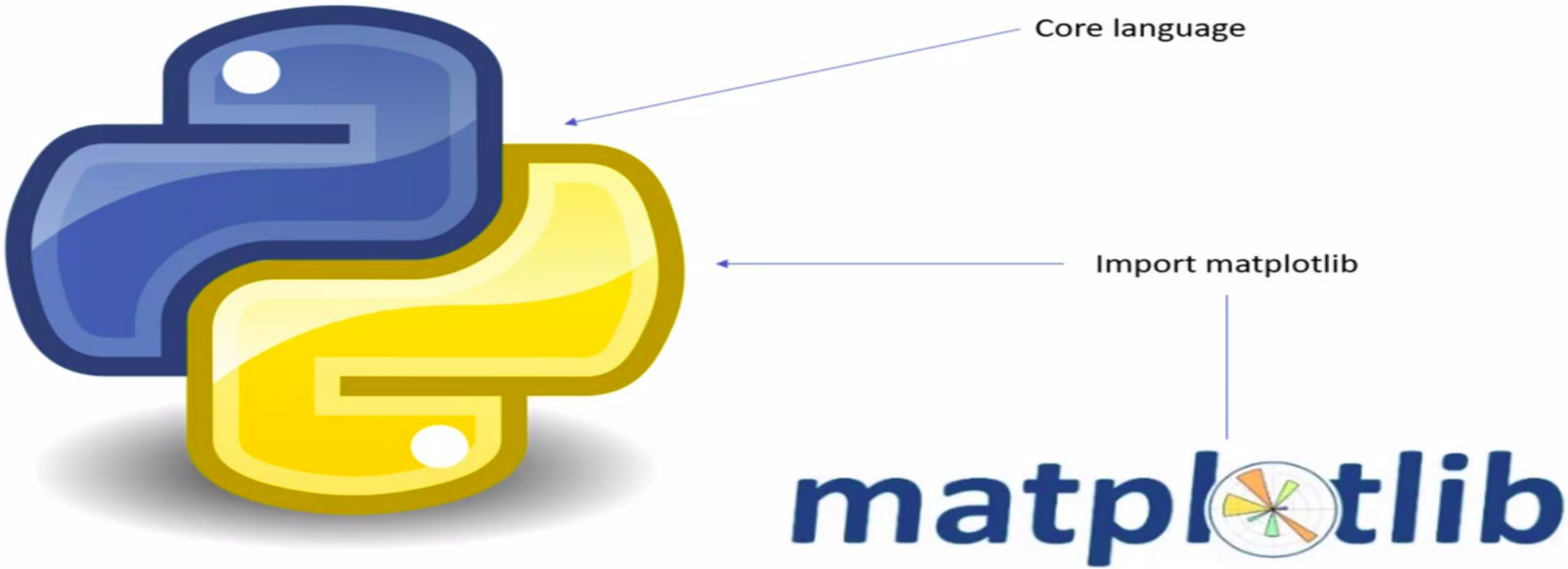
```
x = np.arange(0,100,10)
y = np.power(x,3)
```

Matplolib is used here : remember : linestyle, marker, color creates features

Dont be anxious ,we will discover more features in coming lectures

```
#y = np.square(x) can be use for element-wise square
plt.plot(x, y, linestyle="dashed", marker="o", color="green")
plt.show()
```

Possible linestyles:
Dashed, solid, dotted



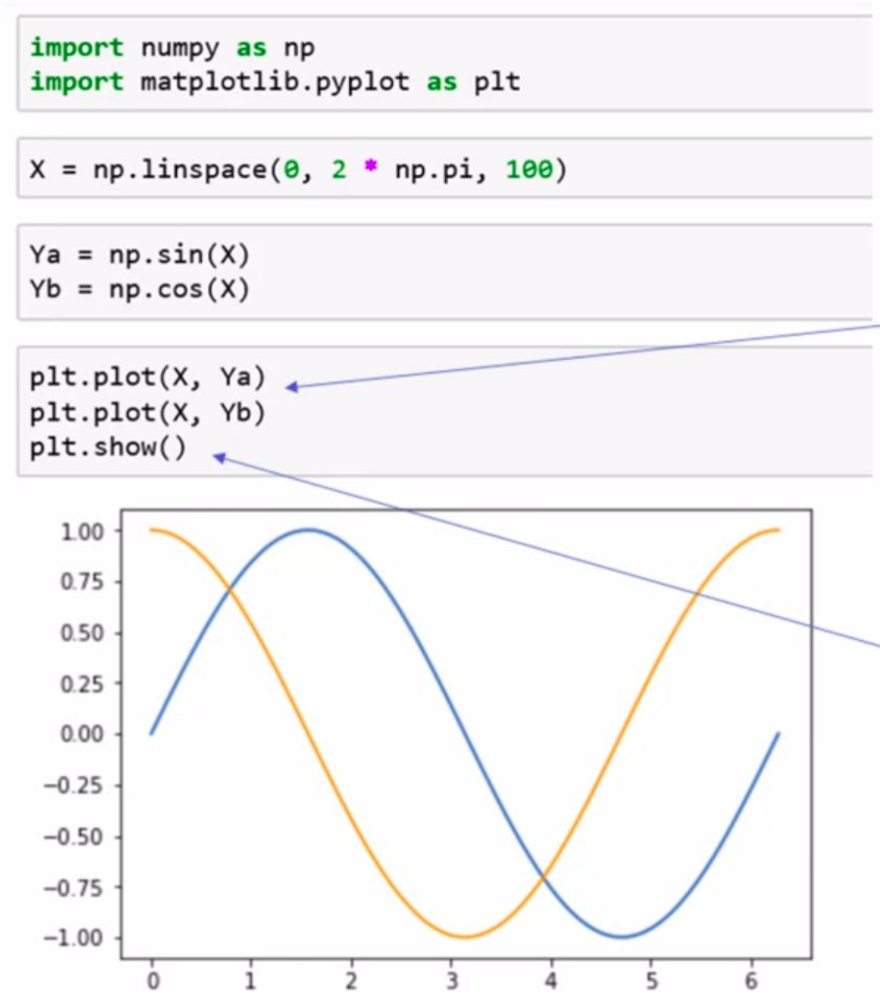
Core language

Import matplotlib

1. Take advantage of existing general purpose programming language python
2. It is orthogonal: it does only plotting

Philosophy of matplotlib

Plotting Multiple Plots



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

```python
X = np.linspace(0, 2 * np.pi, 100)
```

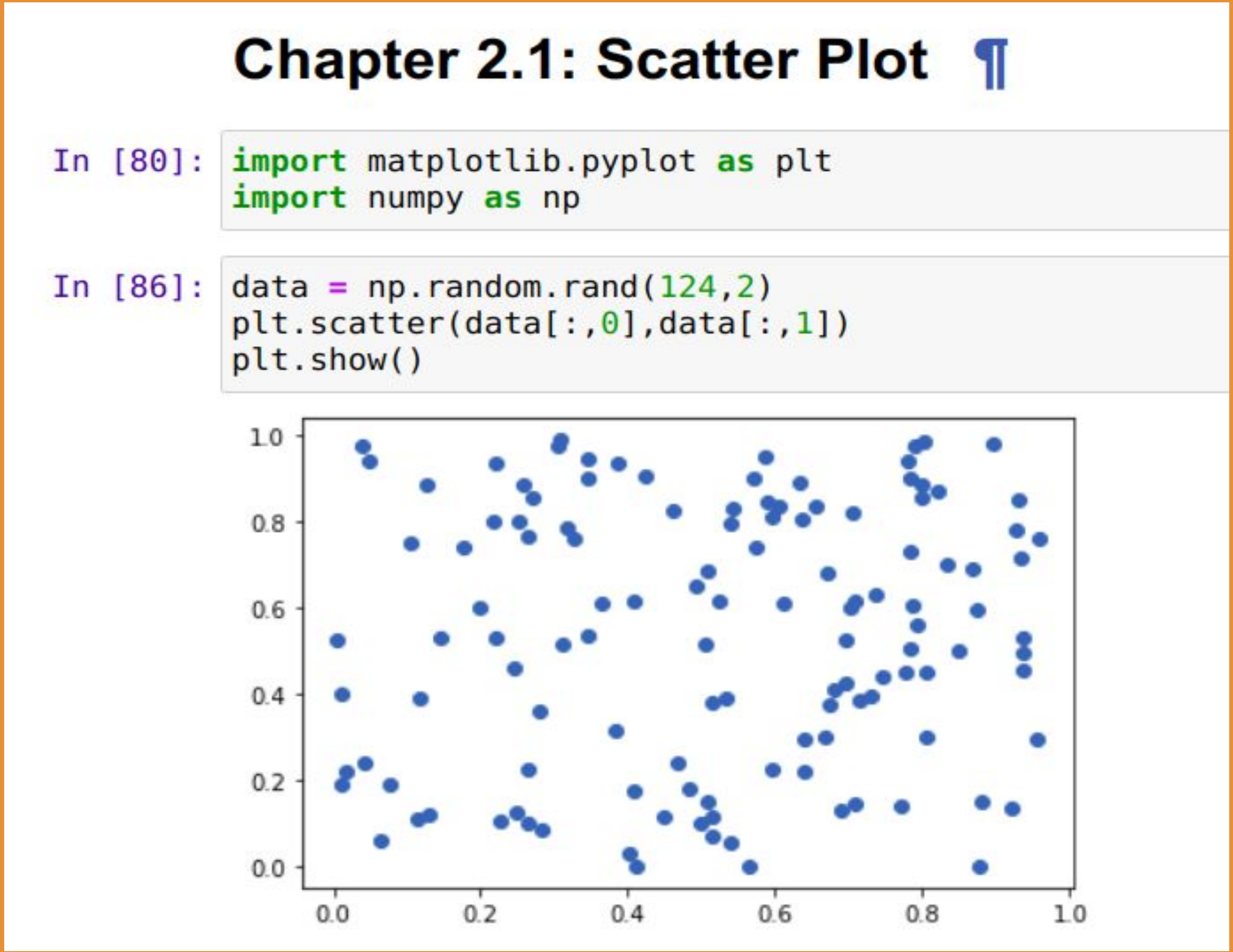
```python
Ya = np.sin(X)
Yb = np.cos(X)
```

```python
plt.plot(X, Ya)
plt.plot(X, Yb)
plt.show()
```

Same function used for creating our curve so we call it twice.

Deferred rendering – The graph is created upon calling plt.show().

SCATTER PLOT

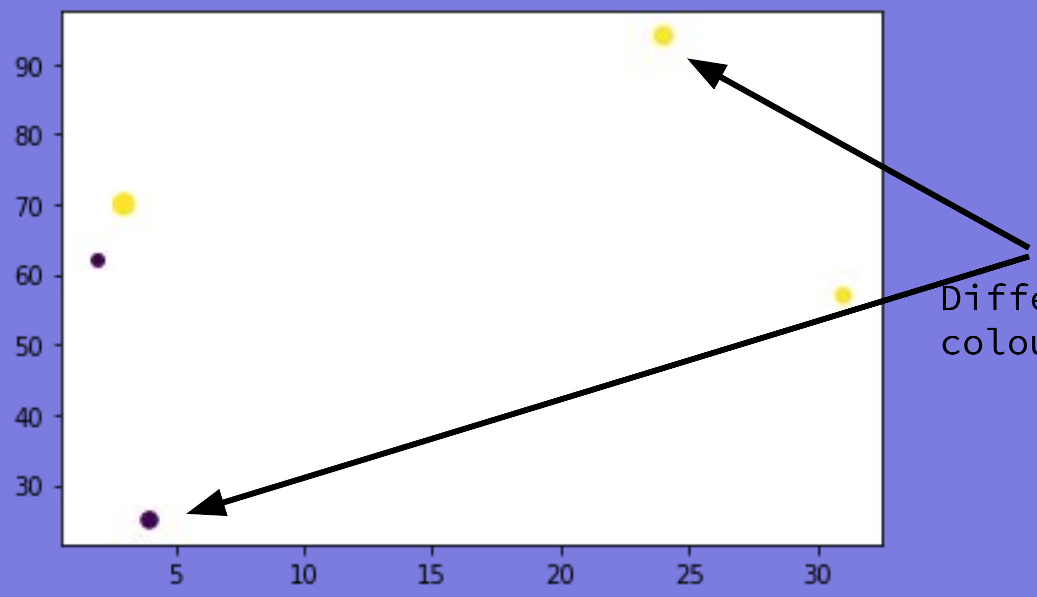


Chapter 2.1: Scatter Plot ¶

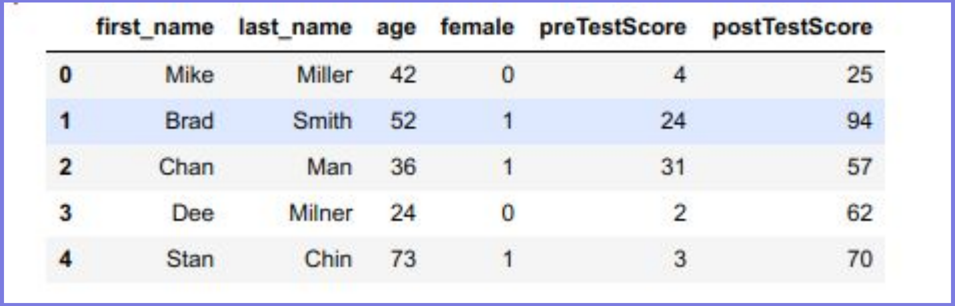
```
In [80]: import matplotlib.pyplot as plt
         import numpy as np
```

```
In [86]: data = np.random.rand(124,2)
         plt.scatter(data[:,0],data[:,1])
         plt.show()
```

Pandas Dataframe



Different colours

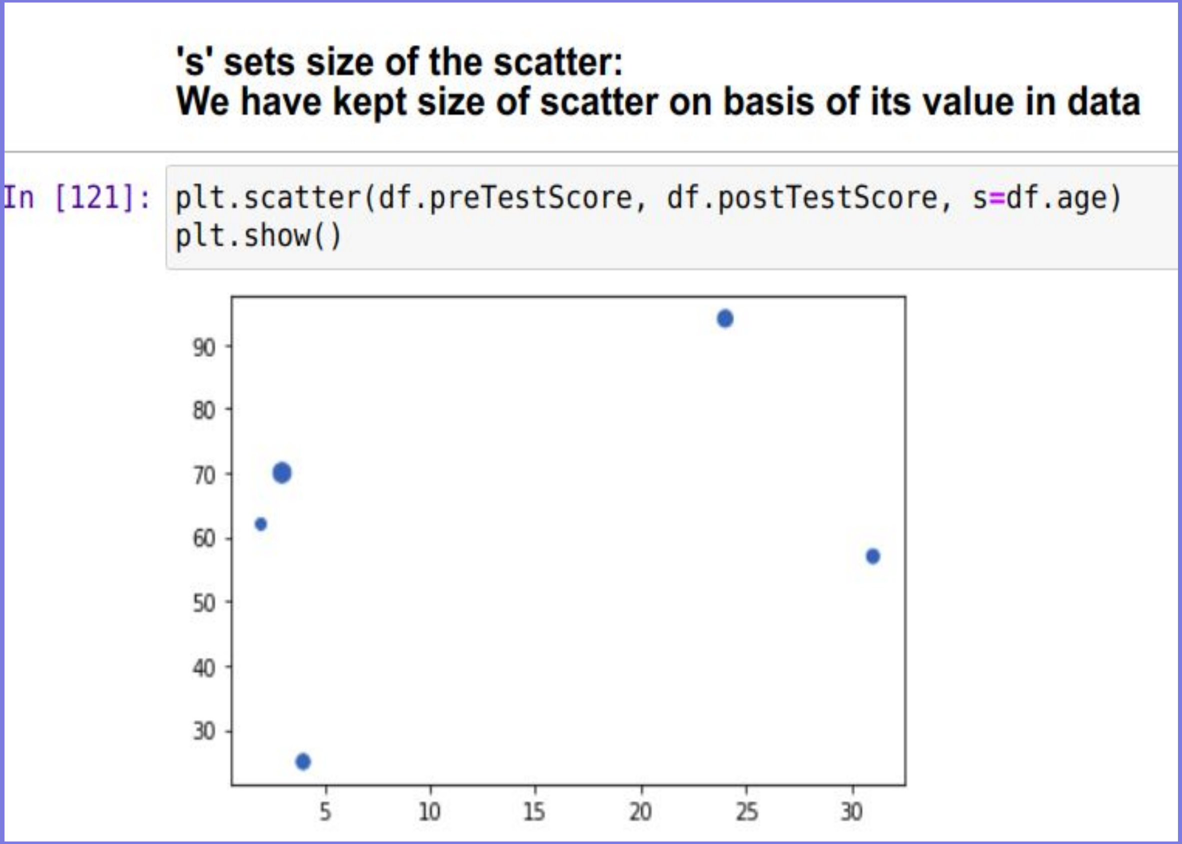


	first_name	last_name	age	female	preTestScore	postTestScore
0	Mike	Miller	42	0	4	25
1	Brad	Smith	52	1	24	94
2	Chan	Man	36	1	31	57
3	Dee	Milner	24	0	2	62
4	Stan	Chin	73	1	3	70

c sets color
And we utilised one of the data entry to customize color of scatter

```
plt.scatter(df.preTestScore,df.postTestScore, s=df.age), c = df.female;
plt.show()                                                              ;
```

