Lecture 4

Matplotlib Overview

Importing Library

Import the `matplotlib.pyplot` with `plt`

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

You'll also need to use this line to see plots in the notebook:

In [2]:
```
%matplotlib inline
```

That line is only for jupyter notebooks, if you are using another editor, you'll use: **plt.show()** at the end of all your plotting commands to have the figure pop up in another window.

Example with numpy

Let's walk through a very simple example using two numpy arrays. You can also use lists, but most likely you'll be passing numpy arrays or pandas columns (which essentially also behave like arrays).

** The data we want to plot:**

In [3]:
```
import numpy as np
x = np.arange(0,20,2)
y = x ** 2
```

In [115]:
```
x
```

Out[115]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

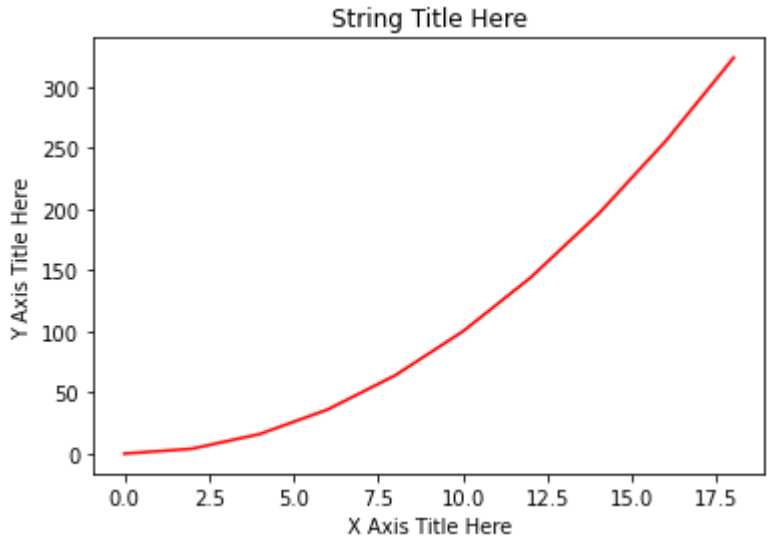
In [116]:
```
y
```

Out[116]:
```
array([  0,   4,  16,  36,  64, 100, 144, 196, 256, 324], dtype=int32)
```

Basic Matplotlib Commands

We can create a very simple line plot using the following.

In [4]:
```
plt.plot(x, y, 'r') # 'r' is the color red
plt.xlabel('X Axis Title Here')
plt.ylabel('Y Axis Title Here')
plt.title('String Title Here')
plt.show()
```



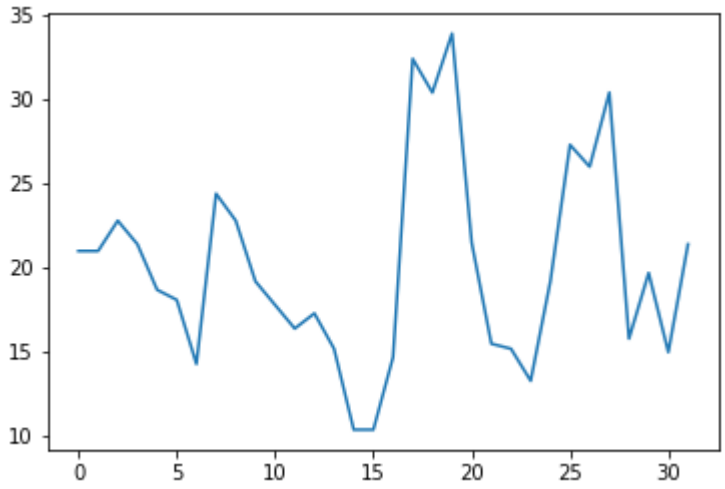
In [80]:
```
#Example
x = range(1,10)
y = [1,2,3,4,0,4,3,2,1]


```

Now, we want to show the line chart of the real dataset.First, download the dataset from Blackboard and then change the address to the address of your downloaded dataset.

