ML-5

MATPLOTLIB

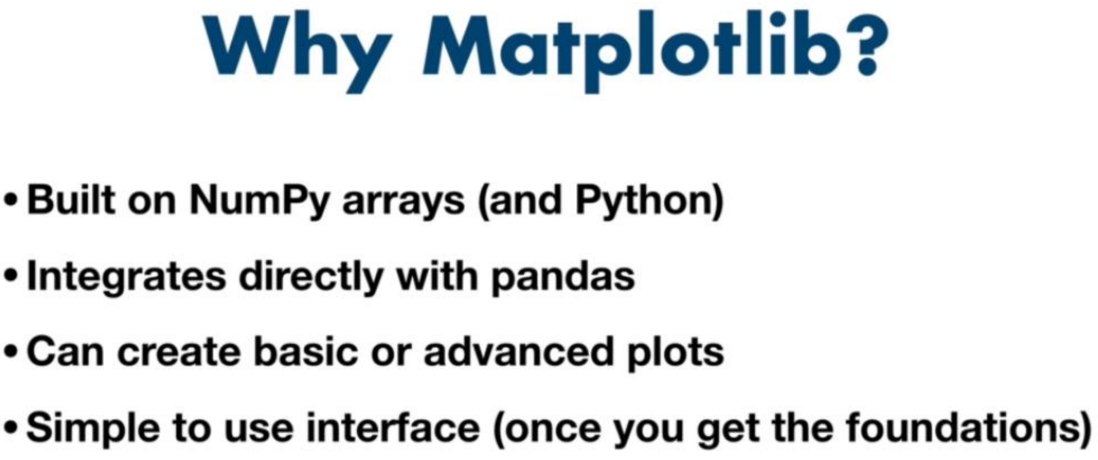
Muhammad Naqeeb | Machine learning / Matplotlib | August 9, 2021

Contents

Matplotlib

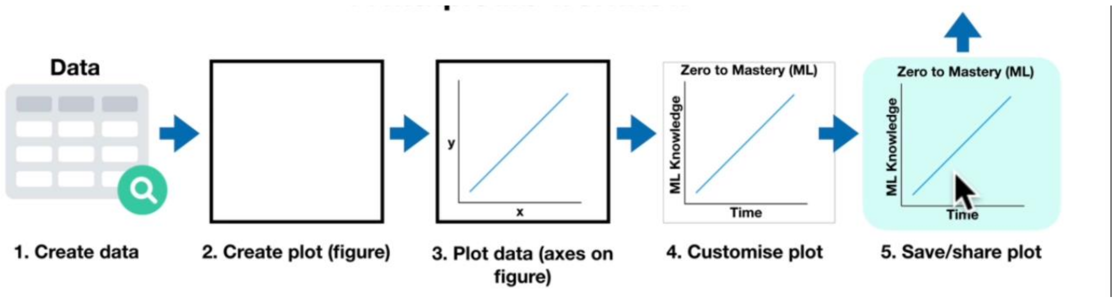
What is Matplotlib?

It's a plotting library more specifically a python plotting library. It allows us to turn our data into some pretty visualizations also known as plots or figures.



Why Matplotlib?

- Built on NumPy arrays (and Python)
- Integrates directly with pandas
- Can create basic or advanced plots
- Simple to use interface (once you get the foundations)



1. Create data
2. Create plot (figure)
3. Plot data (axes on figure)
4. Customise plot
5. Save/share plot

Humans were visual creatures. We want to see things visually. So that's where these plots come in handy, so you can save or share them to visually communicate your work rather than just having it in a table full of numbers

Import

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

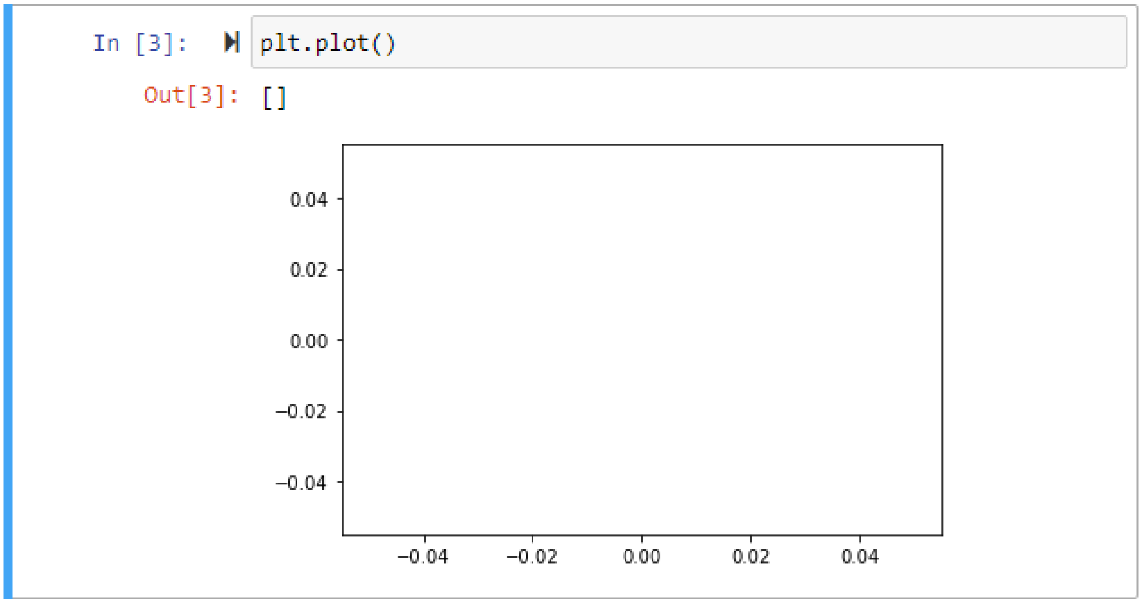
```
import pandas as pd
import numpy as np
```

Creating Plot

plt.plot()

Plot y versus x as lines and/or markers.

```
plt.plot()
```

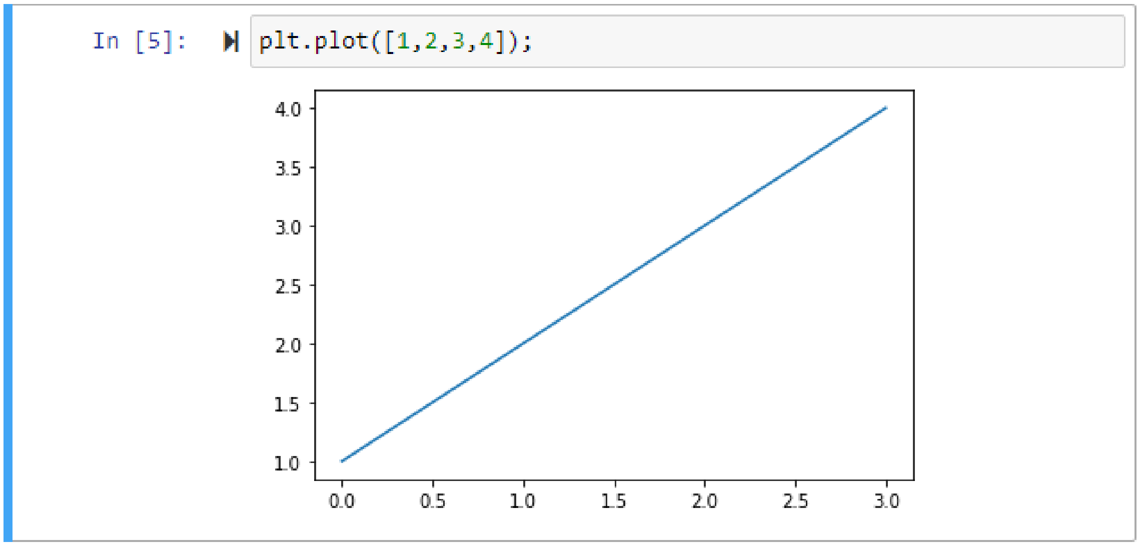


```
In [3]:  plt.plot()
Out[3]:  []
```

Same thing but in this we don’t have those square brackets

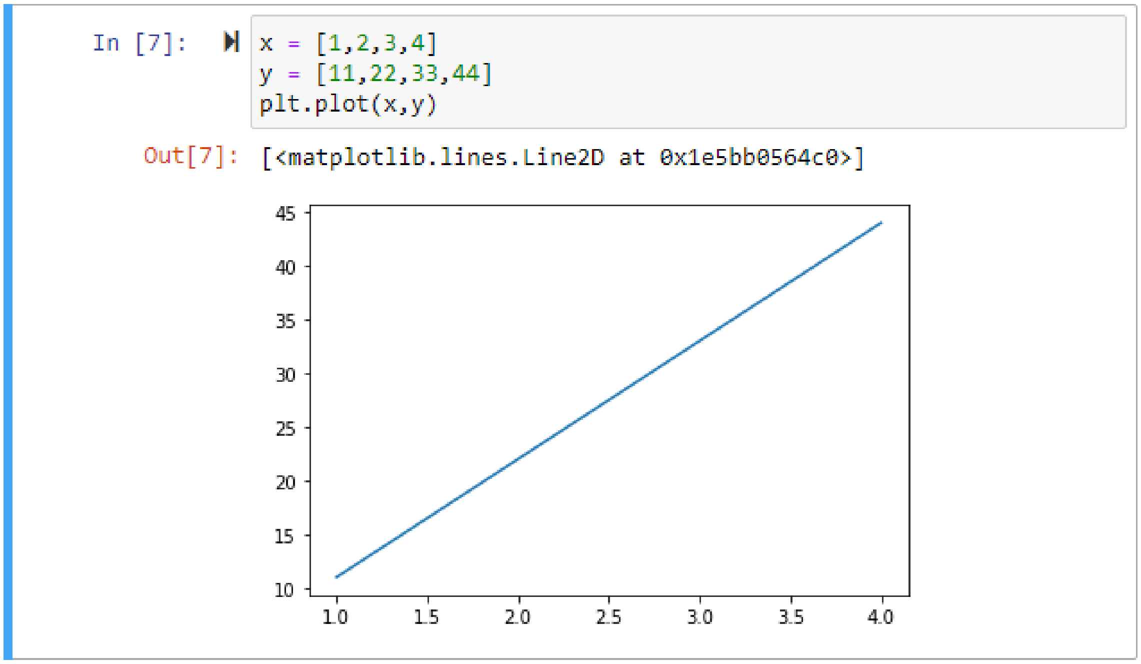
```
plt.plot();
```

```
plt.plot([1,2,3,4]);
```



```
In [5]:  plt.plot([1,2,3,4]);
```

```
x = [1,2,3,4]
y = [11,22,33,44]
plt.plot(x,y)
```