Using pandas Dataframe



's' sets size of the scatter:
We have kept size of scatter on basis of its value in data

```
In [121]: plt.scatter(df.preTestScore, df.postTestScore, s=df.age)
          plt.show()
```

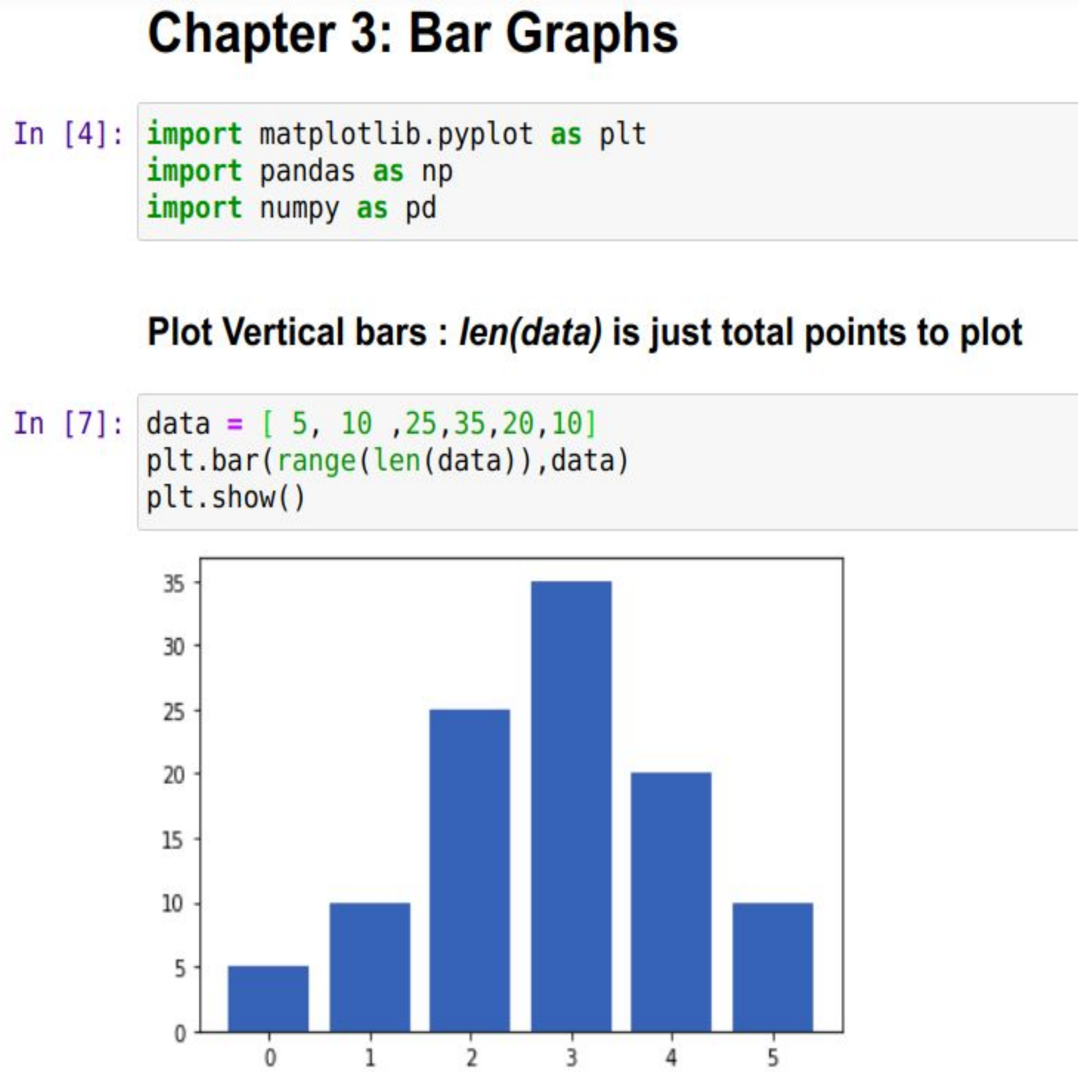
s sets size
And we utilised
one of the data
entry to
customize size
of scatter

```
plt.scatter(df.preTestScore,df.postTestScore, s=df.age);
plt.show()                                              ;
```

```
Horizontal bar plot          :
data = [ 5, 10 ,25,35,20,10]  ;
plt.barh(range(len(data)),data)
plt.show()                    ;
```

BAR PLOT

```
Vertical bar plot           :
data = [ 5, 10 ,25,35,20,10] ;
plt.bar(range(len(data)),data)
plt.show()                   ;
```



Chapter 3: Bar Graphs

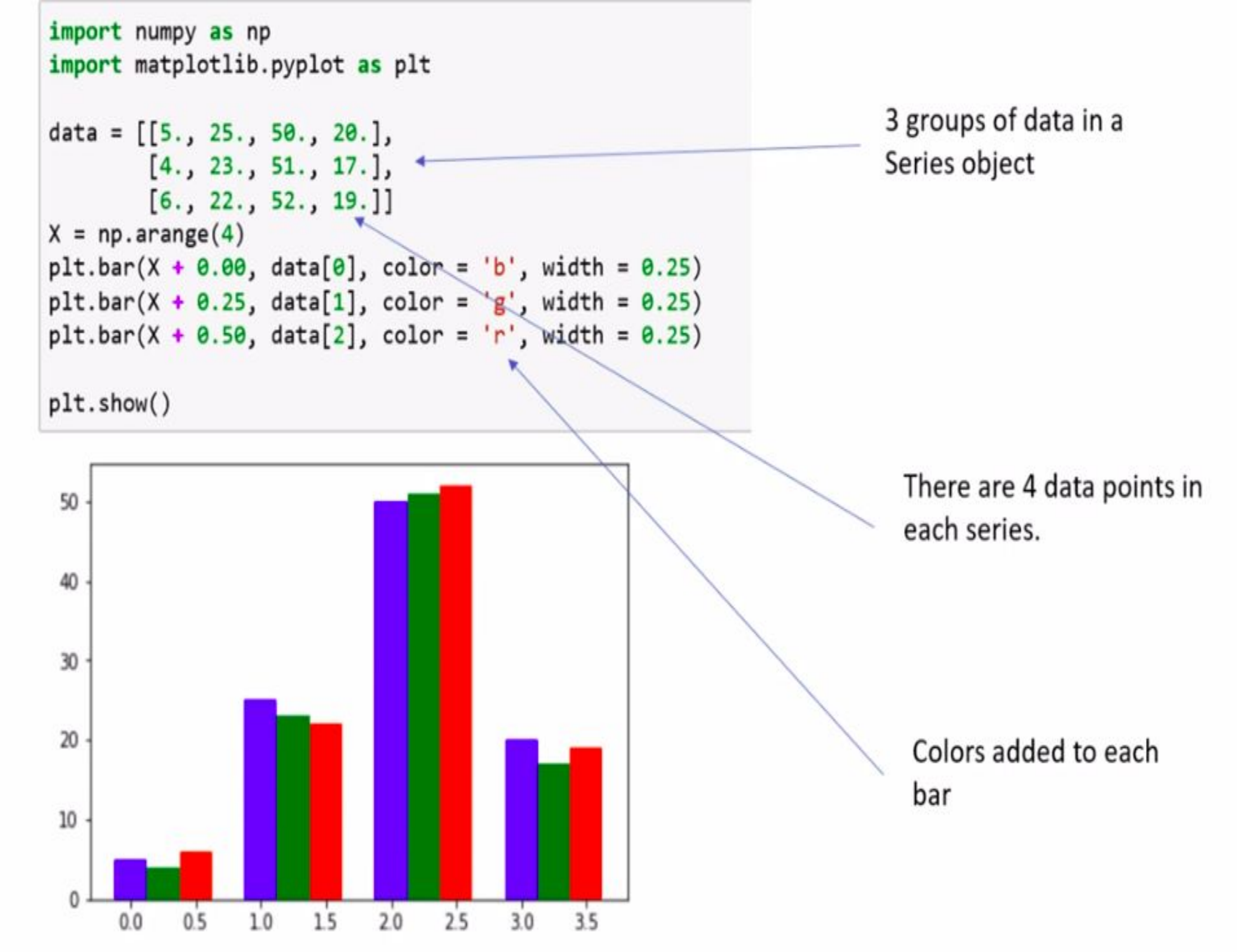
```
In [4]: import matplotlib.pyplot as plt
        import pandas as np
        import numpy as pd
```

Plot Vertical bars : *len(data)* is just total points to plot

```
In [7]: data = [ 5, 10 ,25,35,20,10]
        plt.bar(range(len(data)),data)
        plt.show()
```

Multiple Bar plots

Shift each bar with some offset



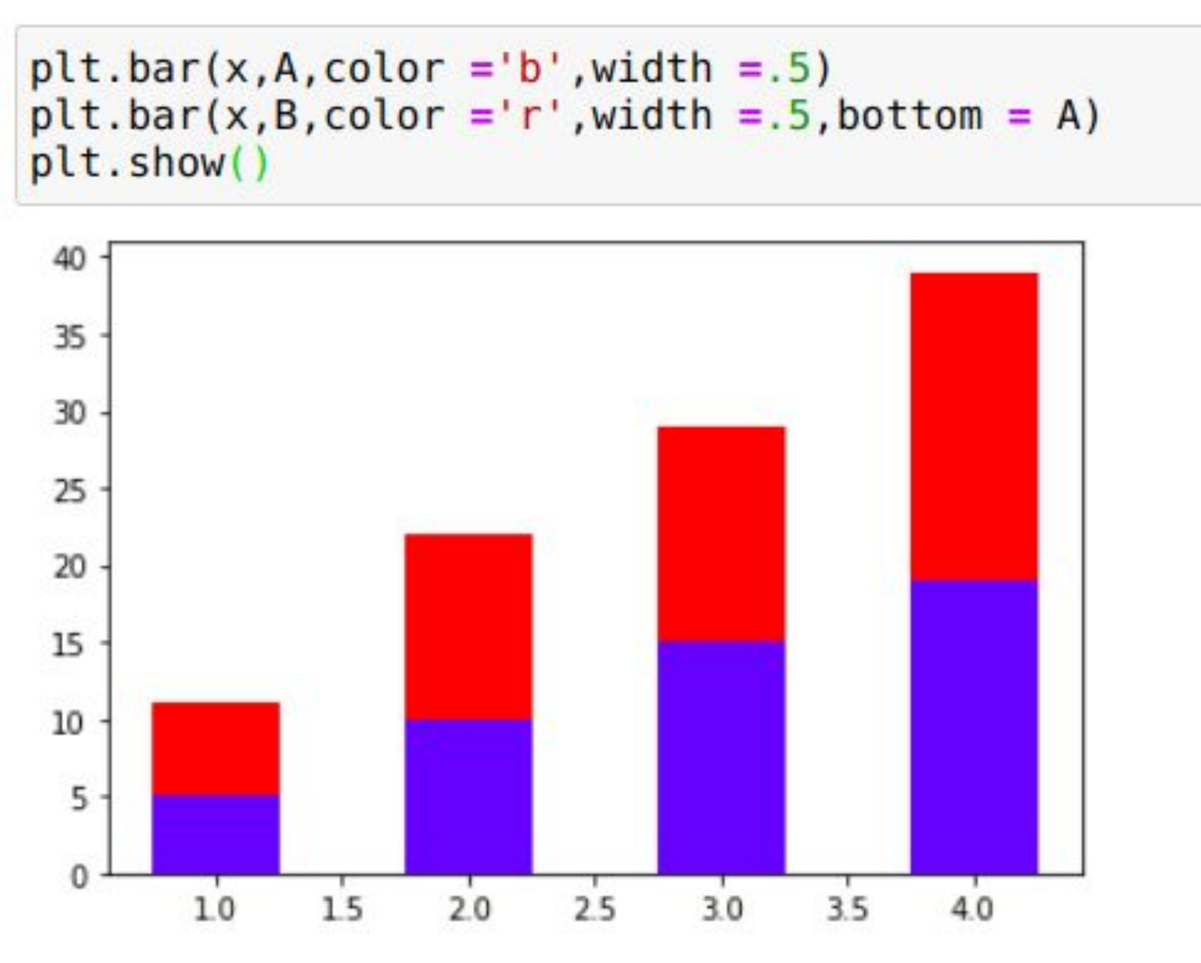
```
import numpy as np
import matplotlib.pyplot as plt

data = [[5., 25., 50., 20.],
        [4., 23., 51., 17.],
        [6., 22., 52., 19.]]
X = np.arange(4)
plt.bar(X + 0.00, data[0], color = 'b', width = 0.25)
plt.bar(X + 0.25, data[1], color = 'g', width = 0.25)
plt.bar(X + 0.50, data[2], color = 'r', width = 0.25)

plt.show()
```

3 groups of data in a Series object

There are 4 data points in each series.