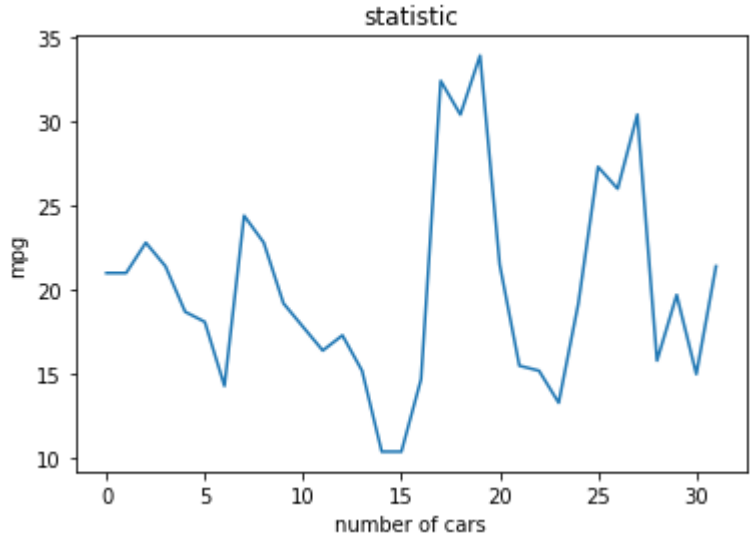
In [5]:
```
cars = pd.read_csv('Downloads/mtcars.csv')
cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec',

mpg = cars['mpg']
```