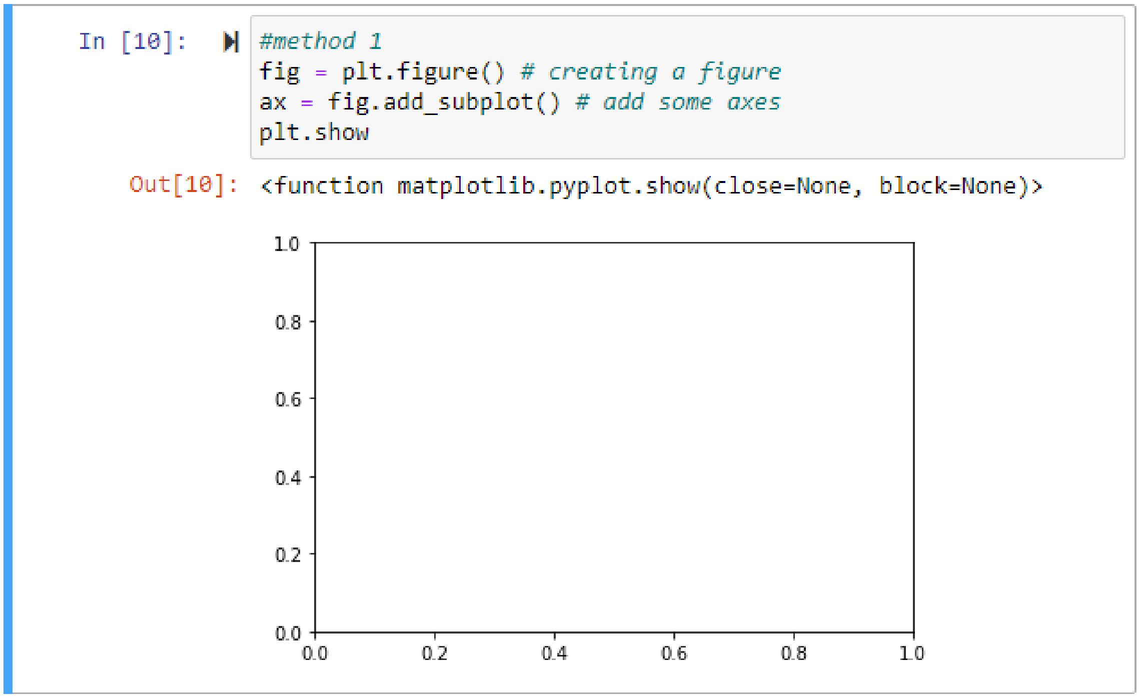
```
In [7]:  x = [1,2,3,4]
         y = [11,22,33,44]
         plt.plot(x,y)

Out[7]:  [<matplotlib.lines.Line2D at 0x1e5bb0564c0>]
```

Method of plotting

```
x = [1,2,3,4]
y = [11,22,33,44]
```

Method 1

```
#method 1
fig = plt.figure() # creating a figure
ax = fig.add_subplot() # add some axes
plt.show
```

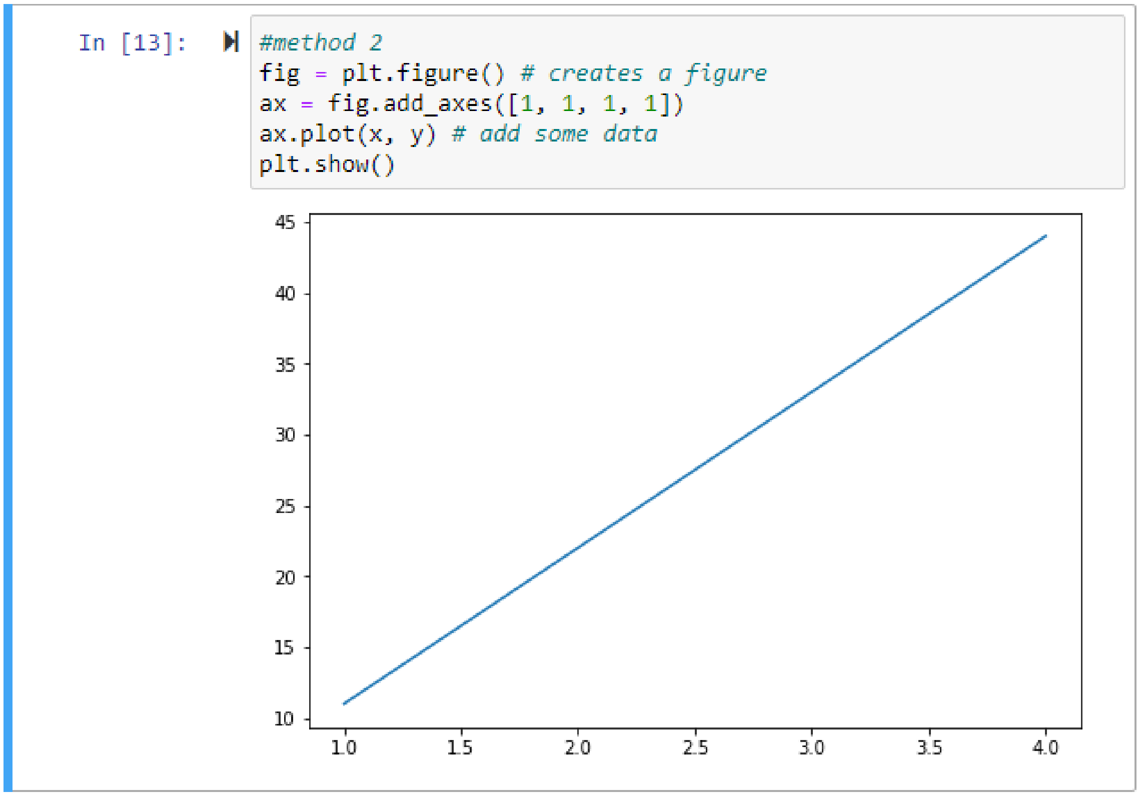


```
In [10]:  #method 1
          fig = plt.figure() # creating a figure
          ax = fig.add_subplot() # add some axes
          plt.show
```

```
Out[10]: <function matplotlib.pyplot.show(close=None, block=None)>
```

Method 2

```
#method 2
fig = plt.figure() # creates a figure
ax = fig.add_axes([1, 1, 1, 1])
ax.plot(x, y) # add some data
plt.show()
```

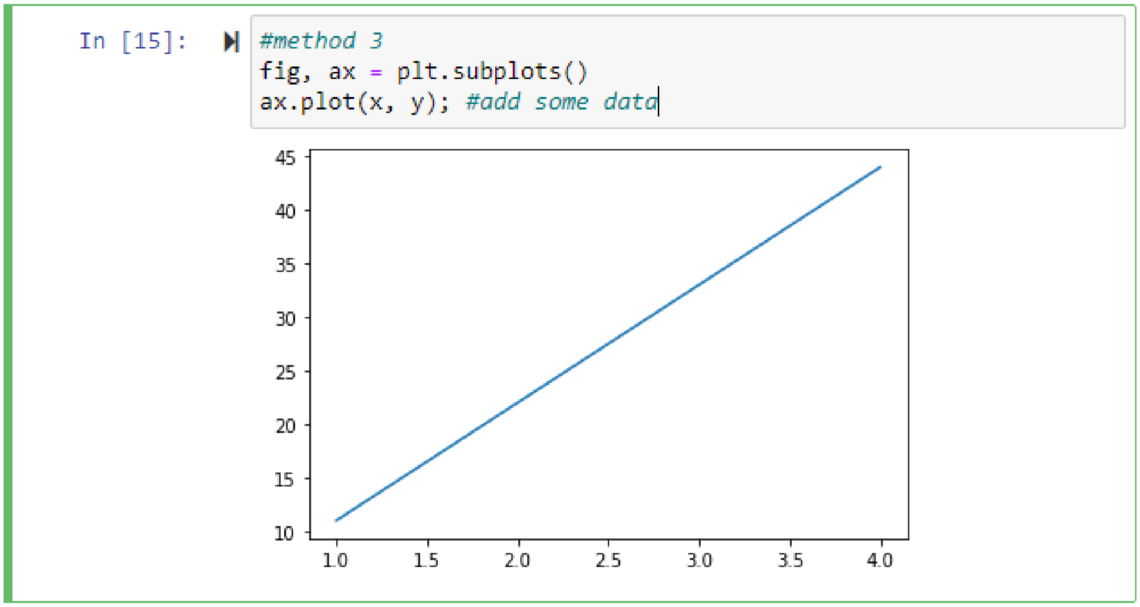


In [13]:

```
#method 2
fig = plt.figure() # creates a figure
ax = fig.add_axes([1, 1, 1, 1])
ax.plot(x, y) # add some data
plt.show()
```