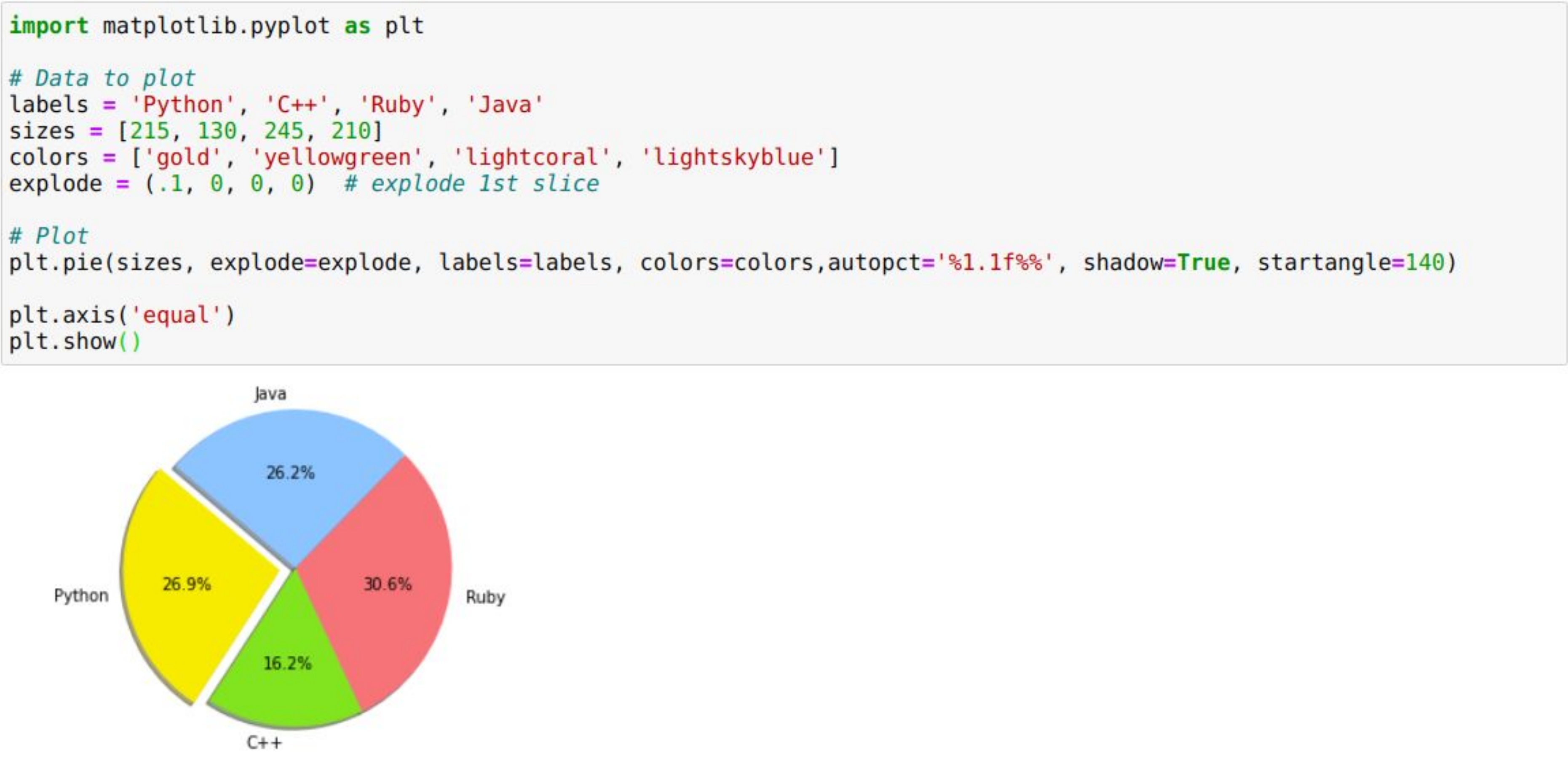
Colors added to each bar

Stacked Bars



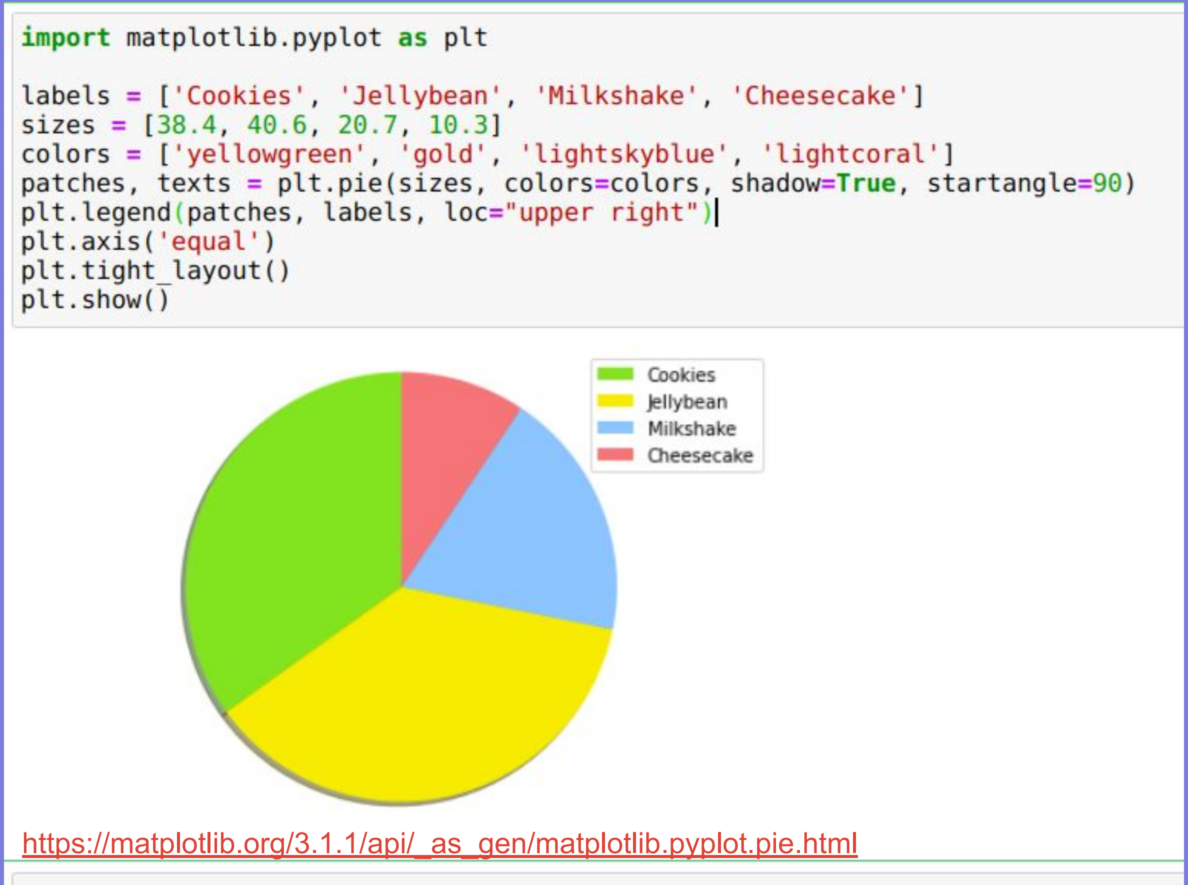
```
plt.bar(x,A,color ='b',width =.5)
plt.bar(x,B,color ='r',width =.5,bottom = A)
plt.show()
```

PIE PLOTS - 1



```
import matplotlib.pyplot as plt

# Data to plot
labels = 'Python', 'C++', 'Ruby', 'Java'
sizes = [215, 130, 245, 210]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (.1, 0, 0, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```

PIE PLOTS - 2



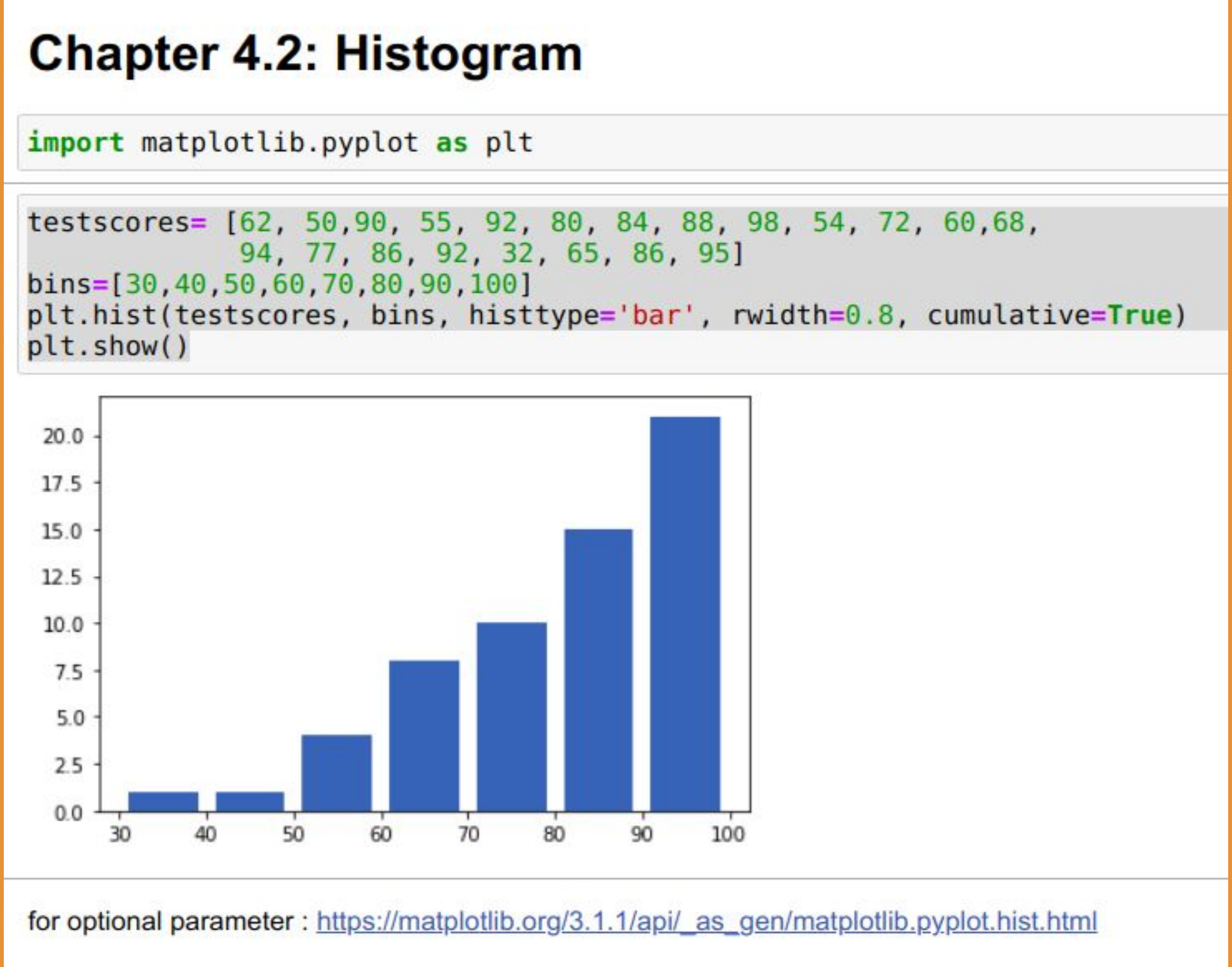
```
import matplotlib.pyplot as plt

labels = ['Cookies', 'Jellybean', 'Milkshake', 'Cheesecake']
sizes = [38.4, 40.6, 20.7, 10.3]
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
patches, texts = plt.pie(sizes, colors=colors, shadow=True, startangle=90)
plt.legend(patches, labels, loc="upper right")
plt.axis('equal')
plt.tight_layout()
plt.show()
```

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.pie.html

shadow, startangle, legends, autopct Are new options we have discovered

We will discuss customisation separately so don’t be anxious

HISTOGRAMS



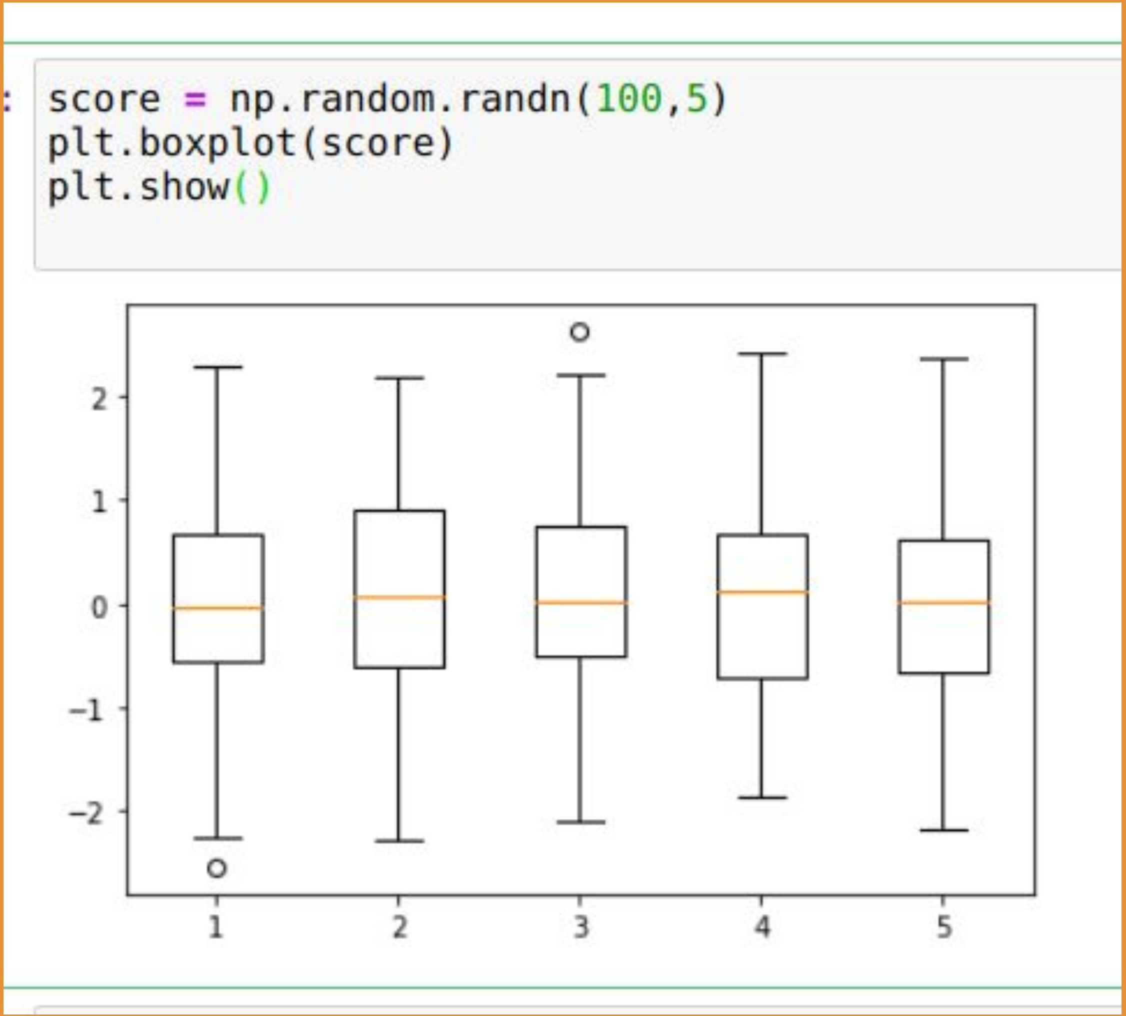
Chapter 4.2: Histogram

```python
import matplotlib.pyplot as plt
```

```python
testscores= [62, 50,90, 55, 92, 80, 84, 88, 98, 54, 72, 60,68,
             94, 77, 86, 92, 32, 65, 86, 95]
bins=[30,40,50,60,70,80,90,100]
plt.hist(testscores, bins, histtype='bar', rwidth=0.8, cumulative=True)
plt.show()
```

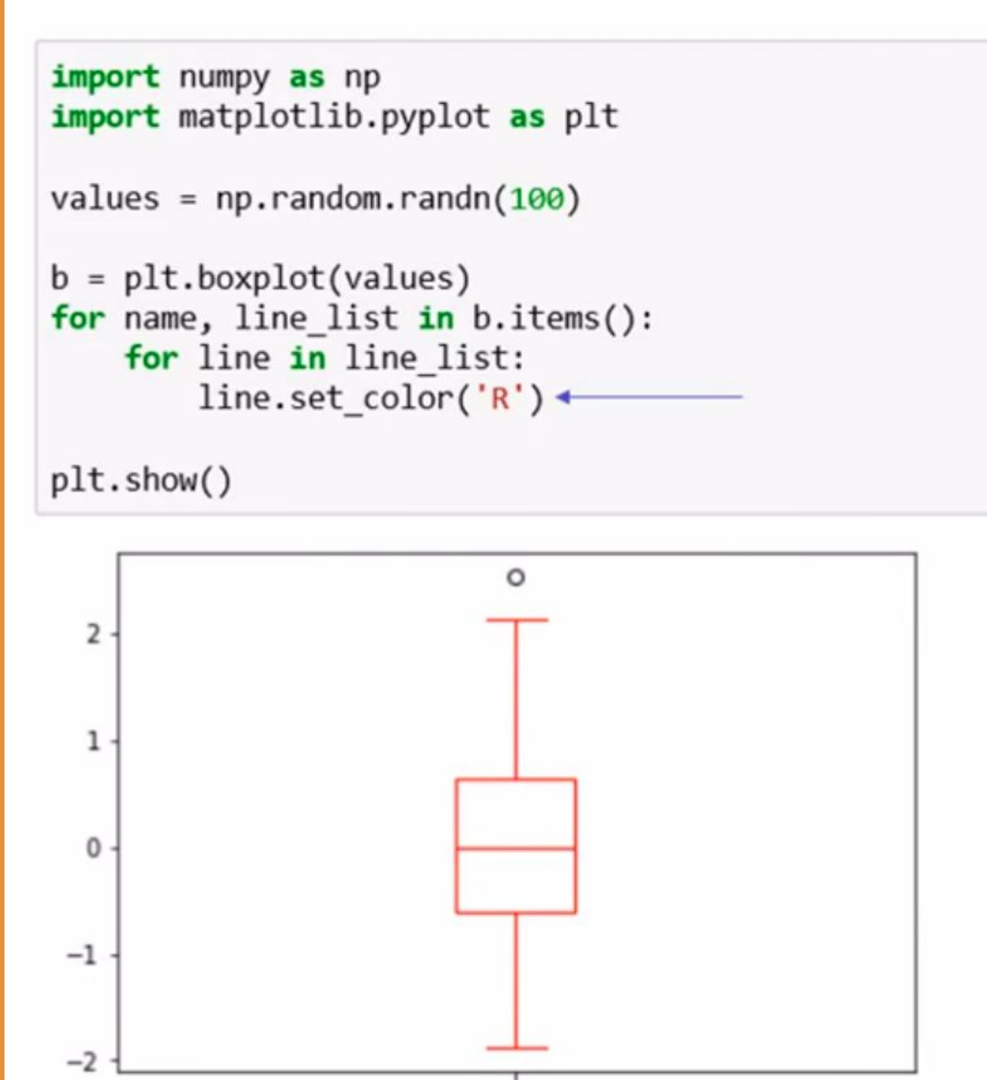
for optional parameter : https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hist.html

Box plot