In [6]:
```
1 mpg.plot()
2
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1a99398bb88>



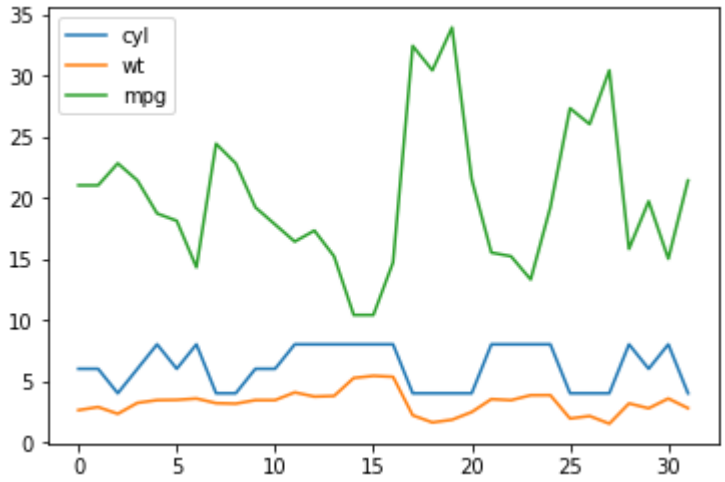
In [7]:
```
1 mpg.plot()
2 plt.xlabel('number of cars')
3 plt.ylabel('mpg')
4 plt.title('statistic')
5 plt.show()
```



Then we want to select 3 variables and compare them in the chart.

In [8]:
```
df = cars[['cyl','wt','mpg']]
df.plot()
```

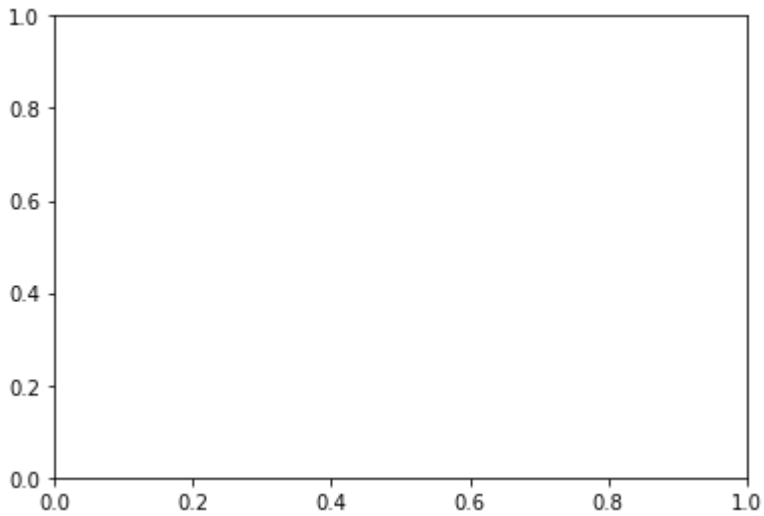
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1a994226e88>



Object Oriented Method

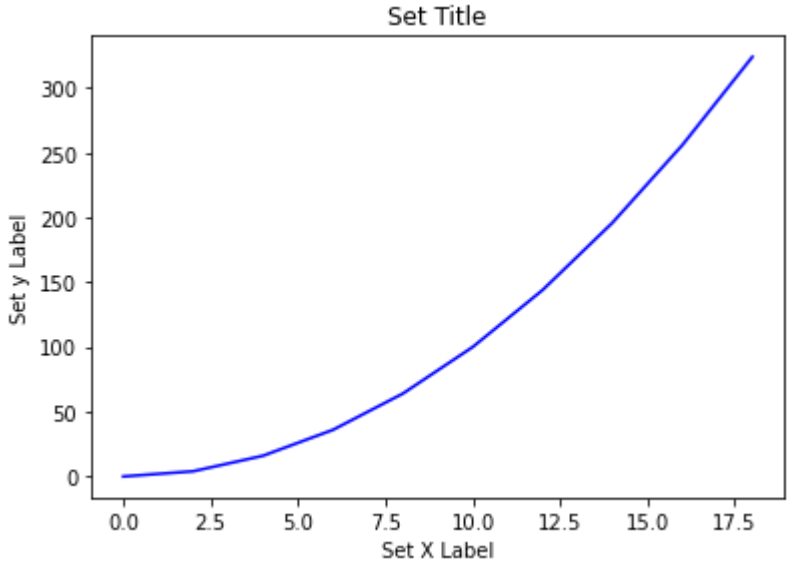
To begin we create a figure instance. Then we can add axes to that figure:

In [12]:
```
# Create Figure (empty canvas)
fig = plt.figure()

# Add set of axes to figure
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # left, bottom, width, height (ran


```



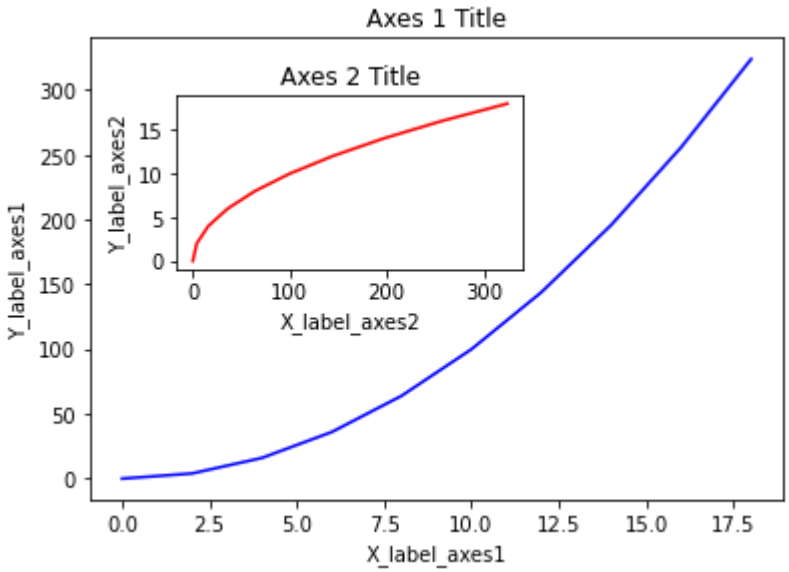
In [13]:
```
# Create Figure (empty canvas)
fig = plt.figure()