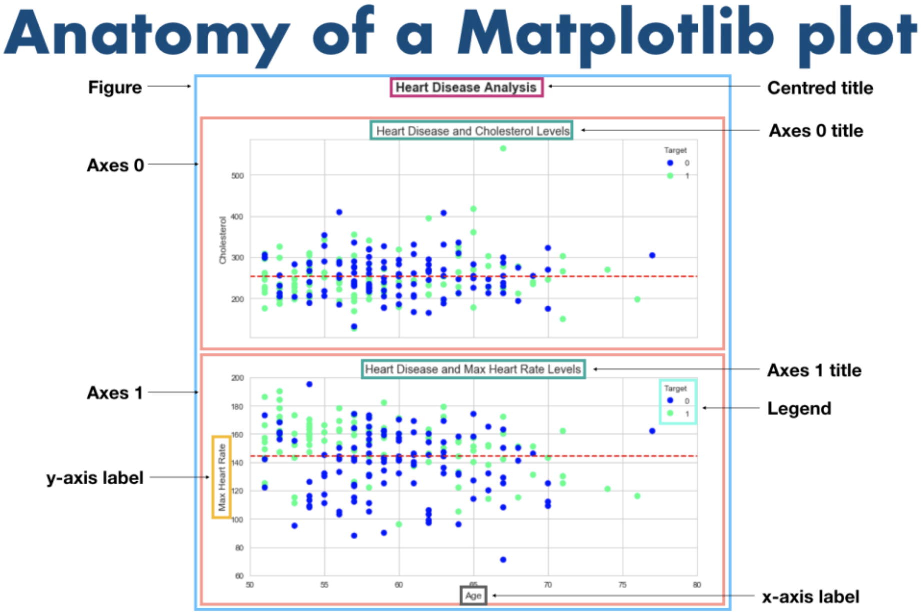
Method 3 (Recommended)

```
#method 3
fig, ax = plt.subplots()
ax.plot(x, y); #add some data
```



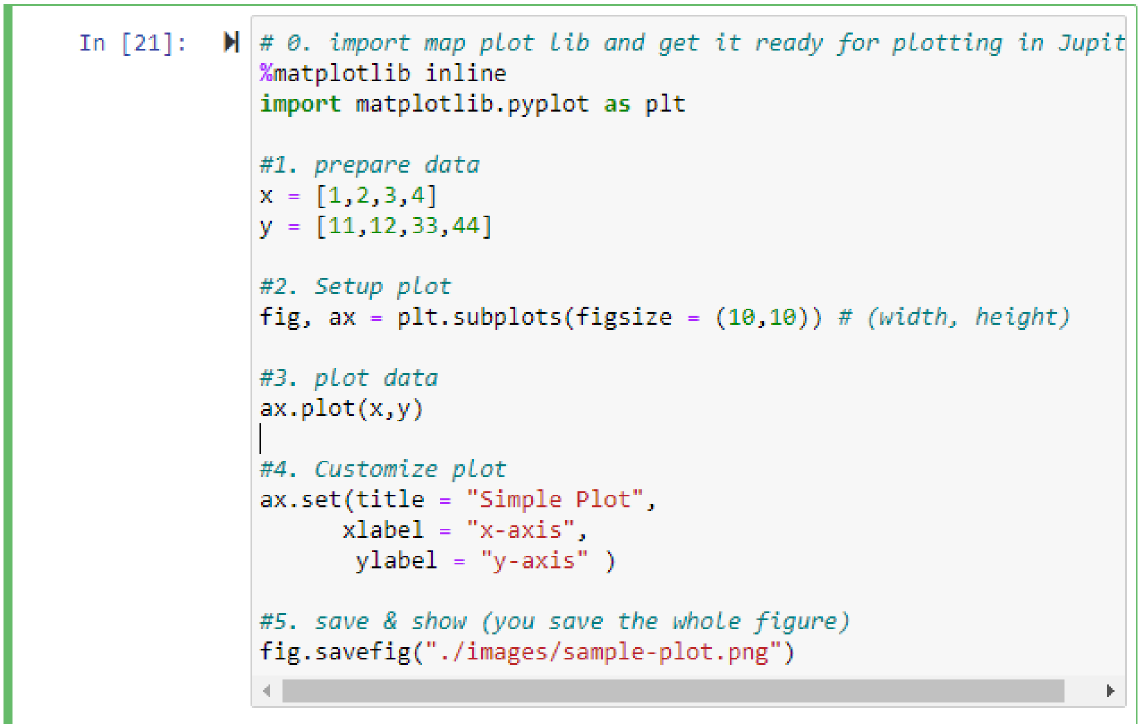
```
In [15]:  #method 3
          fig, ax = plt.subplots()
          ax.plot(x, y); #add some data
```

Anatomy Of A Matplotlib Figure

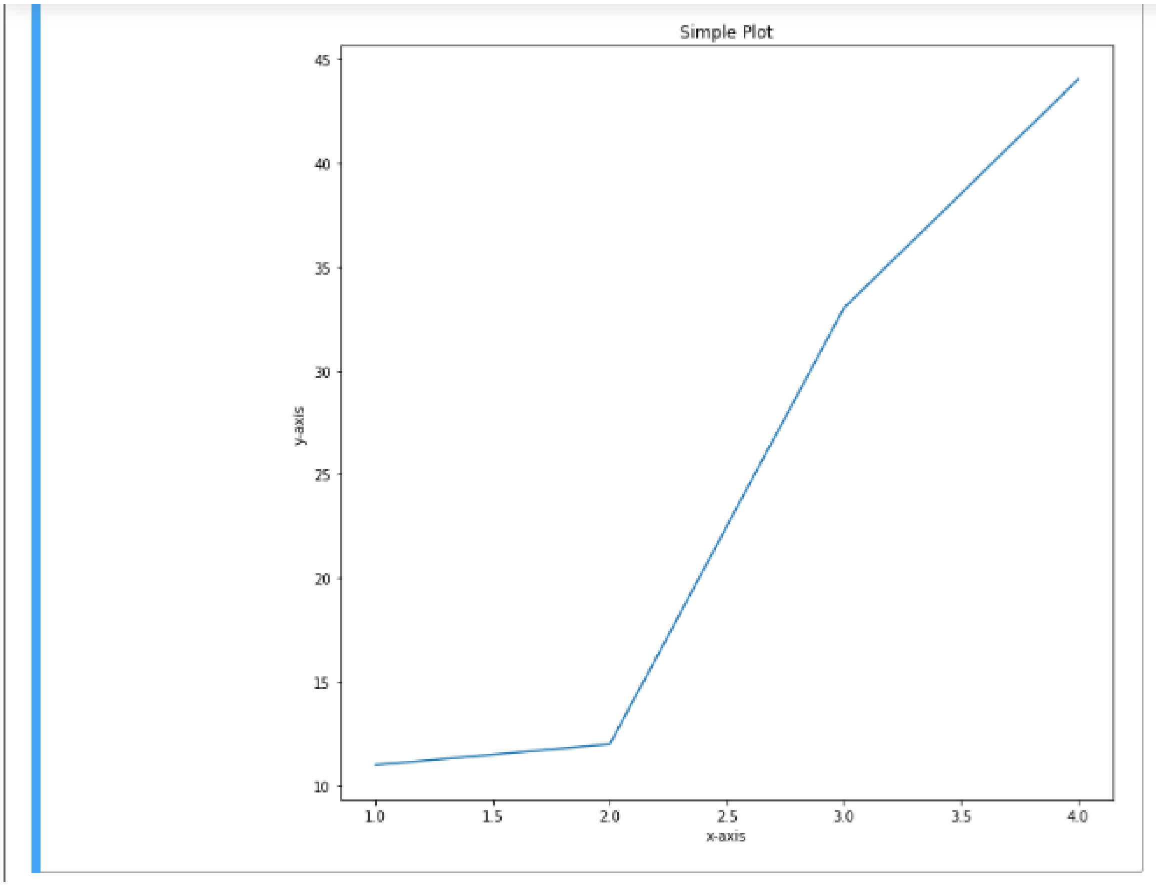


Matplotlib example workflow

```
# 0. import map plot lib and get it ready for plotting in Jupiter
%matplotlib inline
import matplotlib.pyplot as plt

#1. prepare data
x = [1,2,3,4]
y = [11,12,33,44]

#2. Setup plot
fig, ax = plt.subplots(figsize = (10,10)) # (width, height)

#3. plot data
ax.plot(x,y)

#4. Customize plot
ax.set(title = "Simple Plot",
      xlabel = "x-axis",
       ylabel = "y-axis" )

#5. save & show (you save the whole figure)
fig.savefig("./images/sample-plot.png")
```



```
# 0. import map plot lib and get it ready for plotting in Jupit
%matplotlib inline
import matplotlib.pyplot as plt

#1. prepare data
x = [1,2,3,4]
y = [11,12,33,44]

#2. Setup plot
fig, ax = plt.subplots(figsize = (10,10)) # (width, height)

#3. plot data
ax.plot(x,y)

#4. Customize plot
ax.set(title = "Simple Plot",
      xlabel = "x-axis",
       ylabel = "y-axis" )

#5. save & show (you save the whole figure)
fig.savefig("./images/sample-plot.png")
```

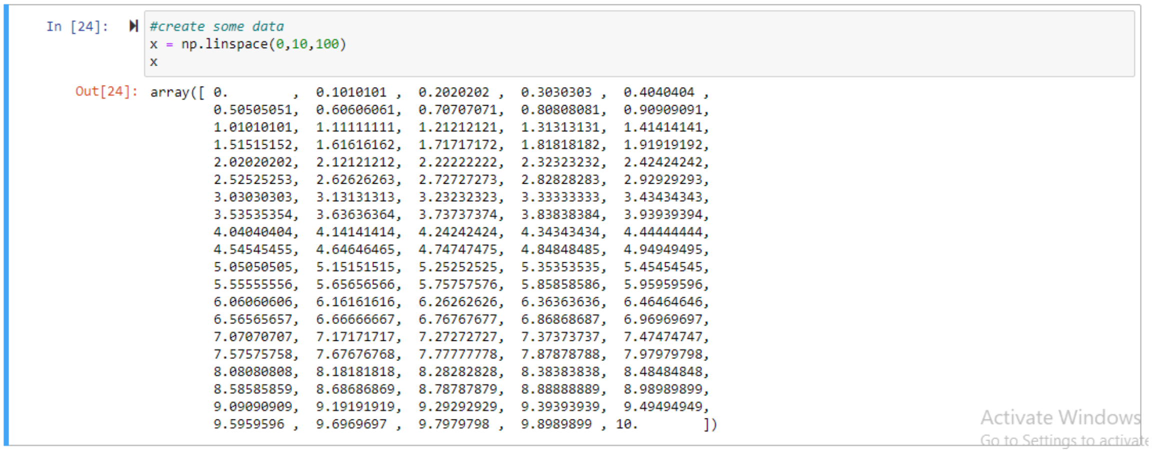


Making figures with NUMPY arrays / type of figures

np.linespace()