```
score = np.random.randn(100,5)
plt.boxplot(score)
plt.show()
```

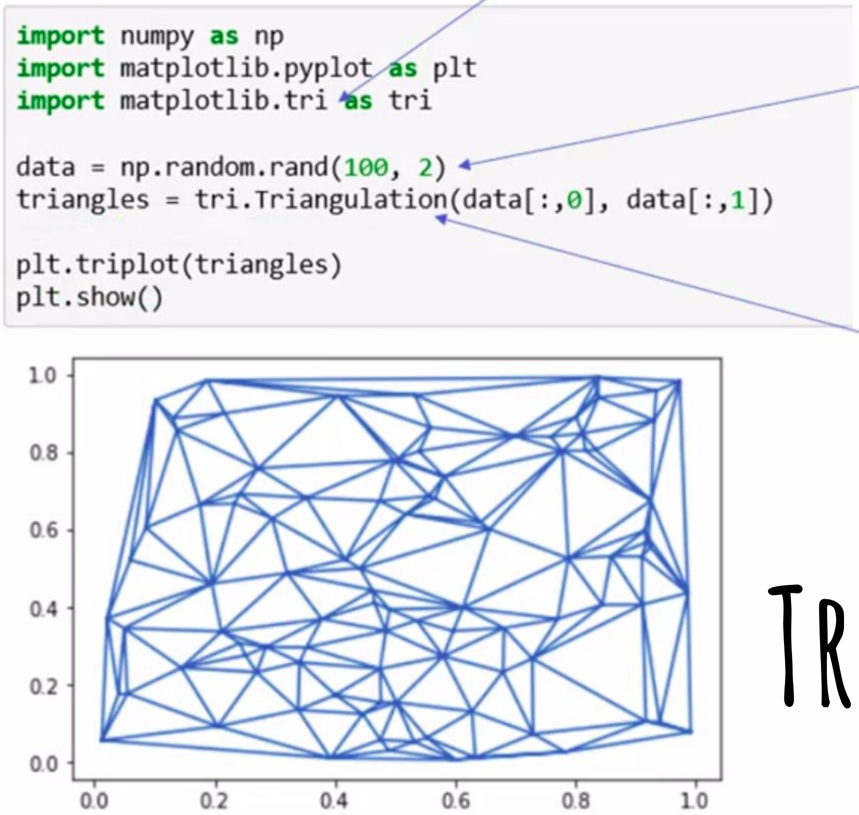
Box plot- Colour part of it



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

values = np.random.randn(100)

b = plt.boxplot(values)
for name, line_list in b.items():
    for line in line_list:
        line.set_color('R')

plt.show()
```

TRIANGULATION

The tri module.



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

data = np.random.rand(100, 2)
triangles = tri.Triangulation(data[:,0], data[:,1])

plt.triplot(triangles)
plt.show()
```

Generate a random cloud of points.

takes triangles as inputs and displays the triangulation result.

SUMMARY

- **Deferred Rendering** is the process of building the graph on the final call using **the show method.**
- The **plot and show functions** live inside **matplotlib**.
- When we **open a file from inside of a Jupyter Notebook** *without* specifying the directory that file must reside in the root python directory.
- When you **plot a graph** with multiple curves *matplot lib will change the colors automatically.*
- **matplotlib** only handles **plotting.**
- The **scatter function** allows us to create scatterplots.
- We use the **bar function** to craft bar charts.
- The **barh function** gives us the ability to create **horizontal bars.**
- The **width argument** allows us to **change the width of our bars** from within the bar function.
- We can use the **bottom argument** *within* the **bar function** to stack bars on top of one another.
- The **pie function** is used to create a **pie chart.**
- **Histograms** are plotted using the **hist function.**
- A histogram is **graphical display where the data is grouped into ranges** and then **plotted as bars.**
- The **bin argument** in the **hist function** provides us with the ability to *bin or group our data into various bars*... or bins.
- The **box plot** is a standardized way of displaying **the distribution of data based on the five number summary:** minimum, first quartile, median, third quartile, and maximum.
- The **boxplot function** is used to plot a boxplot.
- The **function** used to create *triangulated charts* is the **triangulation function.**

CUSTOMISATION

COLOR BASIC

‘color’
Or
‘c’
Is used to set color of plot function

TRIPLETS : rgb

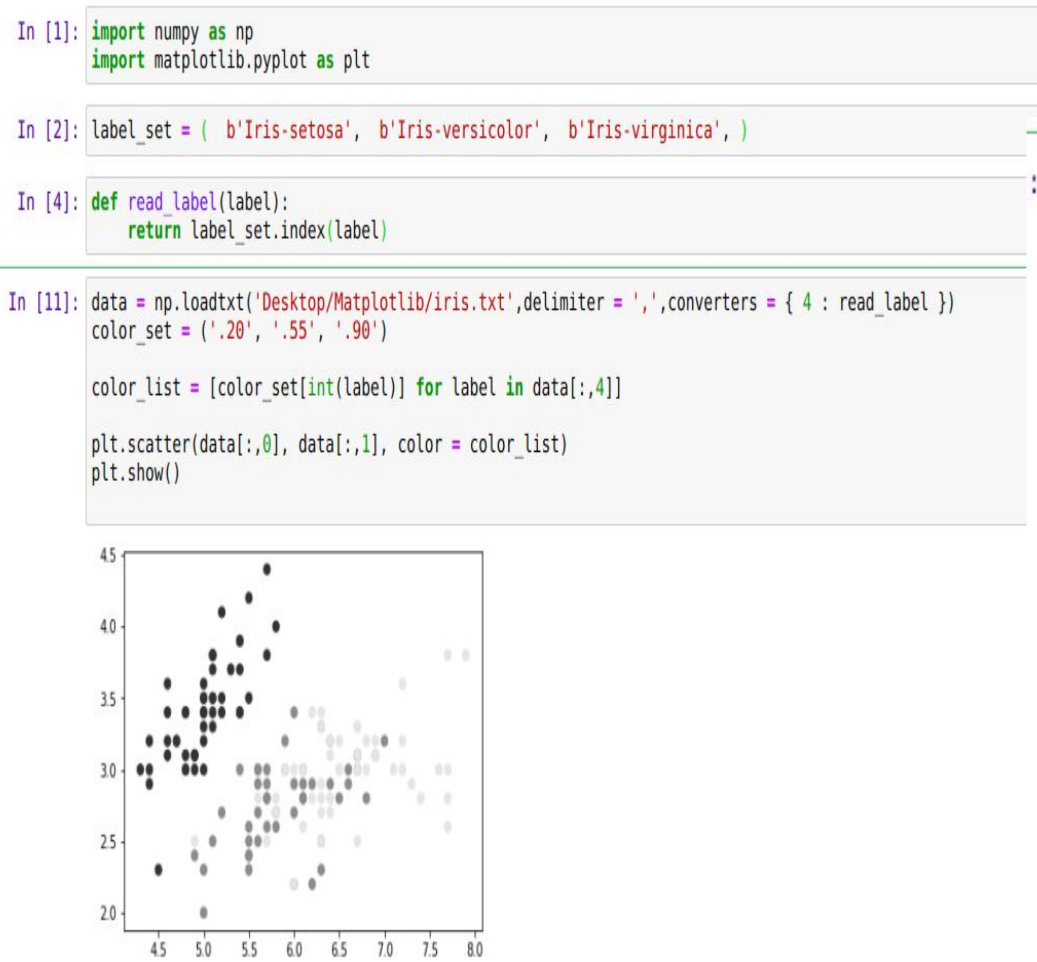
QUADRUPLES: rgba

PREDEFINED NAMES: blue,red

SCATTERPLOT

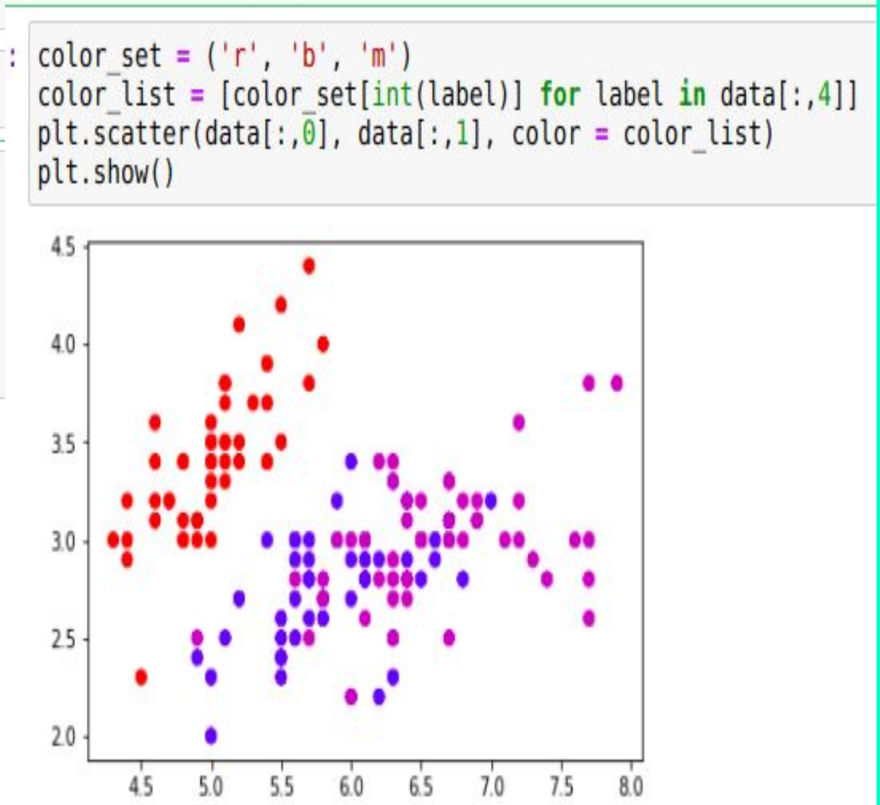
Marker color

Chapter 5 : Customisation of plots



```
In [1]: import numpy as np
        import matplotlib.pyplot as plt

In [2]: label_set = (  b'Iris-setosa',  b'Iris-versicolor',  b'Iris-virginica', )

In [4]: def read_label(label):
            return label_set.index(label)

In [11]: data = np.loadtxt('Desktop/Matplotlib/iris.txt',delimiter = ',',converters = { 4 : read_label })
         color_set = ('.20', '.55', '.90')

         color_list = [color_set[int(label)] for label in data[:,4]]

         plt.scatter(data[:,0], data[:,1], color = color_list)
         plt.show()
```

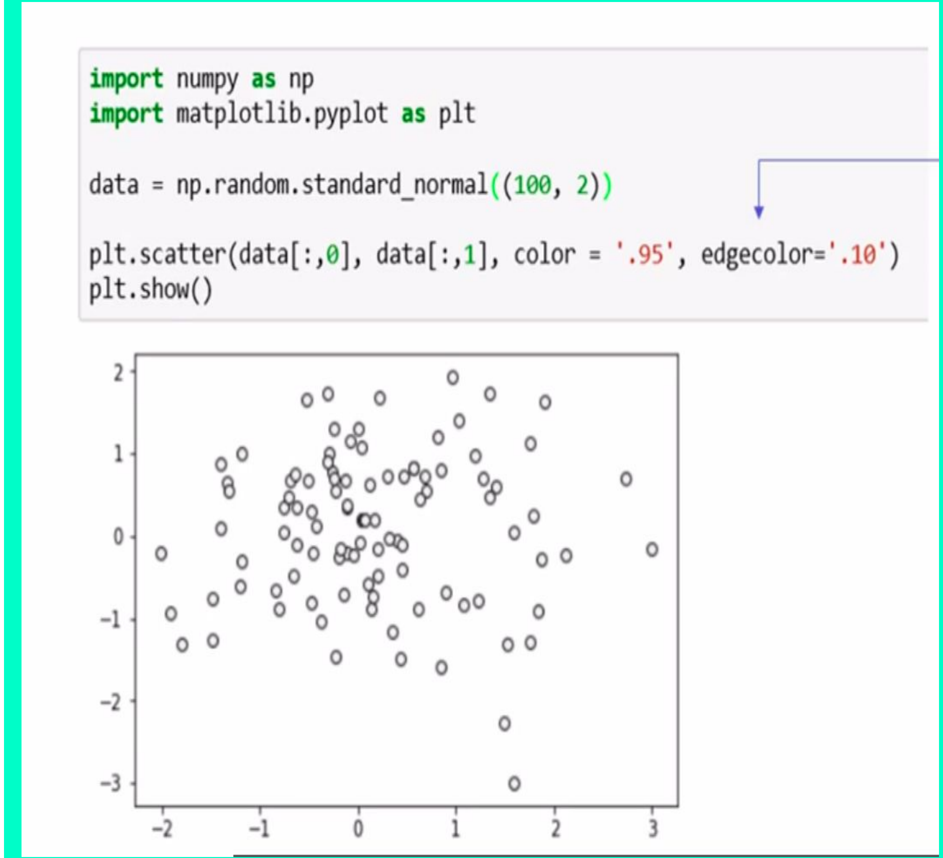
We have mapped labels with the colour in scatter plot to be able to distinguish
Gray scale and color both forms



```
: color_set = ('r', 'b', 'm')
  color_list = [color_set[int(label)] for label in data[:,4]]
  plt.scatter(data[:,0], data[:,1], color = color_list)
  plt.show()
```

SCATTER PLOT

EDGECOLOR

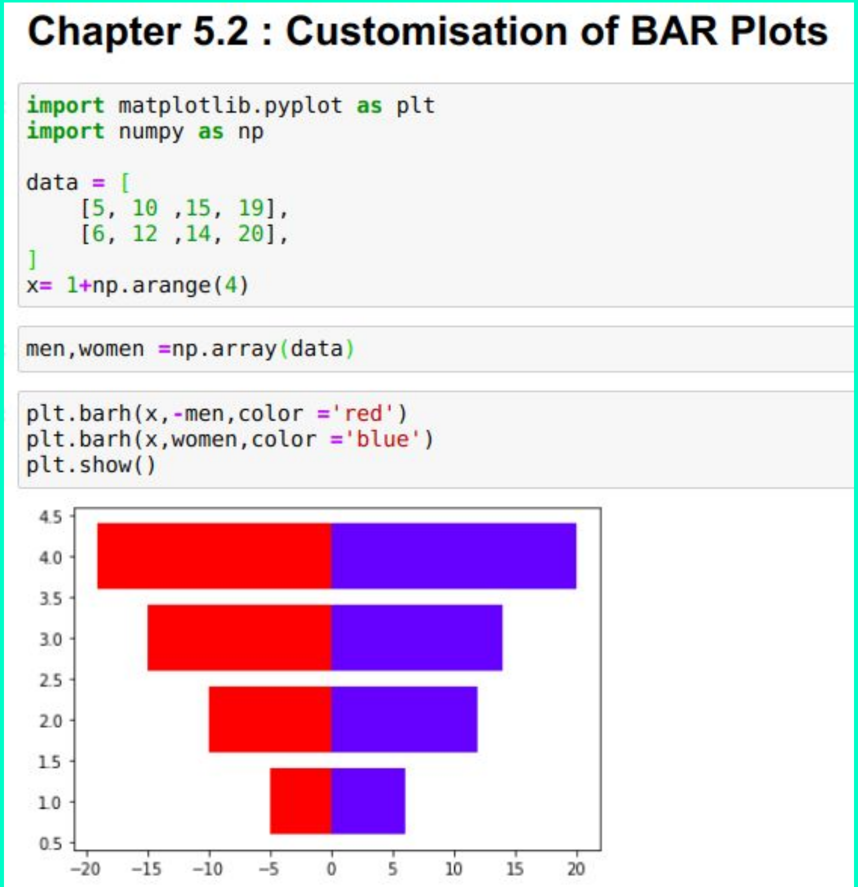