# Add set of axes to figure
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # left, bottom, width, height (ran


# Plot on that set of axes
axes.plot(x, y, 'b')
axes.set_xlabel('Set X Label') # Notice the use of set_ to begin methods
axes.set_ylabel('Set y Label')
axes.set_title('Set Title')
```

Out[13]: Text(0.5, 1.0, 'Set Title')



Code is a little more complicated, but the advantage is that we now have full control of where the plot axes are placed, and we can easily add more than one axis to the figure:

In [14]:
```
# Creates blank canvas
fig = plt.figure()

axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3]) # inset axes

# Larger Figure Axes 1
axes1.plot(x, y, 'b')
axes1.set_xlabel('X_label_axes1')
axes1.set_ylabel('Y_label_axes1')
axes1.set_title('Axes 1 Title')

# Insert Figure Axes 2
axes2.plot(y, x, 'r')
axes2.set_xlabel('X_label_axes2')
axes2.set_ylabel('Y_label_axes2')
axes2.set_title('Axes 2 Title');
```



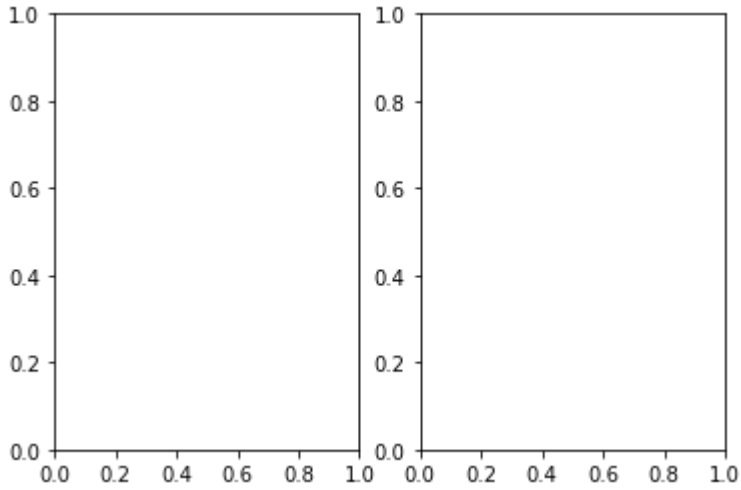
subplots()

Here we will have two subplot , x1 and x2. in this subplot we will have one row and two column. so we will write: ax1.plot(x) and ax2.plot(x,y)

The plt.subplots() object will act as a more automatic axis manager.

Then you can specify the number of rows and columns when creating the subplots() object:

In [15]:
```
# Empty canvas of 1 by 2 subplots
fig, axes = plt.subplots(nrows=1, ncols=2)

```



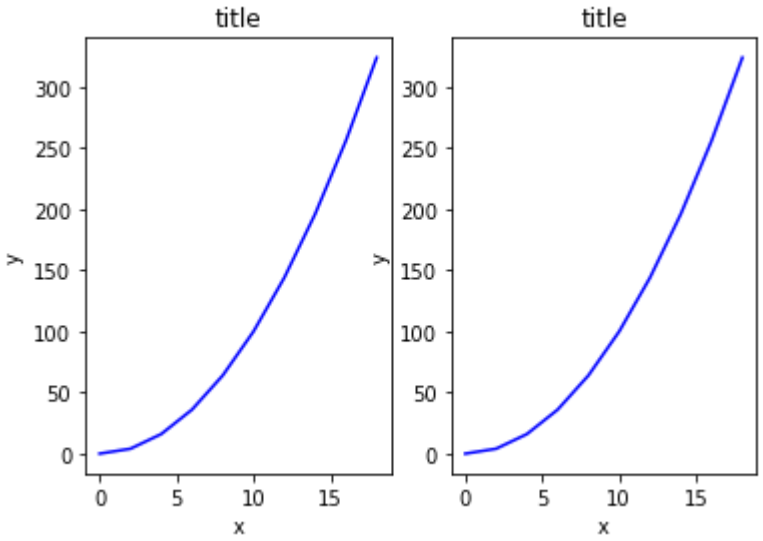
In [16]:
```
# Axes is an array of axes to plot on
axes
```

Out[16]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x000001A995481888>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000001A9954A6308>],
      dtype=object)
```