The numpy.linspace() function returns number spaces evenly w.r.t interval.

```
numpy.linspace(start, stop, num, endpoint, retstep, dtype)
```



```
In [24]:  #create some data
          x = np.linspace(0,10,100)
          x

Out[24]: array([ 0.        ,  0.1010101 ,  0.2020202 ,  0.3030303 ,  0.4040404 ,
                 0.50505051,  0.60606061,  0.70707071,  0.80808081,  0.90909091,
                 1.01010101,  1.11111111,  1.21212121,  1.31313131,  1.41414141,
                 1.51515152,  1.61616162,  1.71717172,  1.81818182,  1.91919192,
                 2.02020202,  2.12121212,  2.22222222,  2.32323232,  2.42424242,
                 2.52525253,  2.62626263,  2.72727273,  2.82828283,  2.92929293,
                 3.03030303,  3.13131313,  3.23232323,  3.33333333,  3.43434343,
                 3.53535354,  3.63636364,  3.73737374,  3.83838384,  3.93939394,
                 4.04040404,  4.14141414,  4.24242424,  4.34343434,  4.44444444,
                 4.54545455,  4.64646465,  4.74747475,  4.84848485,  4.94949495,
                 5.05050505,  5.15151515,  5.25252525,  5.35353535,  5.45454545,
                 5.55555556,  5.65656566,  5.75757576,  5.85858586,  5.95959596,
                 6.06060606,  6.16161616,  6.26262626,  6.36363636,  6.46464646,
                 6.56565657,  6.66666667,  6.76767677,  6.86868687,  6.96969697,
                 7.07070707,  7.17171717,  7.27272727,  7.37373737,  7.47474747,
                 7.57575758,  7.67676768,  7.77777778,  7.87878788,  7.97979798,
                 8.08080808,  8.18181818,  8.28282828,  8.38383838,  8.48484848,
                 8.58585859,  8.68686869,  8.78787879,  8.88888889,  8.98989899,
                 9.09090909,  9.19191919,  9.29292929,  9.39393939,  9.49494949,
                 9.5959596 ,  9.6969697 ,  9.7979798 ,  9.8989899 , 10.        ])
```

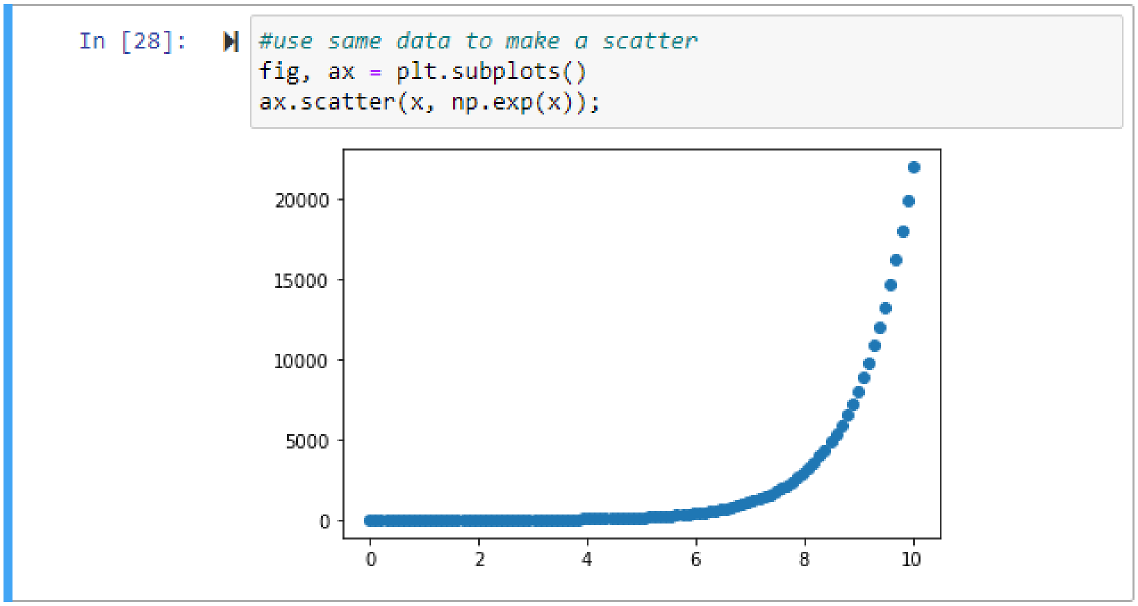
Scatter Plot

Scatter plots are used to observe relationship between variables and uses dots to represent the relationship between them. The scatter () method in the matplotlib library is used to draw a scatter plot.

The scatter() function plots one dot for each observation.

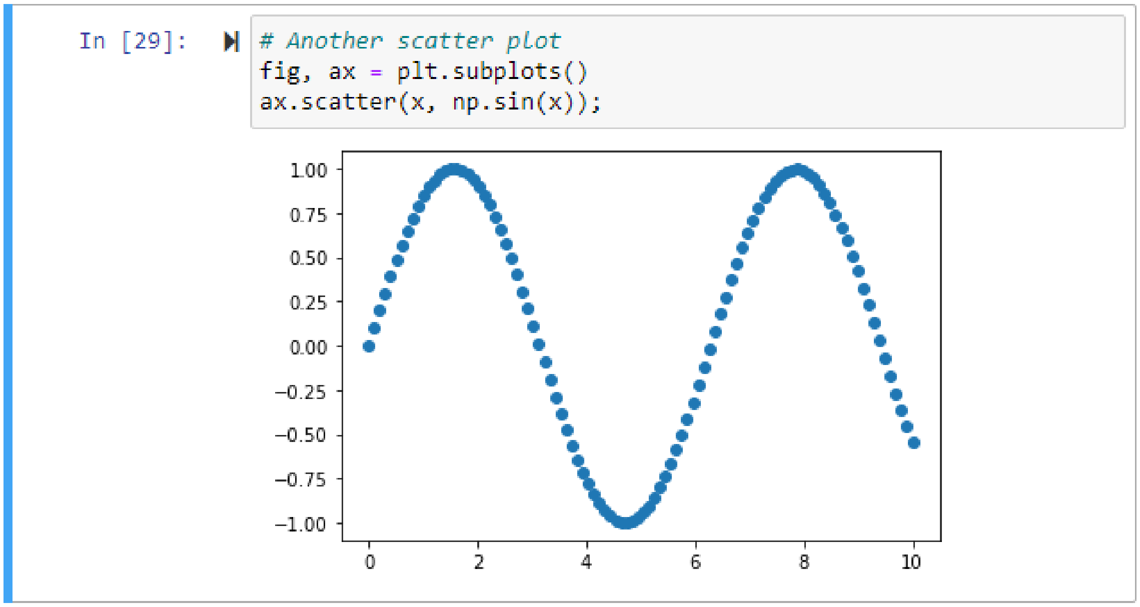
It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

```
#use same data to make a scatter
fig, ax = plt.subplots()
ax.scatter(x, np.exp(x));
```



```
In [28]:  #use same data to make a scatter
          fig, ax = plt.subplots()
          ax.scatter(x, np.exp(x));
```

```
# Another scatter plot
fig, ax = plt.subplots()
ax.scatter(x, np.sin(x));
```



```
In [29]:  # Another scatter plot
          fig, ax = plt.subplots()
          ax.scatter(x, np.sin(x));
```

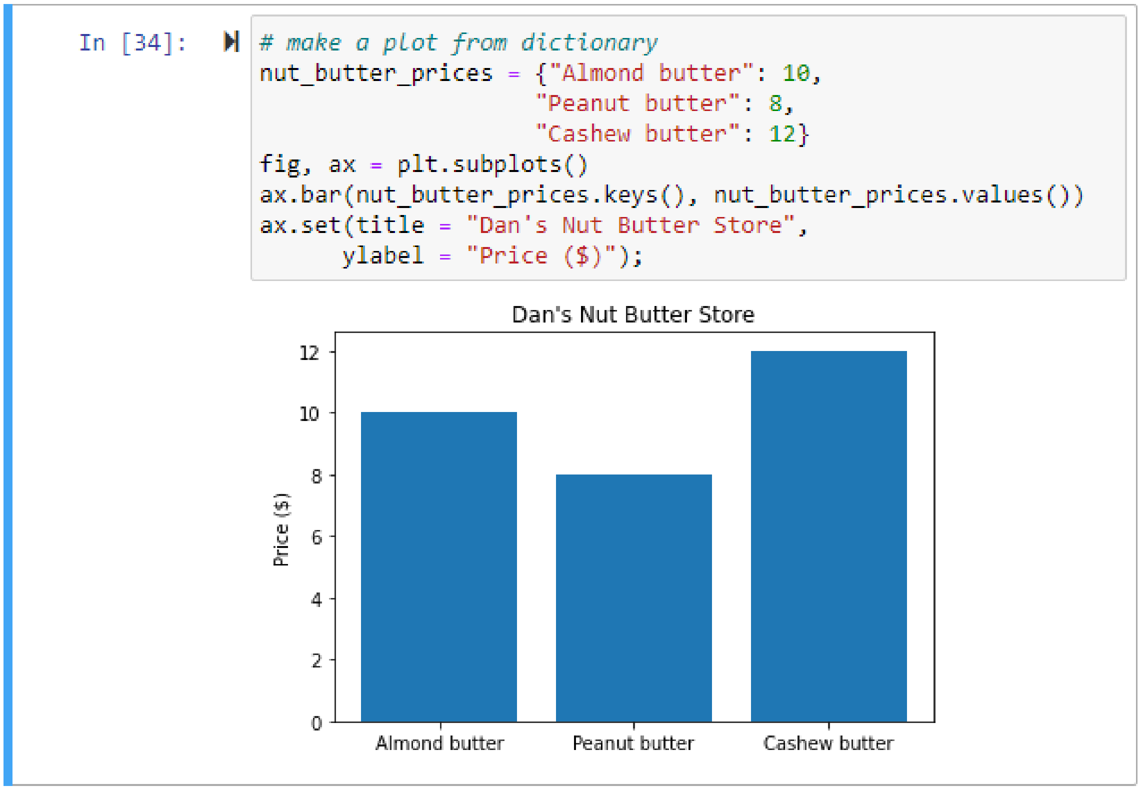
Bar plot

The matplotlib API in Python provides the bar() function which can be used in MATLAB style use or as an object-oriented API.