```
import numpy as np
import matplotlib.pyplot as plt

data = np.random.standard_normal((100, 2))

plt.scatter(data[:,0], data[:,1], color = '.95', edgecolor='.10')
plt.show()
```

BAR PLOT

BARPLOT

Color



Chapter 5.2 : Customisation of BAR Plots

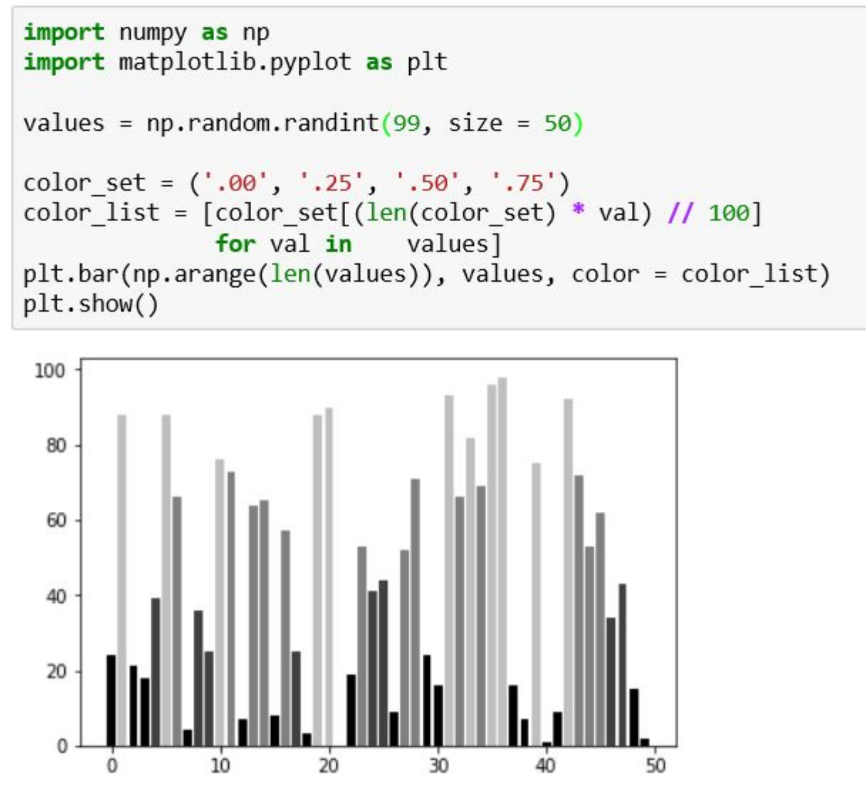
```
import matplotlib.pyplot as plt
import numpy as np

data = [
    [5, 10 ,15, 19],
    [6, 12 ,14, 20],
]
x= 1+np.arange(4)
```

```
men,women =np.array(data)
```

```
plt.barh(x,-men,color ='red')
plt.barh(x,women,color ='blue')
plt.show()
```

BARPLOT

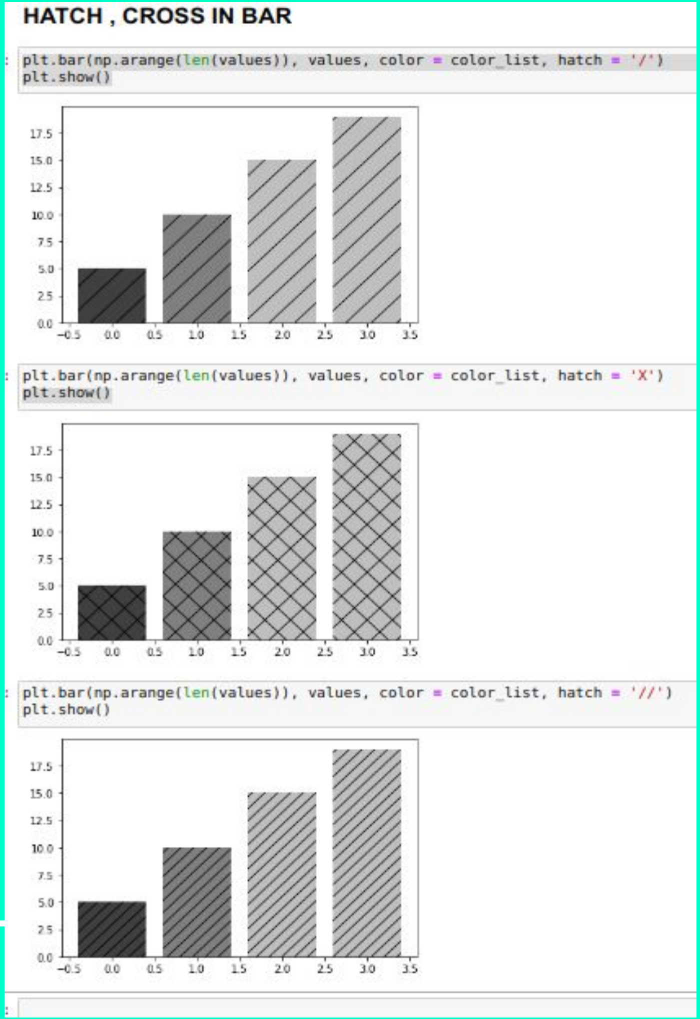
Dependent Values



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

values = np.random.randint(99, size = 50)

color_set = ('.00', '.25', '.50', '.75')
color_list = [color_set[(len(color_set) * val) // 100]
              for val in    values]
plt.bar(np.arange(len(values)), values, color = color_list)
plt.show()
```

BARPLOT

HATCHES



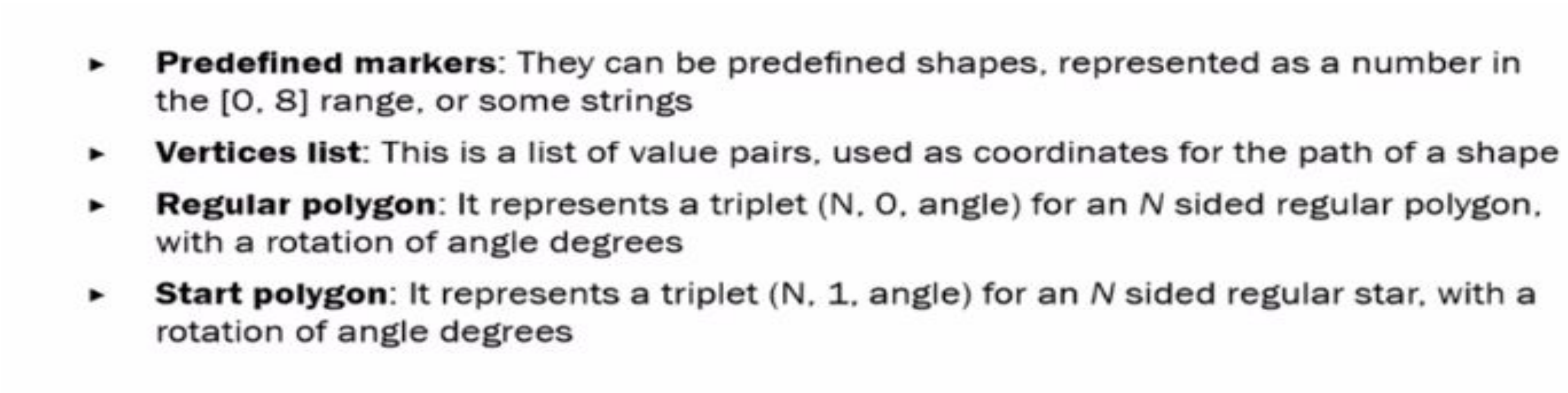
HATCH , CROSS IN BAR

```
plt.bar(np.arange(len(values)), values, color = color_list, hatch = '/')
plt.show()
```

```
plt.bar(np.arange(len(values)), values, color = color_list, hatch = 'X')
plt.show()
```

```
plt.bar(np.arange(len(values)), values, color = color_list, hatch = '//')
plt.show()
```

CREATING CUSTOM MARKERS



- **Predefined markers**: They can be predefined shapes, represented as a number in the [0, 8] range, or some strings
- **Vertices list**: This is a list of value pairs, used as coordinates for the path of a shape
- **Regular polygon**: It represents a triplet (N, 0, angle) for an *N* sided regular polygon, with a rotation of angle degrees
- **Start polygon**: It represents a triplet (N, 1, angle) for an *N* sided regular star, with a rotation of angle degrees

Properties :

```
plt.plot()
marker, markersize, markevery, markeredgecolor, markerfacecolor

plt.scatter()
Marker, s, c, edgecolor                 { s: size,c : color}
```

Note: marker = r’$\clubsuit$’ uses mathtext symbol as marker

Chapter 6.1 : MARKERS : Scatter Plot

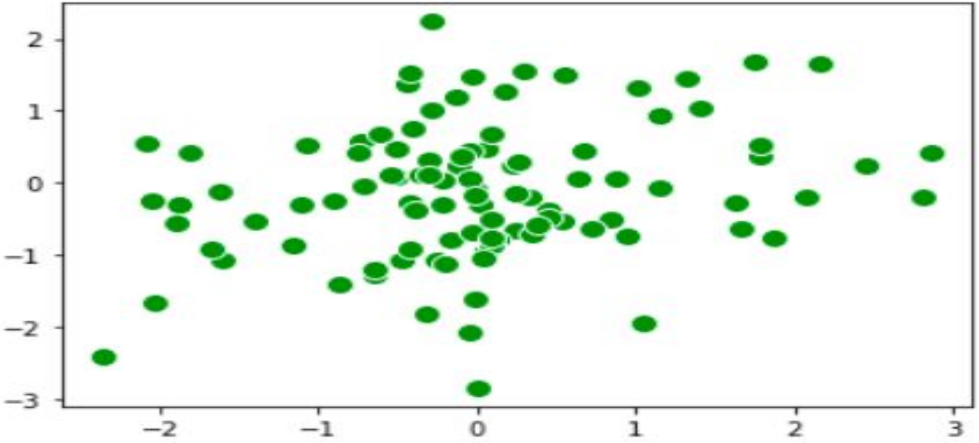
```
import numpy as np
import matplotlib.pyplot as plt
```

```
data = np.random.standard_normal((100,2))
```

Properties : TYPE: 'marker' and SIZE: 's'

Possible marker values : +, ^, *, o and [0,9] and
It can be (N,0,angle) or (N,1,angle) for regular polygon and star respectively
Vertices can be given as input to form shapes

```
plt.scatter(data[:,0], data[:,1],c= 'g', edgecolor = '1',marker = 'o', s = 100)
plt.show()
```



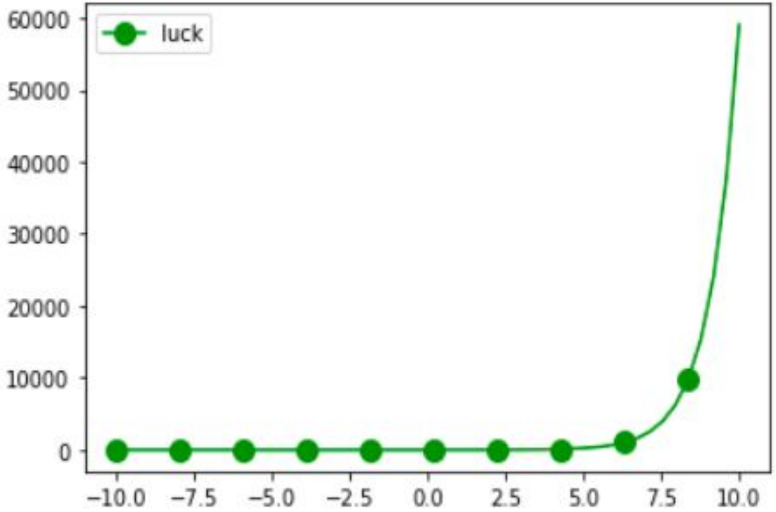
Chapter 6.2 : MARKERS : plt.plot()

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

```
x = np.linspace(-10,10)
```

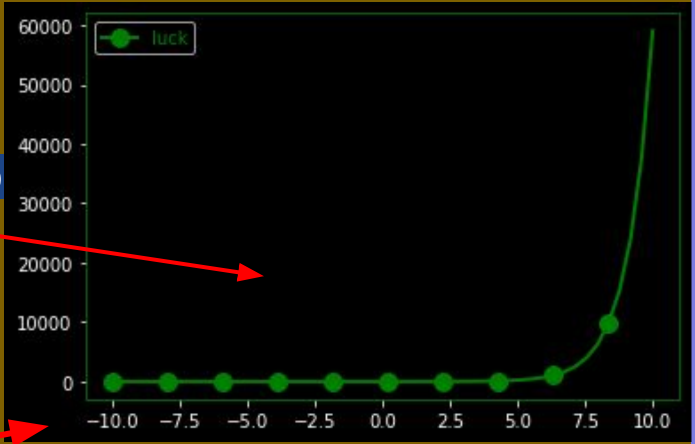
```
y = np.power(3,x)
```

```
plt.plot(x,y,c = 'green', markersize = 10,markevery = 5, marker = 'o',markerfacecolor='green',label='luck')
plt.legend(loc='best')
plt.show()
```



Custom color styles

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

mpl.rc('lines',linewidth = 2)

mpl.rc('axes', facecolor = 'k', edgecolor ='w')

mpl.rc('xtick',color = 'w')

mpl.rc('ytick',color = 'w')

mpl.rc('text', color = 'w')

mpl.rc('figure', facecolor = 'k', edgecolor ='w')
```



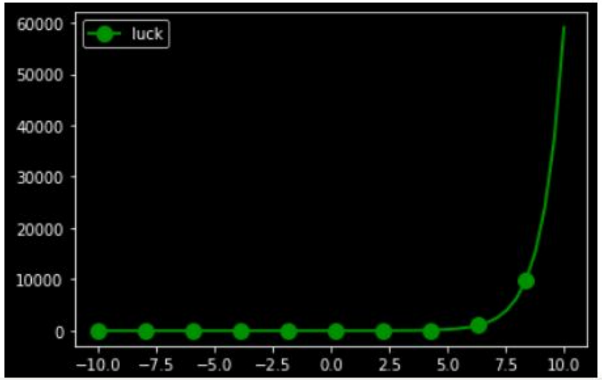
```
import  matplotlib.pyplot as plt
import numpy as np
import matplotlib as mpl
```

```
mpl.rc('lines',linewidth = 2)
mpl.rc('axes', facecolor = 'k', edgecolor ='w')
mpl.rc('xtick',color = 'w')
mpl.rc('ytick',color = 'w')
mpl.rc('text', color = 'w')
mpl.rc('figure', facecolor = 'k', edgecolor ='w')
```

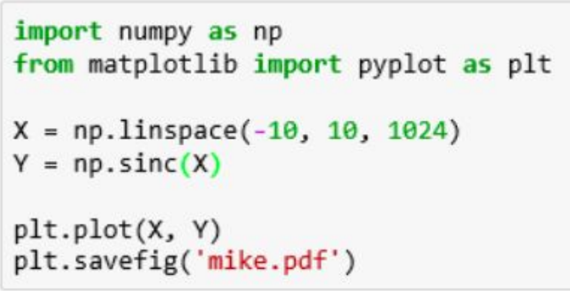
Matplotlib objects get their default colors from a centralized configuration object.

The rc object is that object in matplotlib

```
x = np.linspace(-10,10)
y = np.power(3,x)

plt.plot(x,y,c = 'green', markersize = 10,markevery = 5, marker = 'o',markerfacecolor='green',label='luck')
plt.legend(loc='best')
plt.show()
```



To save type Plt.savefig instead of plt.show()



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

X = np.linspace(-10, 10, 1024)
Y = np.sinc(X)

plt.plot(X, Y)
plt.savefig('mike.pdf')
```