We can iterate through this array:

In [17]:
```
for ax in axes:
    ax.plot(x, y, 'b')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_title('title')

# Display the figure object
fig
```

Out[17]:



Saving figures

Matplotlib can generate high-quality output in a number formats, including PNG, JPG, EPS, SVG, PGF and PDF.

To save a figure to a file we can use the `savefig` method in the `Figure` class:

In [18]:
```
fig.savefig("filename.png")
```

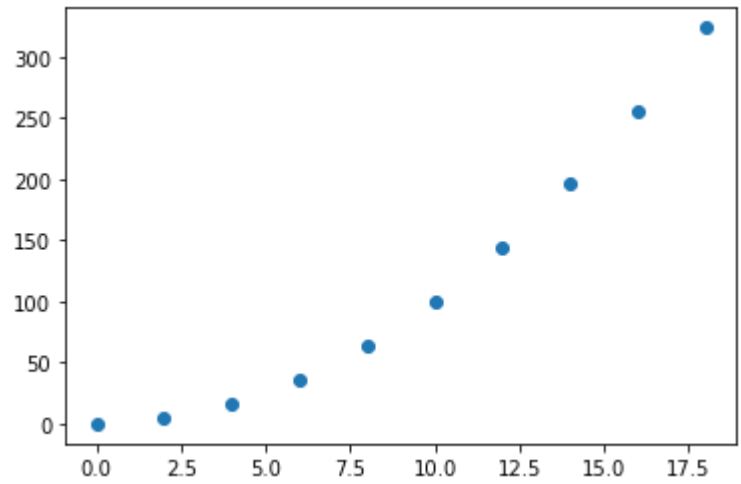
In [19]:
```
%pwd
```

Out[19]:
```
'C:\\Users\\prpou'
```

Special Plot Types

In [20]:
```
1 plt.scatter(x,y)
```

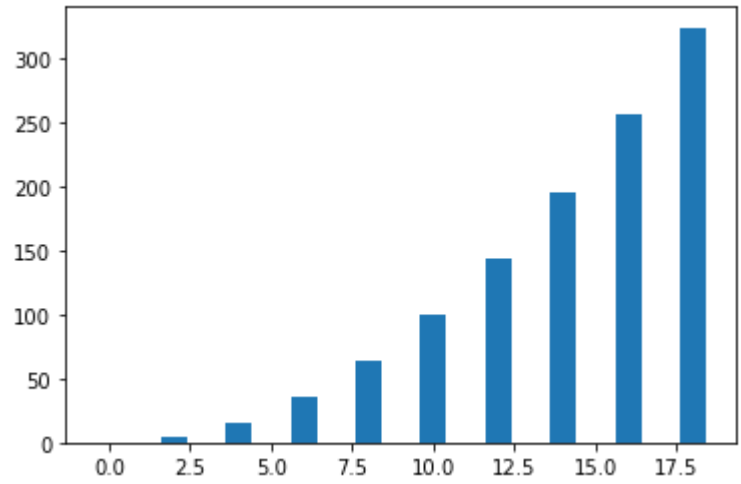
Out[20]: <matplotlib.collections.PathCollection at 0x1a9955802c8>