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories.

```
plt.bar(x, height, width, bottom, align)
```

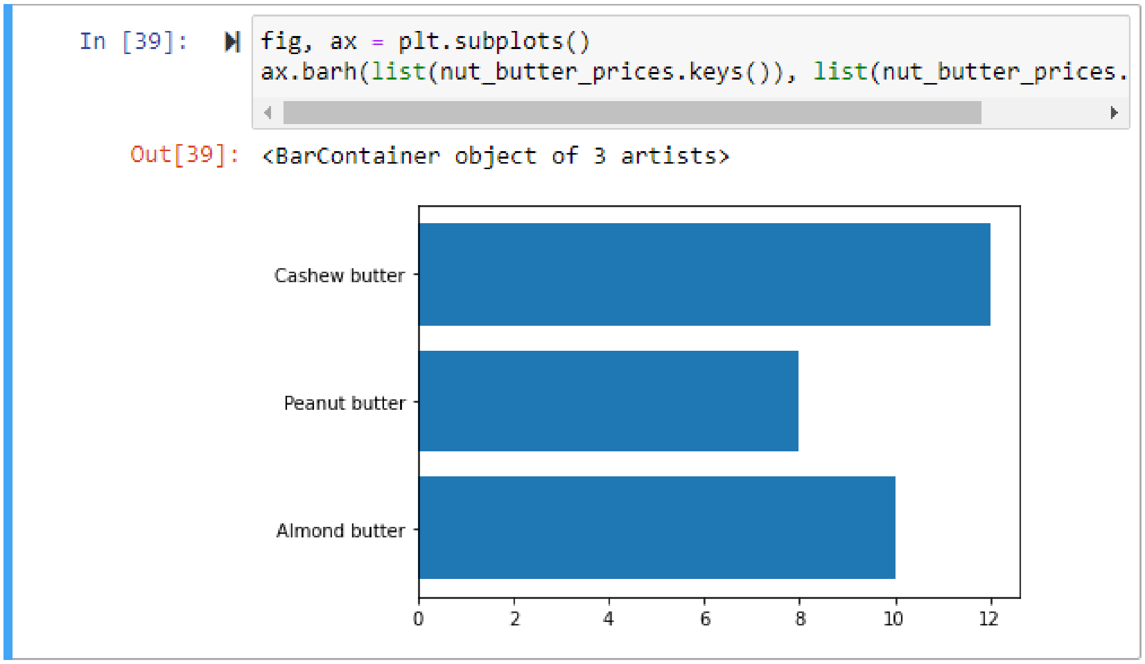
```
# make a plot from dictionary
nut_butter_prices = {"Almond butter": 10,
                     "Peanut butter": 8,
                     "Cashew butter": 12}
fig, ax = plt.subplots()
ax.bar(nut_butter_prices.keys(), nut_butter_prices.values())
ax.set(title = "Dan's Nut Butter Store",
       ylabel = "Price ($)");
```



```
In [34]:  # make a plot from dictionary
          nut_butter_prices = {"Almond butter": 10,
                               "Peanut butter": 8,
                               "Cashew butter": 12}
          fig, ax = plt.subplots()
          ax.bar(nut_butter_prices.keys(), nut_butter_prices.values())
          ax.set(title = "Dan's Nut Butter Store",
                ylabel = "Price ($)");
```

Horizontal bar plot

```
fig, ax = plt.subplots()
ax.barh(list(nut_butter_prices.keys()),
list(nut_butter_prices.values()))
```



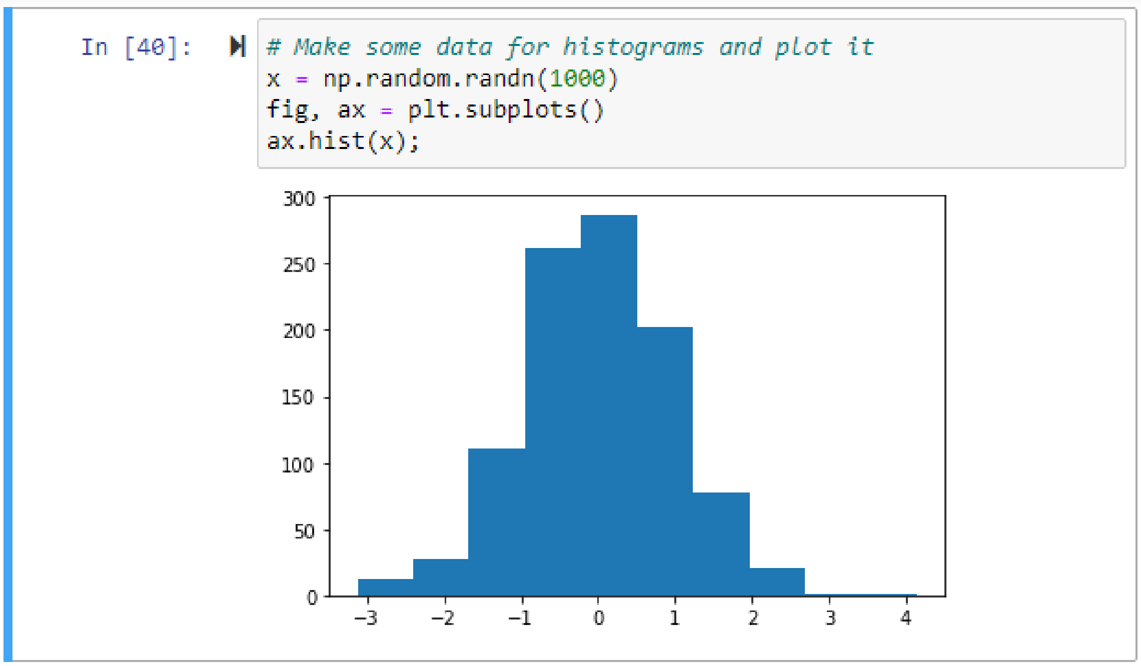
```
In [39]:  fig, ax = plt.subplots()
           ax.barh(list(nut_butter_prices.keys()), list(nut_butter_prices.

Out[39]: <BarContainer object of 3 artists>
```

Histogram / .hist()

Matplotlib histogram is used to visualize the frequency distribution of numeric array by splitting it to small equal-sized bins.

A histogram is basically used to represent data provided in a form of some groups.It is accurate method for the graphical representation of numerical data distribution.It is a type of bar plot where X-axis represents the bin ranges while Y-axis gives information about frequency.

```
# Make some data for histograms and plot it
x = np.random.randn(1000)
fig, ax = plt.subplots()
ax.hist(x);
```



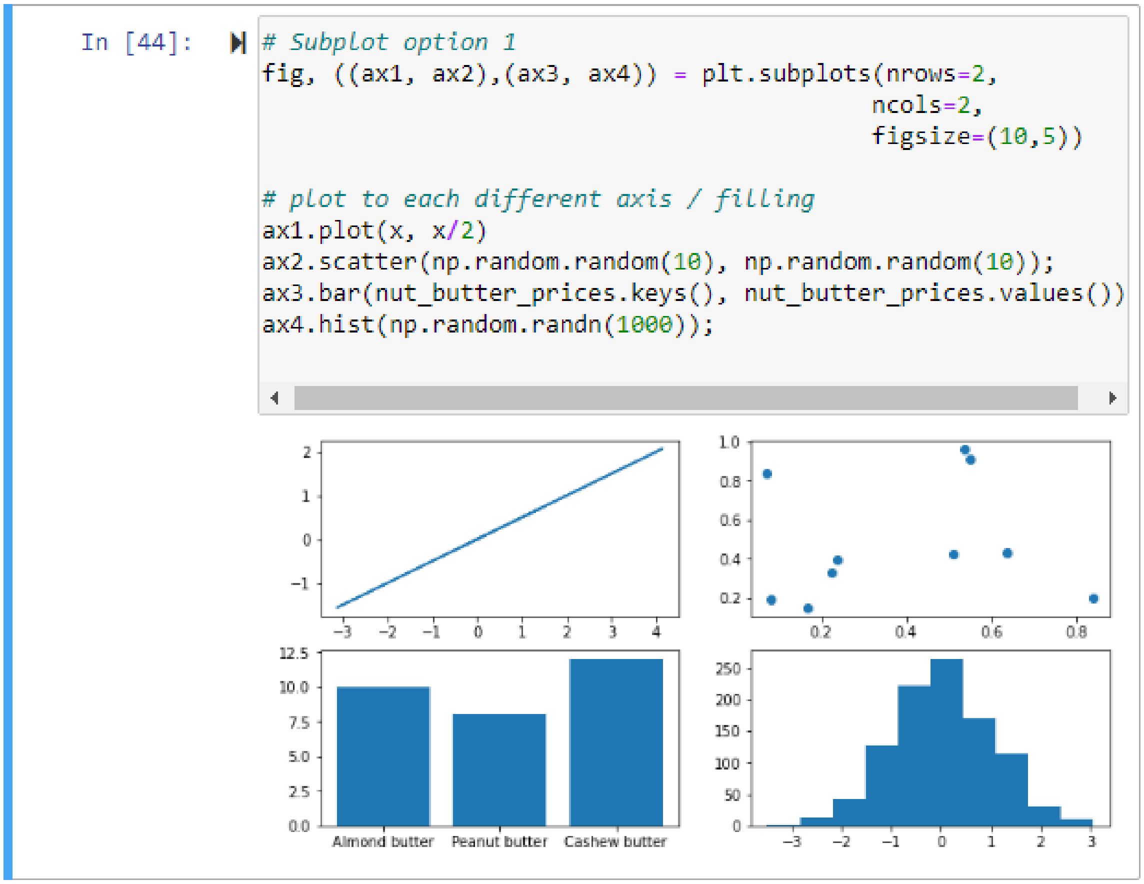
```
In [40]:  # Make some data for histograms and plot it
          x = np.random.randn(1000)
          fig, ax = plt.subplots()
          ax.hist(x);
```

Multiple plot with same command

There are two options for subplots.