Summary

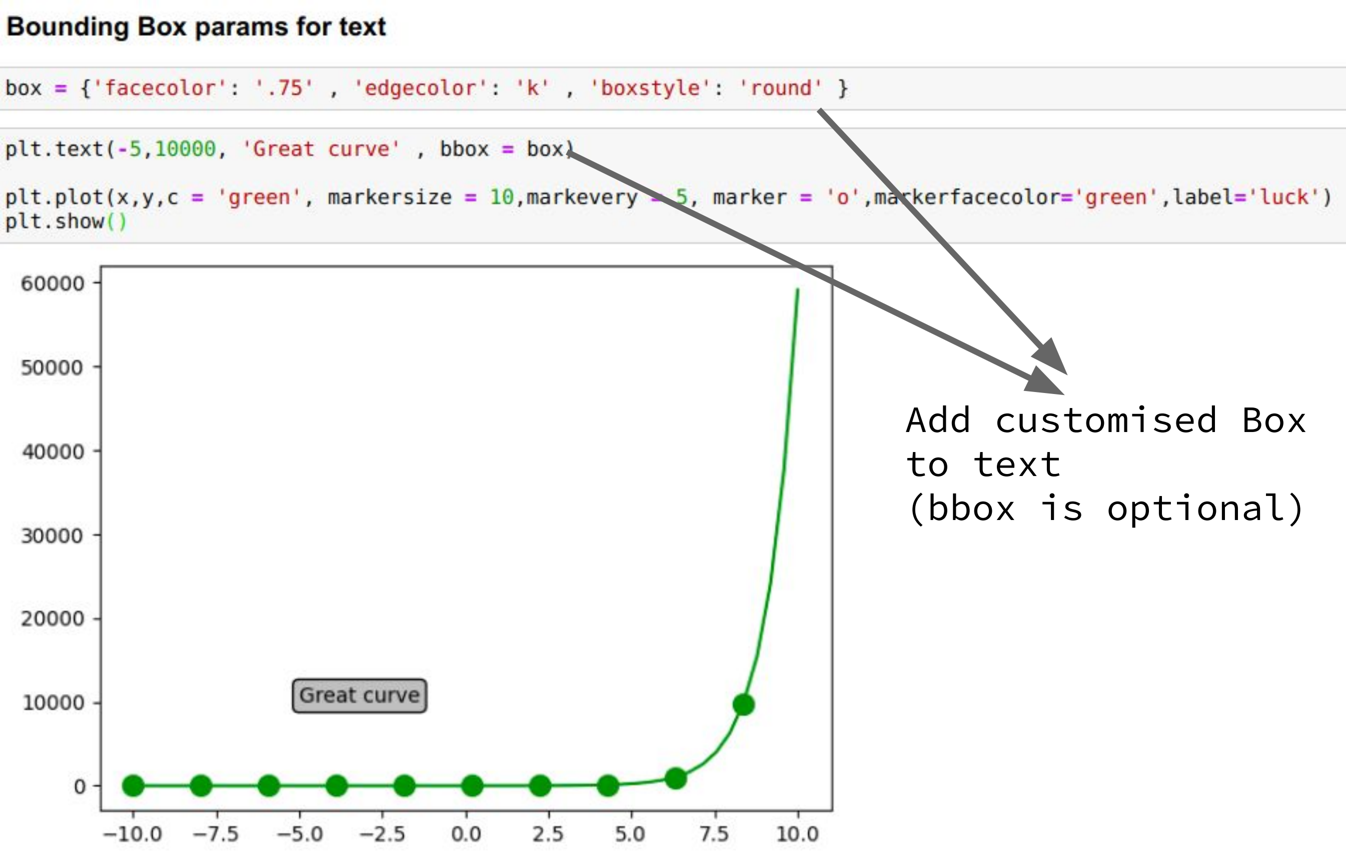
- The **scatterplot function** has a **color argument** that will allow us to alter the color of our plots.
- When we specify **numbers** in the **color parameter** matplotlib knows we want our output to be in black and white.
- The **closer we are to 1.0** the more **white** our output will be.
- The **closer we are to 0** the more **black** our output will be.
- The **edge color parameter** controls the **edge** of our **dots**. It works strictly with the color parameter.
- Altering the **colors** using the **color parameter** is the same regardless of chart type.
- The **linestyle parameter** gives us the ability to **alter the style of our lines.** We can solid lines, dashed and dashdot lines.
- The **pie function** creates a **pie chart.**
- We can use the **marker parameter** to change the marker in the scatterplot.
- Inside the scatterplot function **we have a letter s,** which stands for **size**, to alter the size of our markers.
- We can use the **markevery** parameter to **mark our charts at certain intervals.**
- We can use the **mathtext symbols** to create our own markers on our graphs.
- **All matplotlib objects choose** their color from a *centralized configuration object.*
- Every matplotlib object will **pick it's default settings from the object**, called the **rc object.**
- The **savefig function** gives us an easy way to **save our graphs to disk.**

ANNOTATIONS

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

plt.title(‘put your title’)
plt.xlabel(‘put x label’)
plt.ylabel(‘put y label’)

### Adding Text Anywhere

plt.text(-0.3, -0.28, 'Nice Curve')
```

This helps to Add text on plot

Adding text box anywhere



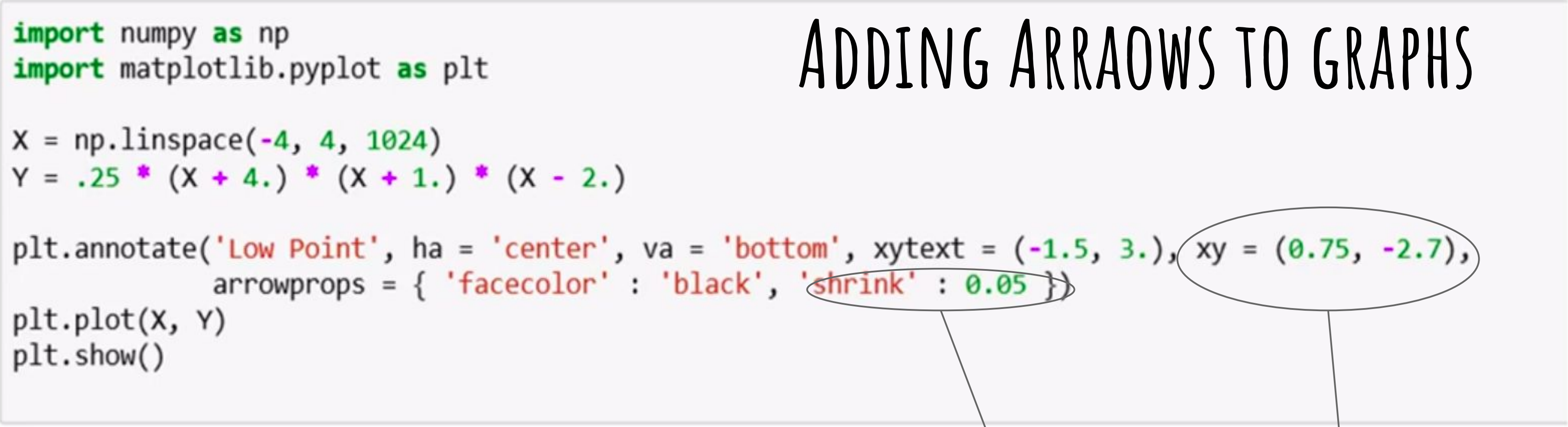
Bounding Box params for text

```
box = {'facecolor': '.75' , 'edgecolor': 'k' , 'boxstyle': 'round' }
```

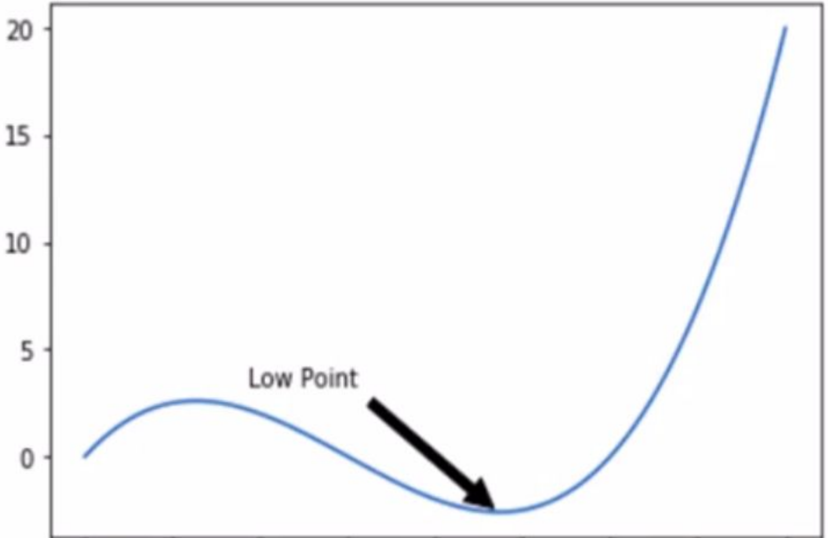
```
plt.text(-5,10000, 'Great curve' , bbox = box)

plt.plot(x,y,c = 'green', markersize = 10,markevery = 5, marker = 'o',markerfacecolor='green',label='luck')
plt.show()
```

Add customised Box to text
(bbox is optional)

Adding Arraows to graphs



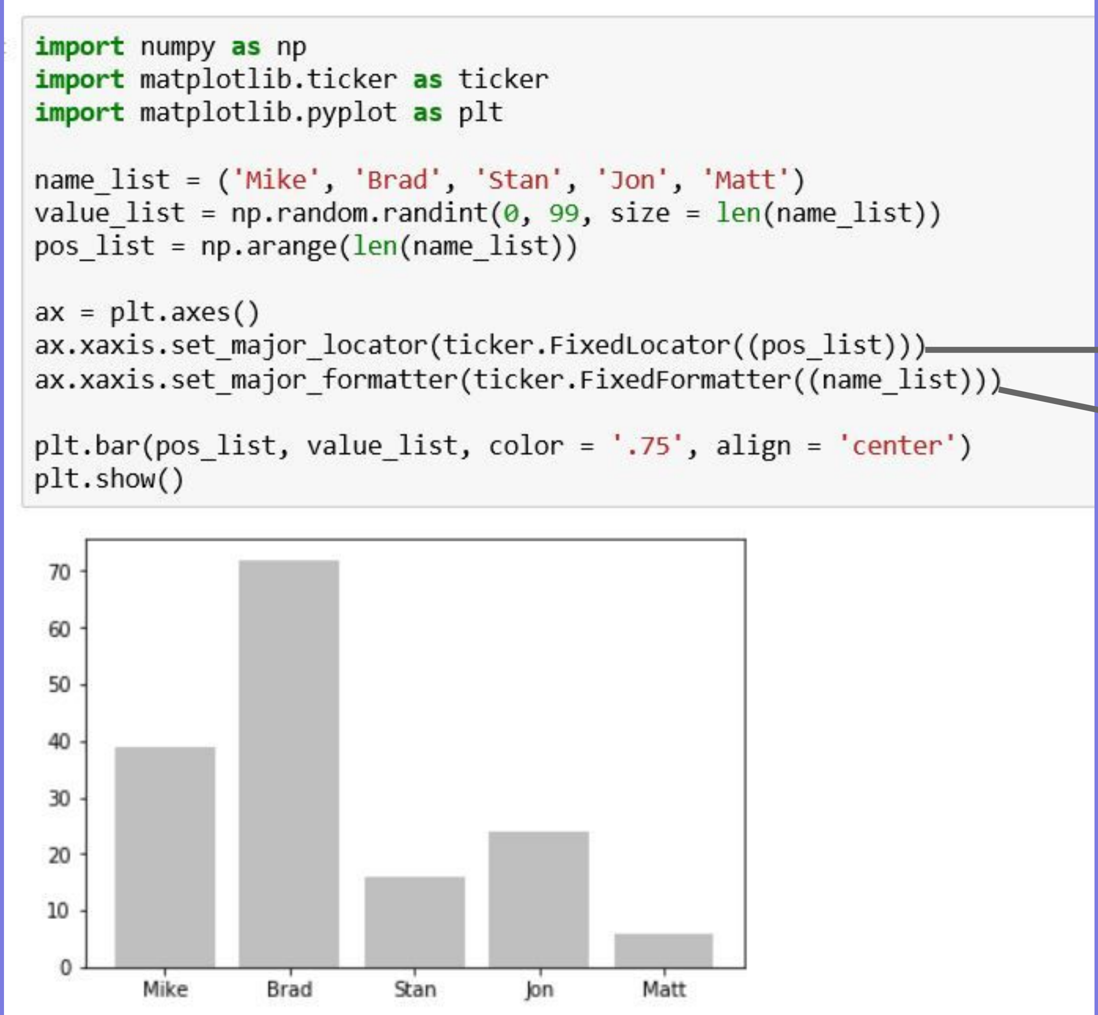
```
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-4, 4, 1024)
Y = .25 * (X + 4.) * (X + 1.) * (X - 2.)

plt.annotate('Low Point', ha = 'center', va = 'bottom', xytext = (-1.5, 3.), xy = (0.75, -2.7),
             arrowprops = { 'facecolor' : 'black', 'shrink' : 0.05 })
plt.plot(X, Y)
plt.show()
```



Distance of arrow from text

Destination of arraow

FORMATTING TICKS

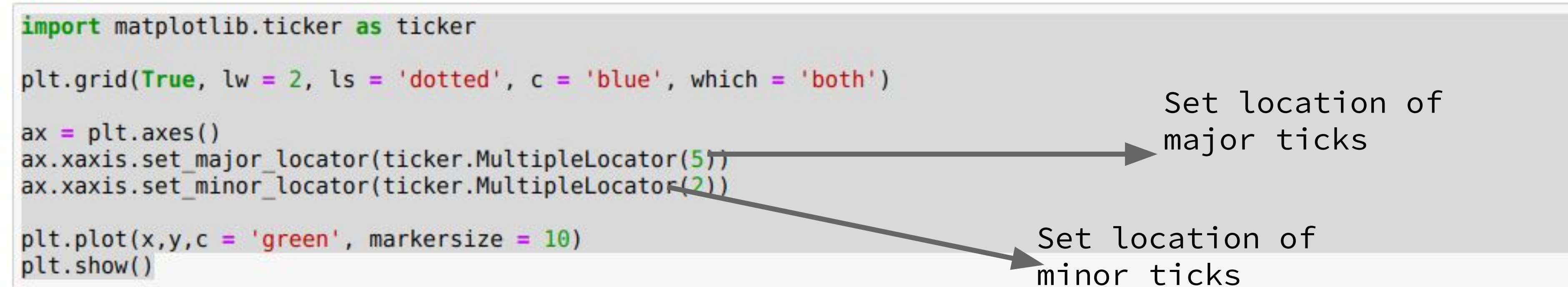


```
import numpy as np
import matplotlib.ticker as ticker
import matplotlib.pyplot as plt

name_list = ('Mike', 'Brad', 'Stan', 'Jon', 'Matt')
value_list = np.random.randint(0, 99, size = len(name_list))
pos_list = np.arange(len(name_list))

ax = plt.axes()
ax.xaxis.set_major_locator(ticker.FixedLocator((pos_list)))
ax.xaxis.set_major_formatter(ticker.FixedFormatter((name_list)))

plt.bar(pos_list, value_list, color = '.75', align = 'center')
plt.show()
```

Set location of major ticks

Set name of tiks

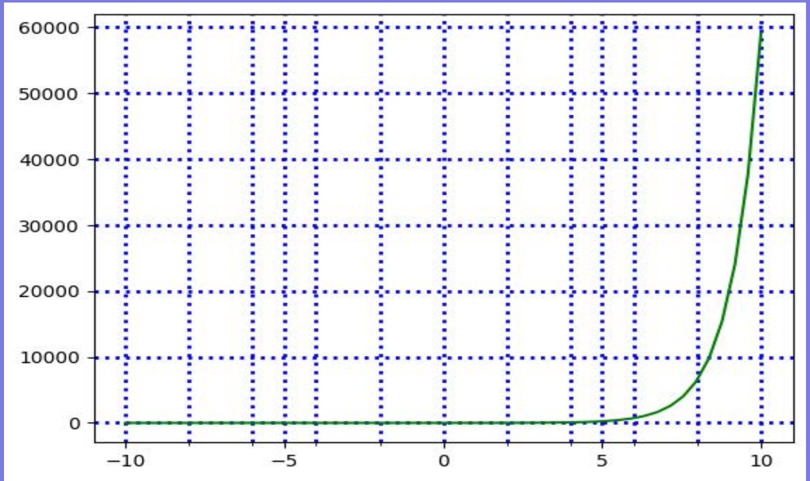
Formatting ticks



