Within ARTISTS you have figures. Within each FIGURE you have AXES which is the plot itself containing details eg: data points, labels styling positioning etc.

Matplotlib

Multiple plots on a figure

Single plot

Basic Plot structure:

```
plt.plot()      #(1) Initialise with x and y co-ordinates and style the line
<plt details>   #(2) Details –includes: labels, axis resolution
plt.show()      #(3) Display graph
```

(1) Create a FIGURE instance:

```
fig = plt.figure(figsize=(15,5))
#Common sizes: (10,7.5) or (12,9)
```

(2) Create many AXES (Subplots):

```
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)
```

If you wanted to make a SINGLE plot:

```
ax = fig.add_axes([x0,y0,w,h])
```

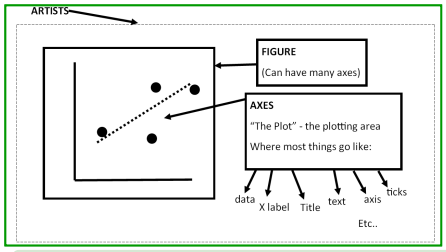
x0,y0 - are origin positions of where you want the AXES (plot) to start drawing from.
h,w - are height and width of plot

A side note on Subplots:
If you want many AXES (Plots) in one FIGURE, you can use subplots:
[] [] []
[] [] []
This is 2 row 3 column arrangement of 6 plots.

```
fig.add_subplot(row,col,count)
```

Count means which plot we are working on. So, we need to go through each plot and define its graph design and data. So the first plot is:

```
plt.subplot(2,3,1)  # row 1 col 1
<make the plot>
plt.subplot(2,3,2)# row 1 col 2
<make the plot>                    …and so on….
```



Add Details - Types of plots

```
.plot([x],[y],<more attributes>)
.fill_between([x],[y],<more attributes>)
.bar([x],[y],width=2,<more attributes>)
.scatter([x],[y],<more attributes>)
.hist([x],bins=5,<more attributes>)
.scatter([x],[y],s=z)#Bubble plot: "z" as size
```

#.plot is a line graph plot. and fill_between is the same #except that the graph underneath is filled with color.

Additional plot attributes:

```
linestyle='none/-/--'
linewidth = 6
marker='D/x/o/*'
markerfacecolor='skyblue'
markersize=16
markeredgecolor="orange"
markeredgewidth=5
alpha=0.3 #Transparancy
color=#FFCCEE
```

eg: ax.scatter([x],[y],marker="D")

Importing Matplotlib:

```
%matplotlib inline
import matplotlib.pyplot as plt
```

A note on the structure of data

Instead of using [X] and [Y] lists of data for plotting, you can use dataframe columns df["x"] and df["y"]

(3) Add details

Add Details - Plot Text:
Place text on position (1.5,1):

```
plt.text(1.5,1,"txt")
```

Additional text attributes:

```
color='black'
weight='semibold'
fontsize=12
fontweight=0
```

Add Details - Labels

```
ax.set_xlabel("Text")
ax.set_ylabel("Text")
ax.set_title("Title")
```

Add Details - Plot Grid:

```
plt.grid(which='major/minor/both',
axis='both/x/y, <additional attr>"
```

Additional attributes include: color,linestyle linewidth, alpha

Add Details - Axis Details:
Setting limis to the axis

```
ax.set_xlim(Xmin, Xmax)
ax.set_Ylim(Ymin,Ymax)
```

Setting custom ticks

```
ax1.xaxis.set_ticks(list_ticks)
ax1.yaxis.set_ticks(list_ticks)
```

Note: you can use this as list_ticks:

```
np.arange(min,max,increment)
```

Note:
Find Min / Max value of a column:

```
min = df["colname"].min()
```

Add Details - Annotation:
#This will display text at "xytext" with an arrow pointing to your specified point at "xy"

```
ax.annotate(s="text", xy=(4,40), xytext=(6,50), arrowprops=dict(facecolor="green"))
```

Add Details - Legends
As an attribute to your type of plot, include:

```
label = "data reference text"
```

Also include in your code:

```
ax.legend()
```

Add Details - Plotting with Datetime:
One of your axis has datetime data. Your will plot with this data however it needs formatting:

```
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
ax.xaxis.set_minor_locator(MaxNLocator(integer=True))
```

Warning - If you are plotting into a bar, scatter or .fill_between, you will get an error. To avoid this, set datetime data as your index. Now reference your datetime column like this:

```
dataframe.index.value  # this gives a list of your index values.
```

NOTE: ensure the working Dataframe has numerical columns and NOT objects. Use df.info() to check datatypes. If you see objects do: df = df.apply(pd.to_numeric, errors="ignore")

Eg: ax.fill_between(df.index.values, df["y-col-data"], color="green", alpha=0.3)

Dont forget the imports!

```
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.ticker import MaxNLocator
```

#(1) Initialize

```
plt.plot(x,y,style)
```

Put a list of x and y co-ordinates

```
x=[1,2,3]#row
y=[1,2,3]#col
style = "y-"
plt.plot(x,y,style)
```

What if data is in a different format like in a Dataframe:
You can directly put Dataframe columns in: df.colx df.coly
Or you can convert whatever data you have to np.array objects prior to plotting.
The Dataframe needs 2 columns representing x and y.

```
data = [[0,10],
        [1,20],
        [2,30]]
df = pd.DataFrame(data,columns=['myx','myy'])
df_array = df.values
plt.plot(df_array,style)
```

Style:
note plt.plot() takes two variables: "data" and "style":
"r-" means red line
"go" means green circles
"bs" means blue squares
"y^" means yellow triangles
Simply plug in any combination of colour and shape.

#(2) Plot details:

```
plt.title("My Graph")
plt.xlabel("X Axis")
plt.ylabel("Y Axis ")
#Where you want the axis values to start/end:
plt.axis([x_min, x_max, y_min, y_max])
#How you want the ticks displayed:
plt.xticks(np.arange(min,max,increment))
plt.xticks(np.arange(min,max,increment))
```

#(3) Display graph:

```
plt.show()
```

Seaborn
If you know how to make a chart with Matplotlib, just load the seaborn library and your chart will look better with extra styling options:

```
import seaborn as sns
sns.set_style("darkgrid/whitegrid/dark/white/ticks")
```

plots:

```
sns.scatterplot(df["x"],df["y"],data=df,hue=df[z])
#hue is a 3rd "color/classification" dimension
sns.barplot(df["x"],df["y"], data="df")
sns.distplot(df["count_col"], bins=20)
```

Same additional attributes apply