Option 1

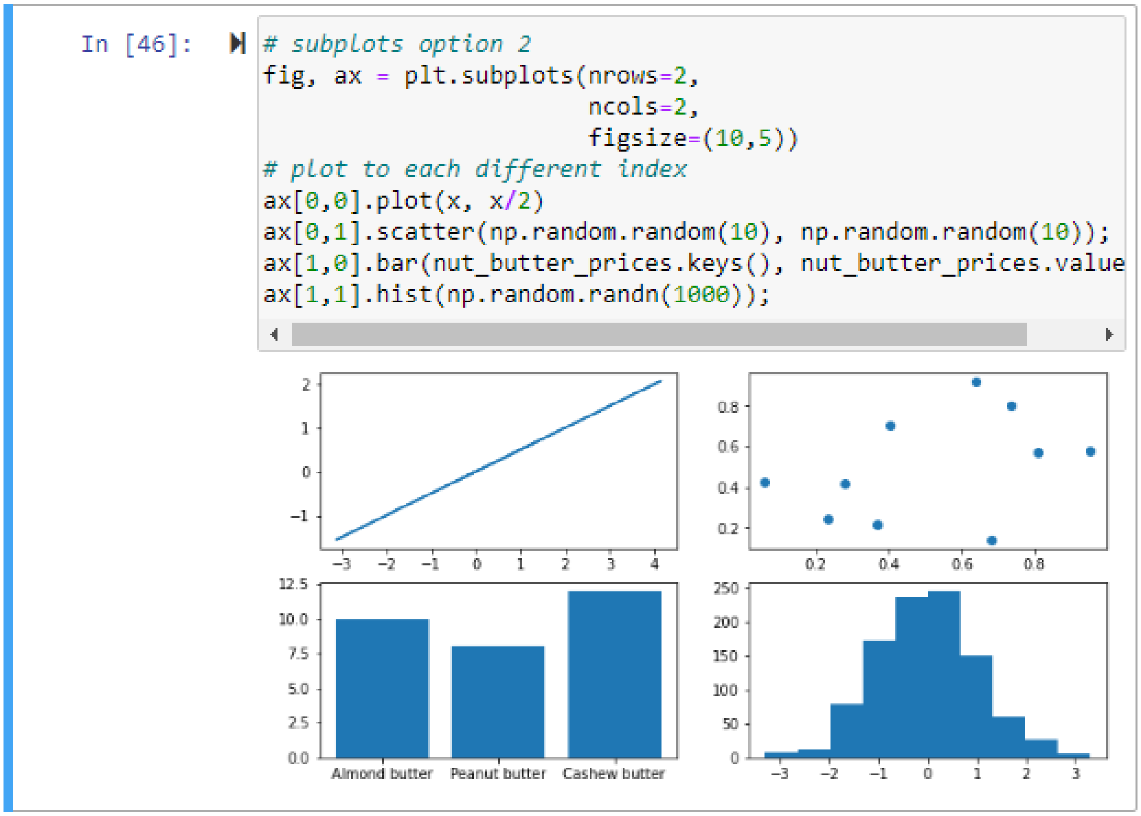
```
# Subplot option 1
fig, ((ax1, ax2),(ax3, ax4)) = plt.subplots(nrows=2,
                                            ncols=2,
                                            figsize=(10,5))

# plot to each different axis / filling
ax1.plot(x, x/2)
ax2.scatter(np.random.random(10), np.random.random(10));
ax3.bar(nut_butter_prices.keys(), nut_butter_prices.values());
ax4.hist(np.random.randn(1000));
```



```
# Subplot option 1
fig, ((ax1, ax2),(ax3, ax4)) = plt.subplots(nrows=2,
                                            ncols=2,
                                            figsize=(10,5))

# plot to each different axis / filling
ax1.plot(x, x/2)
ax2.scatter(np.random.random(10), np.random.random(10));
ax3.bar(nut_butter_prices.keys(), nut_butter_prices.values())
ax4.hist(np.random.randn(1000));
```

Option 2

```
# subplots option 2
fig, ax = plt.subplots(nrows=2,
                       ncols=2,
                       figsize=(10,5))
# plot to each different index
ax[0,0].plot(x, x/2)
ax[0,1].scatter(np.random.random(10), np.random.random(10));
ax[1,0].bar(nut_butter_prices.keys(), nut_butter_prices.values());
ax[1,1].hist(np.random.randn(1000));
```



```
In [46]:  # subplots option 2
          fig, ax = plt.subplots(nrows=2,
                                 ncols=2,
                                 figsize=(10,5))
          # plot to each different index
          ax[0,0].plot(x, x/2)
          ax[0,1].scatter(np.random.random(10), np.random.random(10));
          ax[1,0].bar(nut_butter_prices.keys(), nut_butter_prices.value
          ax[1,1].hist(np.random.randn(1000));
```

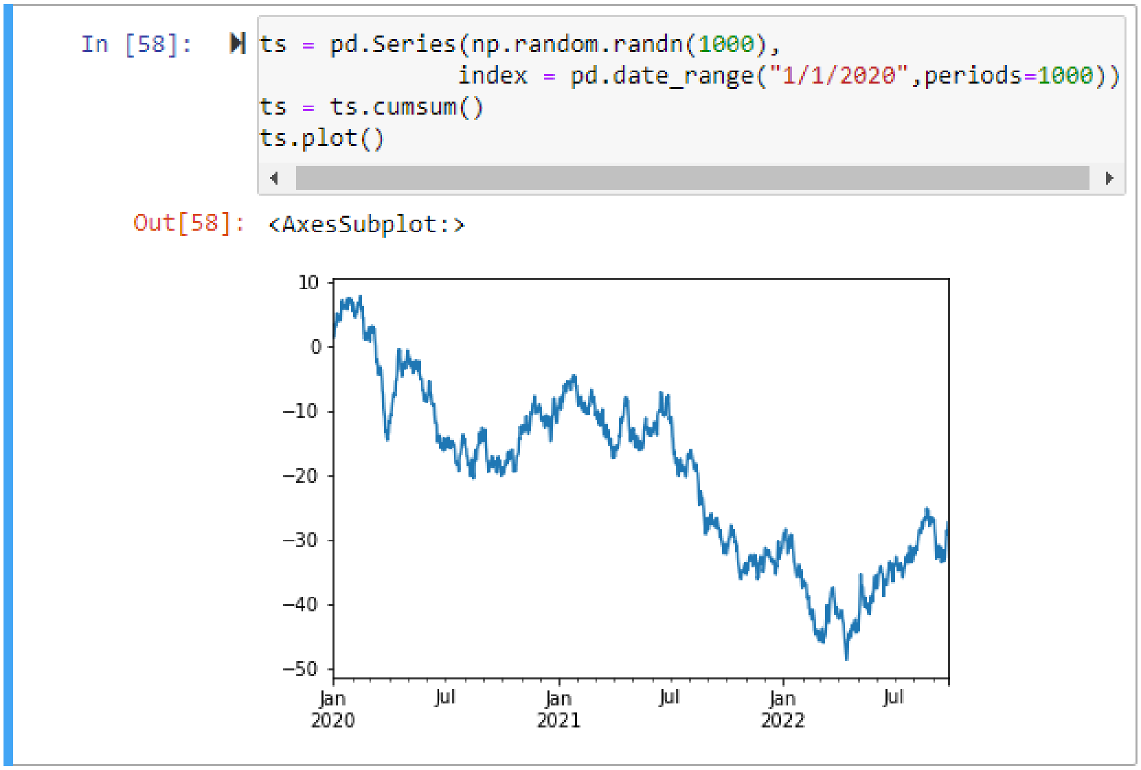
Plotting From Pandas DataFrames

.cumsum()

Return cumulative sum over a DataFrame or Series axis.

```
import pandas as pd
```

```
ts = pd.Series(np.random.randn(1000),
               index = pd.date_range("1/1/2020",periods=1000))
ts = ts.cumsum()
ts.plot()
```