```
import matplotlib.ticker as ticker

plt.grid(True, lw = 2, ls = 'dotted', c = 'blue', which = 'both')

ax = plt.axes()
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_minor_locator(ticker.MultipleLocator(2))

plt.plot(x,y,c = 'green', markersize = 10)
plt.show()
```

Set location of major ticks

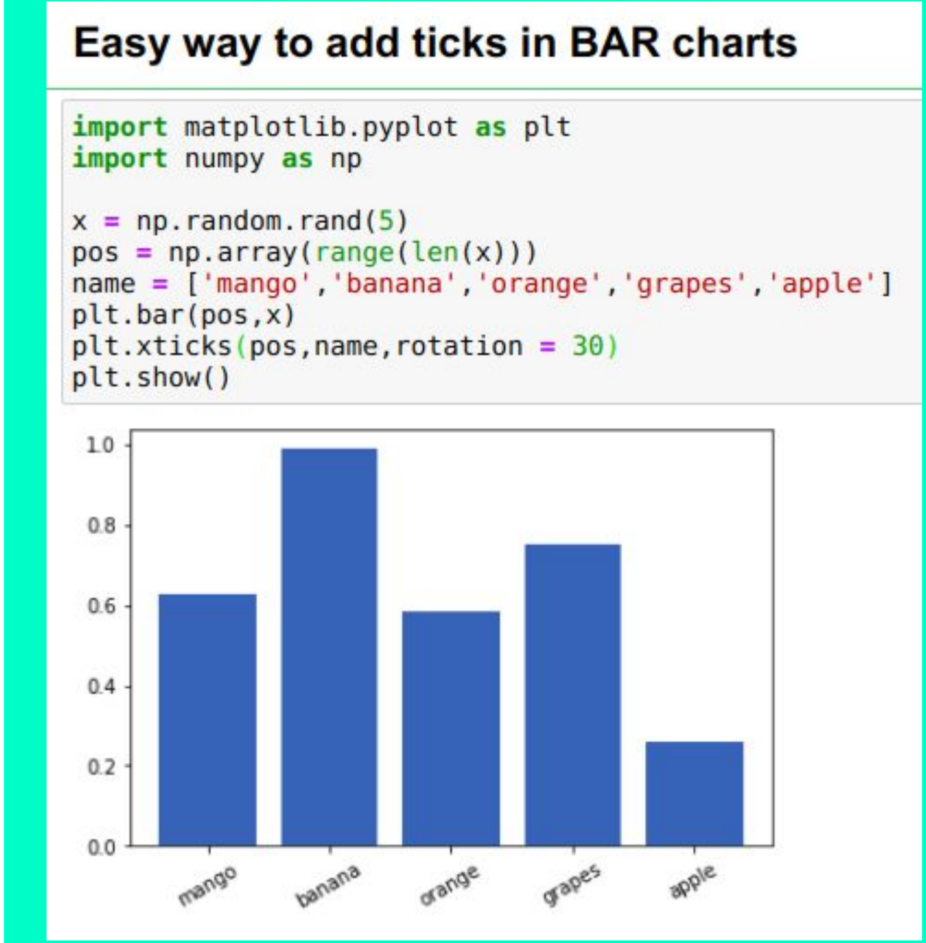
Set location of minor ticks



ADD TICKS

Easy way to add ticks

```
plt.xticks(pos,name,rotation)
```



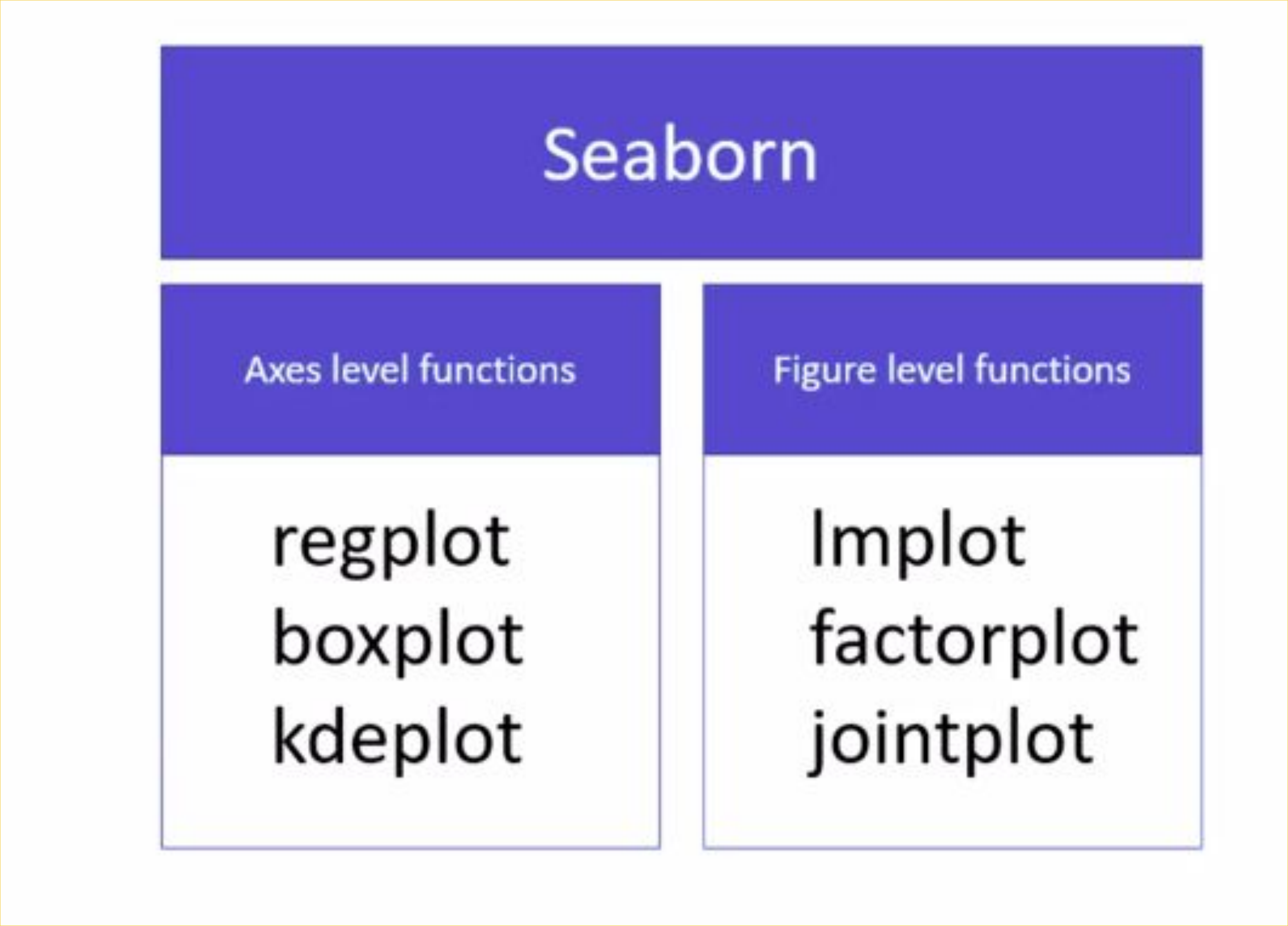
Easy way to add ticks in BAR charts

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

x = np.random.rand(5)
pos = np.array(range(len(x)))
name = ['mango','banana','orange','grapes','apple']
plt.bar(pos,x)
plt.xticks(pos,name,rotation = 30)
plt.show()
```

Summary

- The **title function** allows us to annotate our graphs by **adding at title.**
- The **xlabel and ylabel functions** allow us to label our **x and y axes.**
- The **text function** gives us the ability to place text **anywhere on our chart.**
- The **text function** supports a **bbox parameter** for adding bounded boxes to our charts.
- The **annotate function** gives us the ability to **add arrows to our charts.**
- The **axes function** manages the **axes** of a given chart.
- The **xticks function** provides us with a very easy way to **add ticks to our charts.**

SEABORN

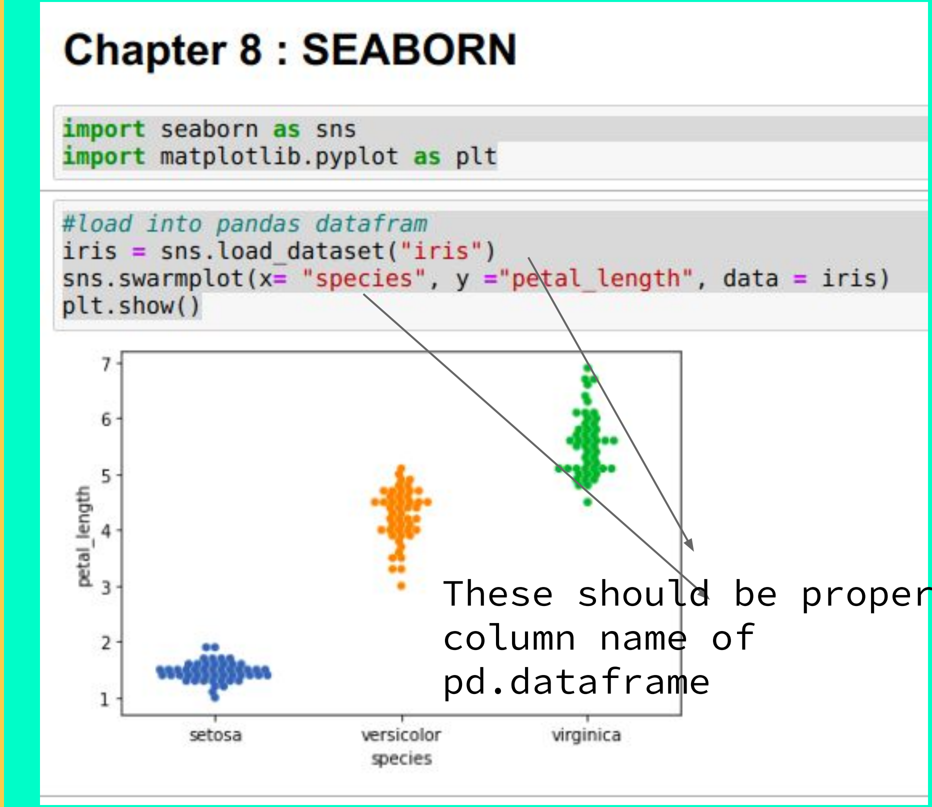


Seaborn

Axes level functions	Figure level functions
regplot	lmplot
boxplot	factorplot
kdeplot	jointplot

SEABORN

SITS ON MATPLOTLIB
HAS ITS OWN DATA TOO
OR
LOAD YOU DATA INTO PANDAS



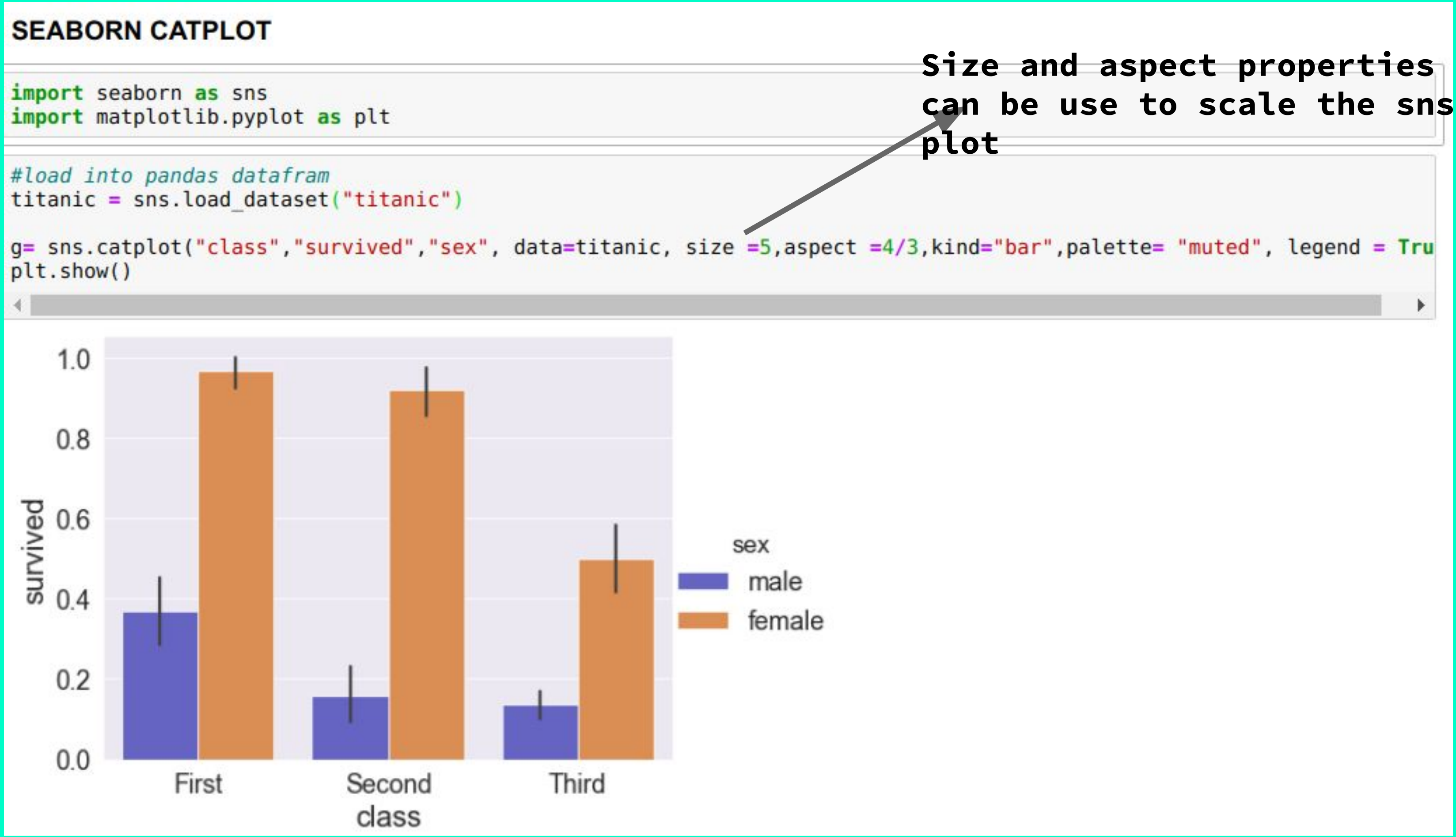
Chapter 8 : SEABORN

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#load into pandas datafram
iris = sns.load_dataset("iris")
sns.swarmplot(x= "species", y ="petal_length", data = iris)
plt.show()
```

These should be proper column name of pd.dataframe

Load Data to pd.DataFrame and use this seaborn command for quick analysis



SEABORN CATPLOT