Creating bar charts

In [21]:
```
1 plt.bar(x,y)
```

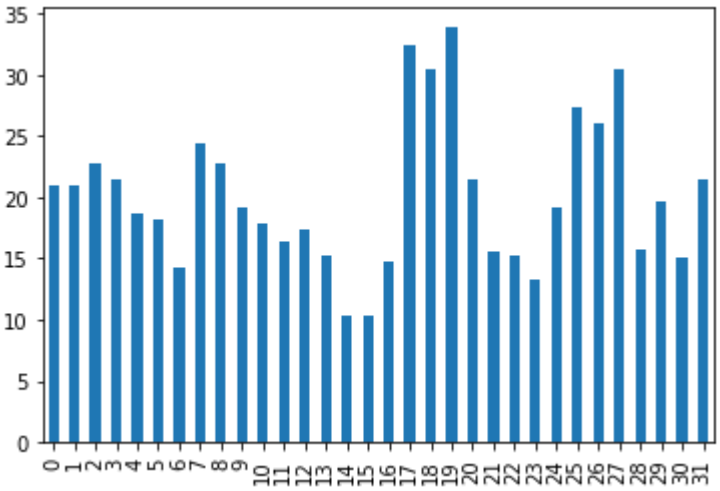
Out[21]: <BarContainer object of 10 artists>



Creating bar charts for dataset

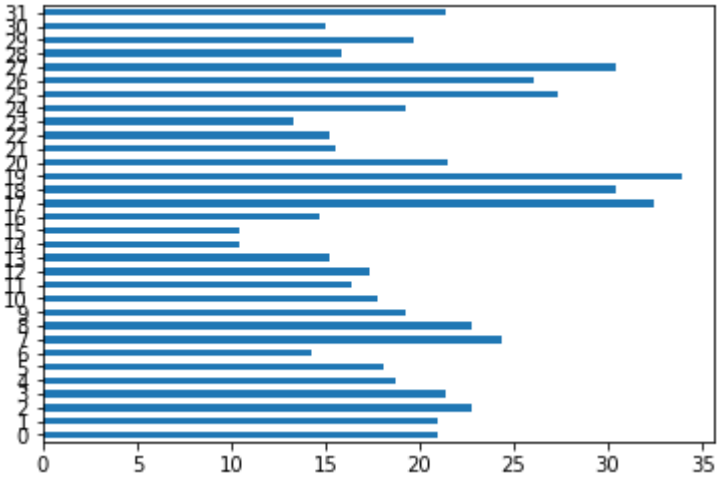
In [85]:
```
1 mpg.plot(kind="bar")
```

Out[85]: <matplotlib.axes._subplots.AxesSubplot at 0x21544dc1e08>



In [62]:
```
1 mpg.plot(kind="barh")
```

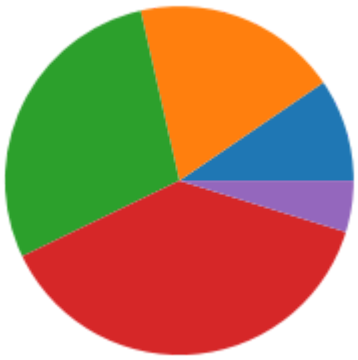
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x21544214d08>



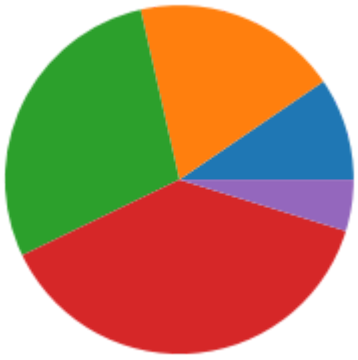
#Question: what is the different of "bar"and "barh"

Creating a pie chart

In [64]:
```
x = [1,2,3,4,0.5]
plt.pie(x)
plt.show()
```



In [65]:
```
plt.pie(x)
plt.savefig('pie_chart.png')
plt.show()
```



In [66]:
```
%pwd
```

Out[66]: 'C:\\Users\\prpou'