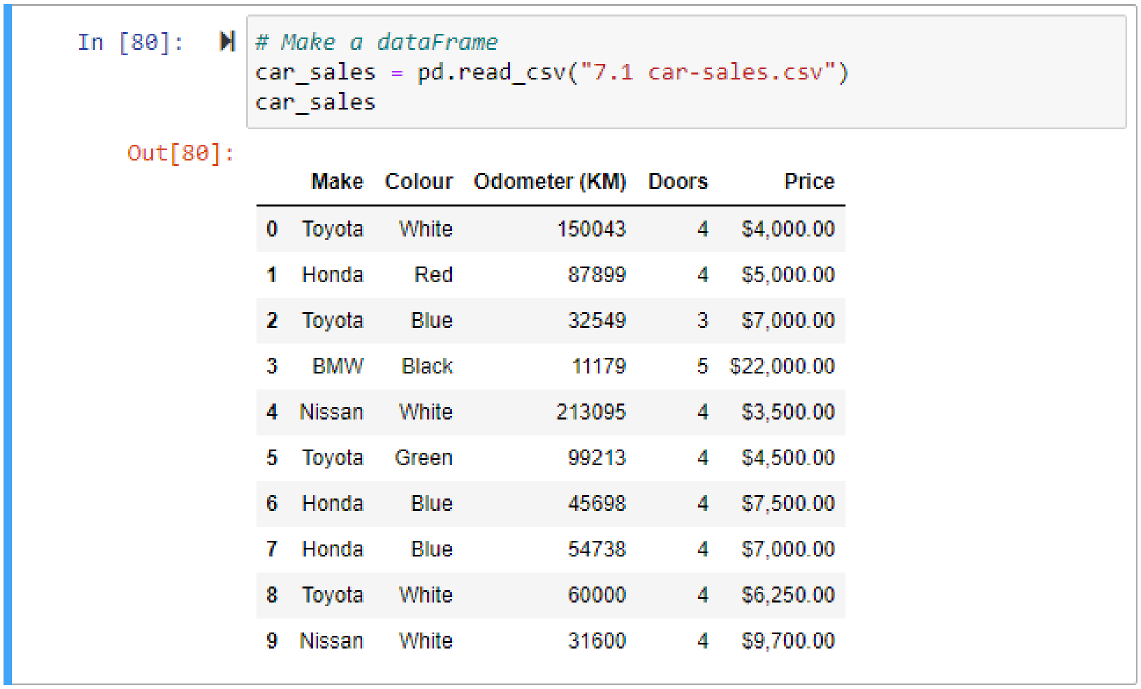
In [58]:

```
ts = pd.Series(np.random.randn(1000),
               index = pd.date_range("1/1/2020",periods=1000))
ts = ts.cumsum()
ts.plot()
```

Out[58]:

```
<AxesSubplot:>
```

Car sales problems

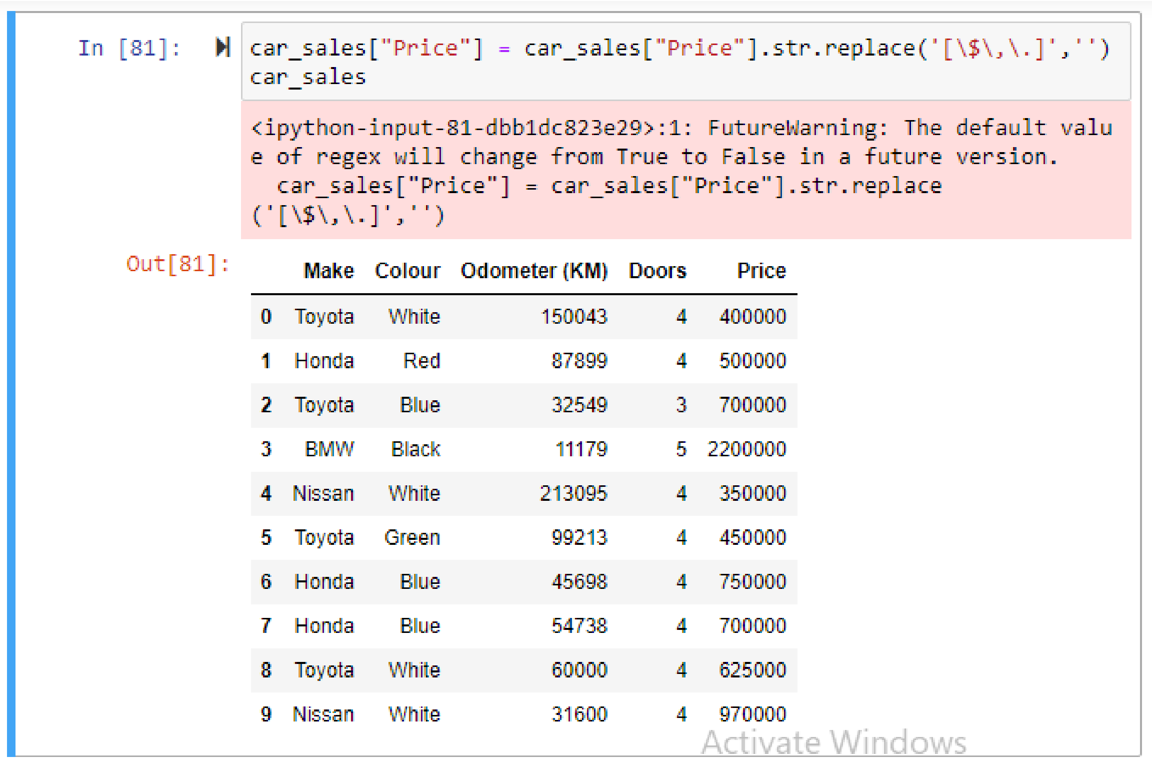


In [80]:

```
# Make a dataFrame
car_sales = pd.read_csv("7.1 car-sales.csv")
car_sales
```

Out[80]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00



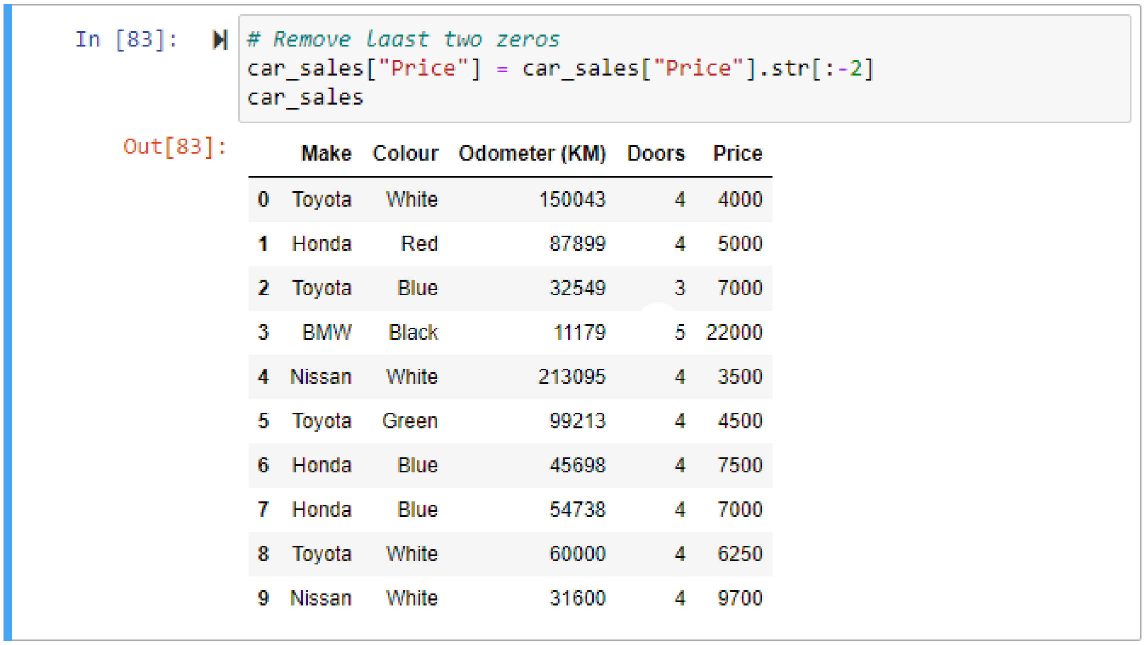
In [81]:

```
car_sales["Price"] = car_sales["Price"].str.replace('[\$\,\.]','')
car_sales
```

```
<ipython-input-81-dbb1dc823e29>:1: FutureWarning: The default valu
e of regex will change from True to False in a future version.
  car_sales["Price"] = car_sales["Price"].str.replace
('[\$\,\.]','')
```

Out[81]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	400000
1	Honda	Red	87899	4	500000
2	Toyota	Blue	32549	3	700000
3	BMW	Black	11179	5	2200000
4	Nissan	White	213095	4	350000
5	Toyota	Green	99213	4	450000
6	Honda	Blue	45698	4	750000
7	Honda	Blue	54738	4	700000
8	Toyota	White	60000	4	625000
9	Nissan	White	31600	4	970000

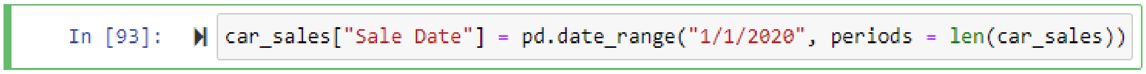


In [83]:

```
# Remove laast two zeros
car_sales["Price"] = car_sales["Price"].str[:-2]
car_sales
```