```
import seaborn as sns
import matplotlib.pyplot as plt
```

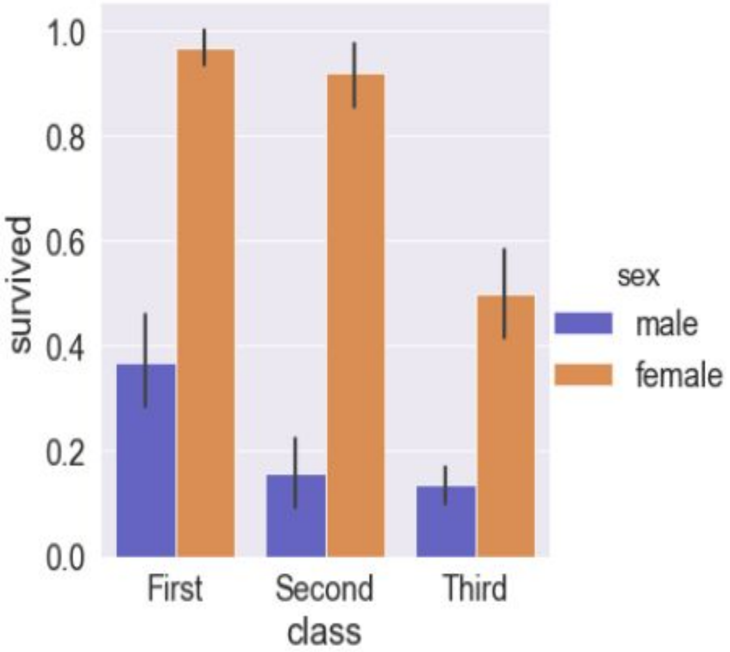
```
#load into pandas datafram
titanic = sns.load_dataset("titanic")

g= sns.catplot("class","survived","sex", data=titanic, size =5,aspect =4/3,kind="bar",palette= "muted", legend = Tru
plt.show()
```

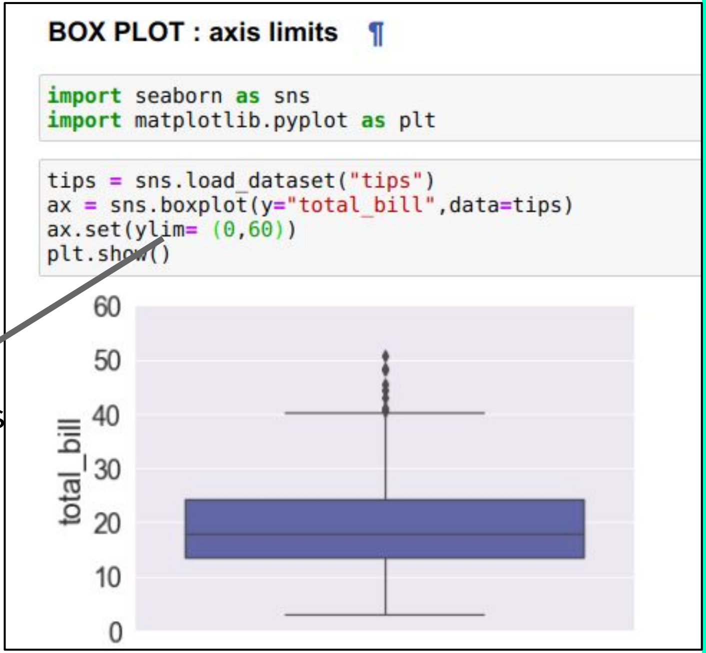
Size and aspect properties can be use to scale the sns plot

sns.set_context() : paper, talk, notebook, poster changes scaling

Scaling our seaborn plots : poster , talk, paper, notebook

```
sns.set() #default params
sns.set_context("paper", font_scale = 2,rc={"font.size":10,"axes.labelsize":20})   ##poster, talk, paper, notebook
g= sns.catplot("class","survived","sex", data=titanic, kind="bar",palette= "muted", legend = True)
plt.show()
```





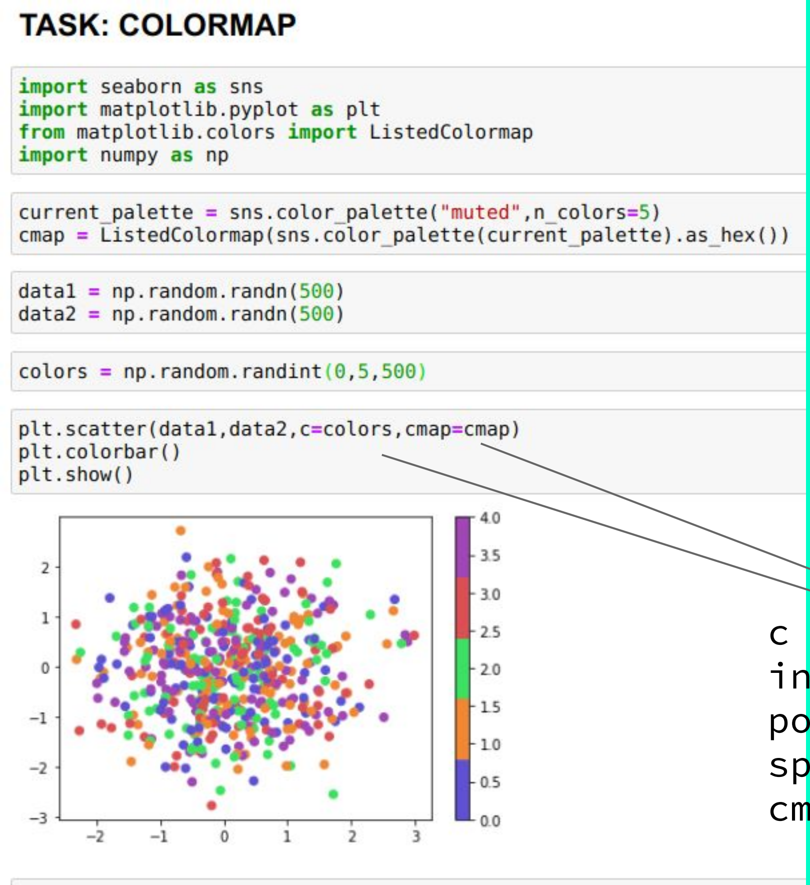
BOX PLOT : axis limits ¶

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
tips = sns.load_dataset("tips")
ax = sns.boxplot(y="total_bill",data=tips)
ax.set(ylim= (0,60))
plt.show()
```

Set axis limits

COLORMAP



TASK: COLORMAP

```
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import numpy as np
```

```
current_palette = sns.color_palette("muted",n_colors=5)
cmap = ListedColormap(sns.color_palette(current_palette).as_hex())
```

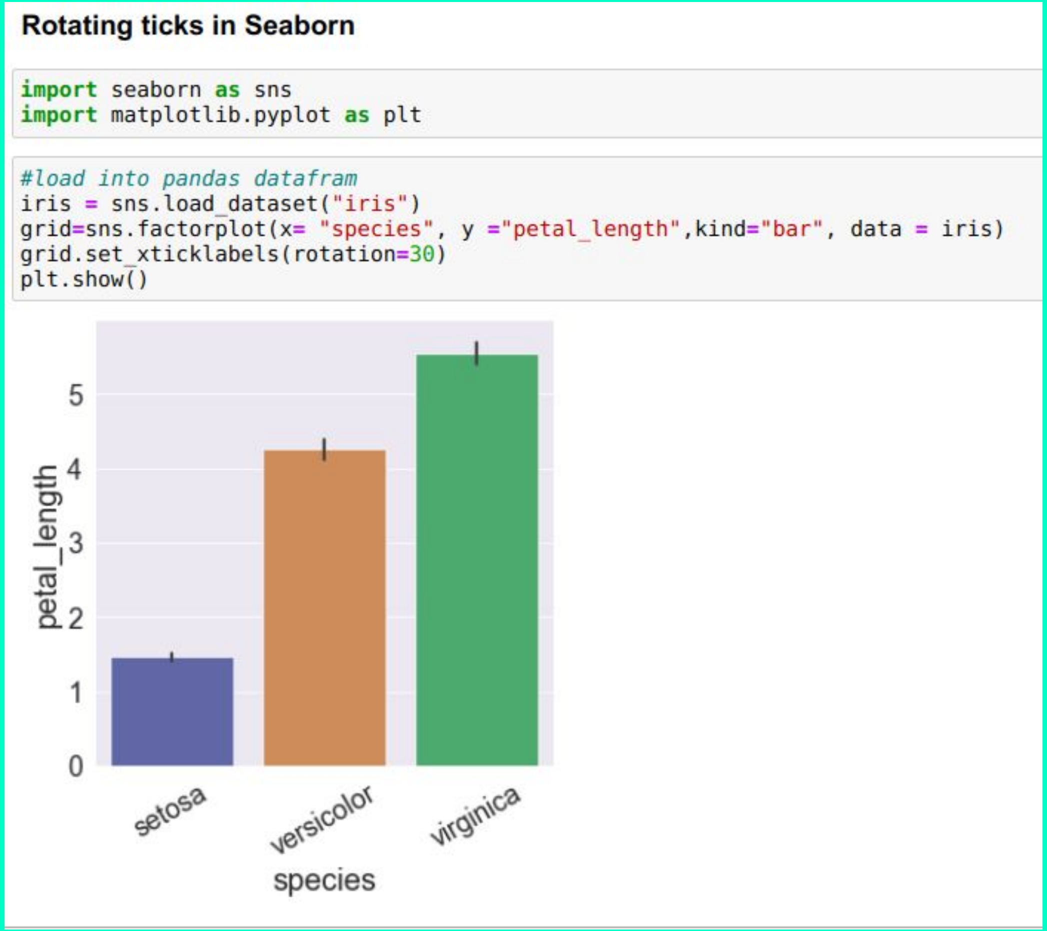
```
data1 = np.random.randn(500)
data2 = np.random.randn(500)
```

```
colors = np.random.randint(0,5,500)
```

```
plt.scatter(data1,data2,c=colors,cmap=cmap)
plt.colorbar()
plt.show()
```

c = colors contains index 1-5 for every point. 1-5 is mapped to specific colour using cmap

ROTATING TICKS IN SEABORN PLOTS

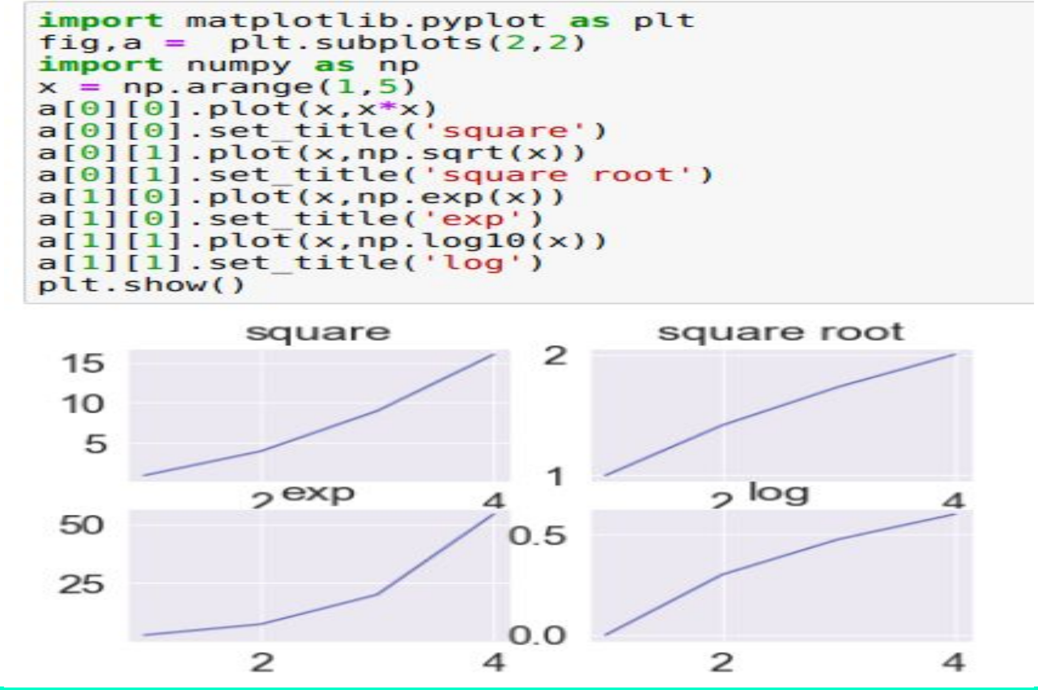


Rotating ticks in Seaborn

```
import seaborn as sns
import matplotlib.pyplot as plt
```

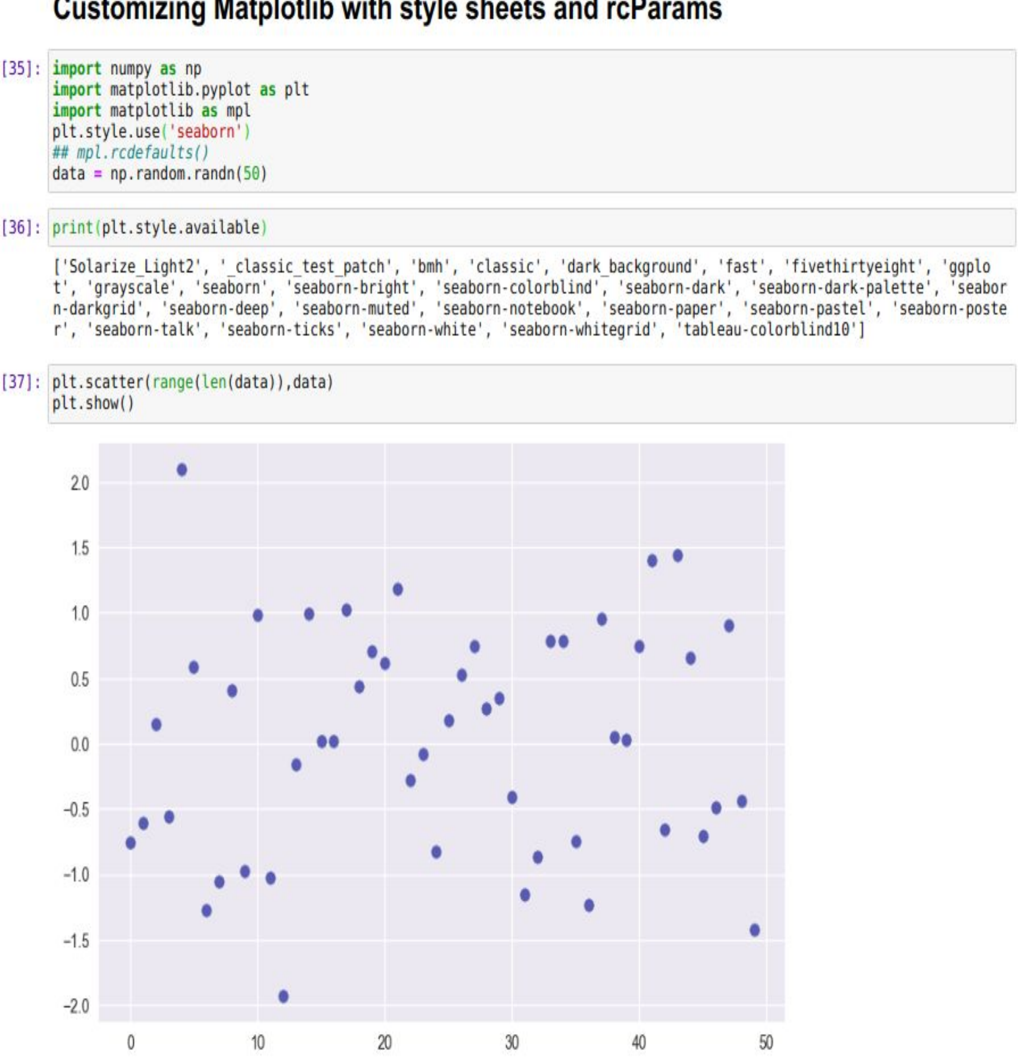
```
#load into pandas datafram
iris = sns.load_dataset("iris")
grid=sns.factorplot(x= "species", y ="petal_length",kind="bar", data = iris)
grid.set_xticklabels(rotation=30)
plt.show()
```

MISCELLANEOUS



```
import matplotlib.pyplot as plt
fig,a =  plt.subplots(2,2)
import numpy as np
x = np.arange(1,5)
a[0][0].plot(x,x*x)
a[0][0].set_title('square')
a[0][1].plot(x,np.sqrt(x))
a[0][1].set_title('square root')
a[1][0].plot(x,np.exp(x))
a[1][0].set_title('exp')
a[1][1].plot(x,np.log10(x))
a[1][1].set_title('log')
plt.show()
```

SUBPLOTS: PLOTTING DIFFERENT PLOTS ON SAME GRAPHS



Customizing Matplotlib with style sheets and rcParams

[35]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
plt.style.use('seaborn')
## mpl.rcdefaults()
data = np.random.randn(50)
```

[36]:
```
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplo
t', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette', 'seabor
n-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poste
r', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

[37]:
```
plt.scatter(range(len(data)),data)
plt.show()
```

Using predefined Stylesheets To customise looks