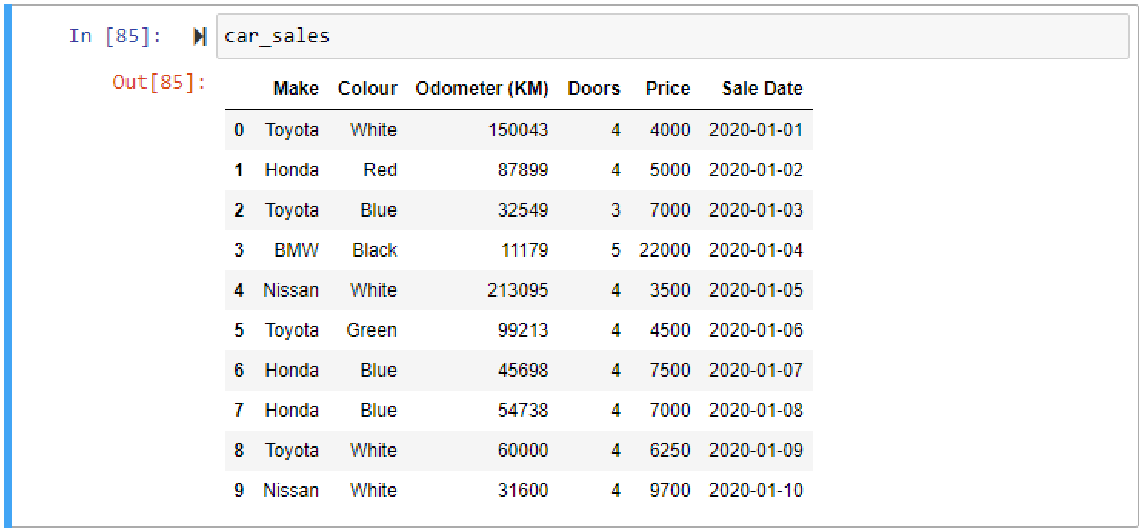
Out[83]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	4000
1	Honda	Red	87899	4	5000
2	Toyota	Blue	32549	3	7000
3	BMW	Black	11179	5	22000
4	Nissan	White	213095	4	3500
5	Toyota	Green	99213	4	4500
6	Honda	Blue	45698	4	7500
7	Honda	Blue	54738	4	7000
8	Toyota	White	60000	4	6250
9	Nissan	White	31600	4	9700



In [93]:

```
car_sales["Sale Date"] = pd.date_range("1/1/2020", periods = len(car_sales))
```

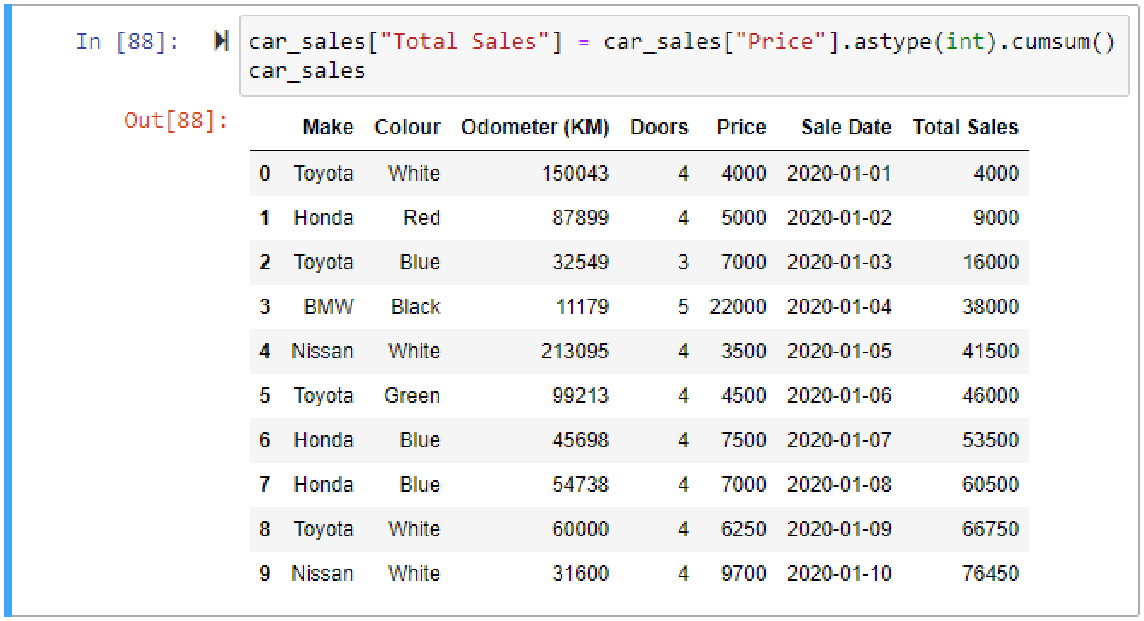


In [85]:

```
car_sales
```

Out[85]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date
0	Toyota	White	150043	4	4000	2020-01-01
1	Honda	Red	87899	4	5000	2020-01-02
2	Toyota	Blue	32549	3	7000	2020-01-03
3	BMW	Black	11179	5	22000	2020-01-04
4	Nissan	White	213095	4	3500	2020-01-05
5	Toyota	Green	99213	4	4500	2020-01-06
6	Honda	Blue	45698	4	7500	2020-01-07
7	Honda	Blue	54738	4	7000	2020-01-08
8	Toyota	White	60000	4	6250	2020-01-09
9	Nissan	White	31600	4	9700	2020-01-10

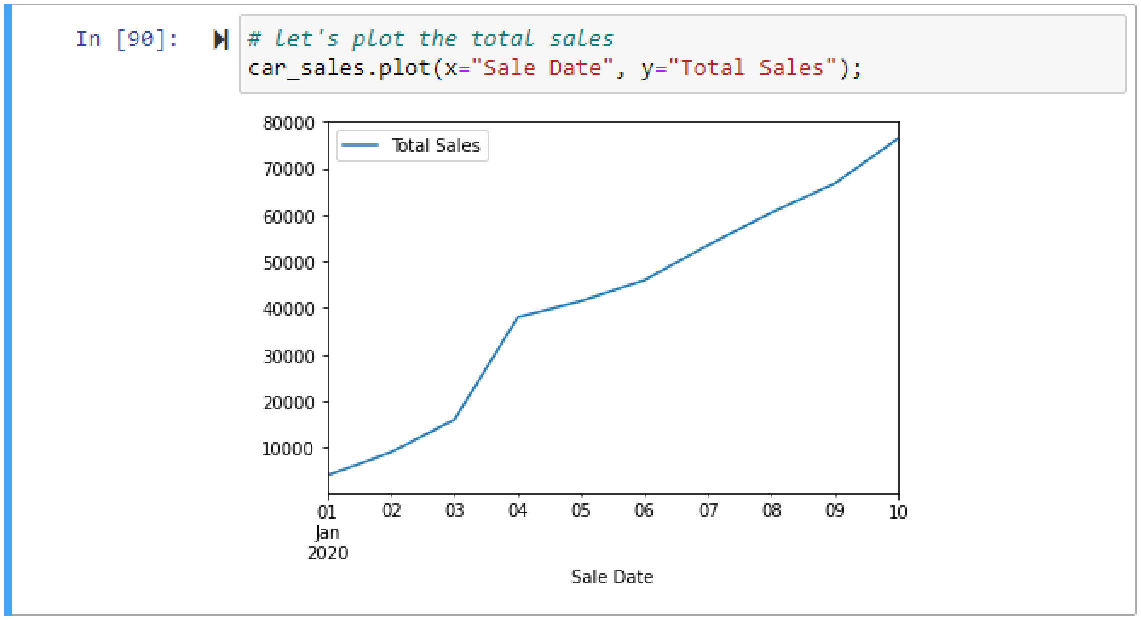


In [88]:

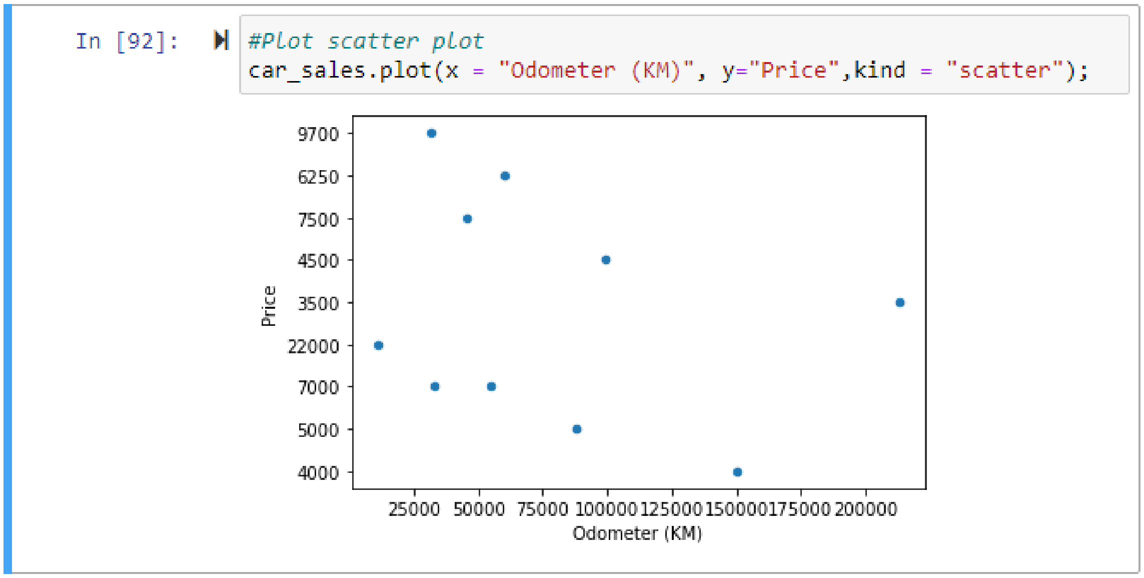
```
car_sales["Total Sales"] = car_sales["Price"].astype(int).cumsum()
car_sales
```

Out[88]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	4000	2020-01-01	4000
1	Honda	Red	87899	4	5000	2020-01-02	9000
2	Toyota	Blue	32549	3	7000	2020-01-03	16000
3	BMW	Black	11179	5	22000	2020-01-04	38000
4	Nissan	White	213095	4	3500	2020-01-05	41500
5	Toyota	Green	99213	4	4500	2020-01-06	46000
6	Honda	Blue	45698	4	7500	2020-01-07	53500
7	Honda	Blue	54738	4	7000	2020-01-08	60500
8	Toyota	White	60000	4	6250	2020-01-09	66750
9	Nissan	White	31600	4	9700	2020-01-10	76450



```
# let's plot the total sales
car_sales.plot(x="Sale Date", y="Total Sales");
```



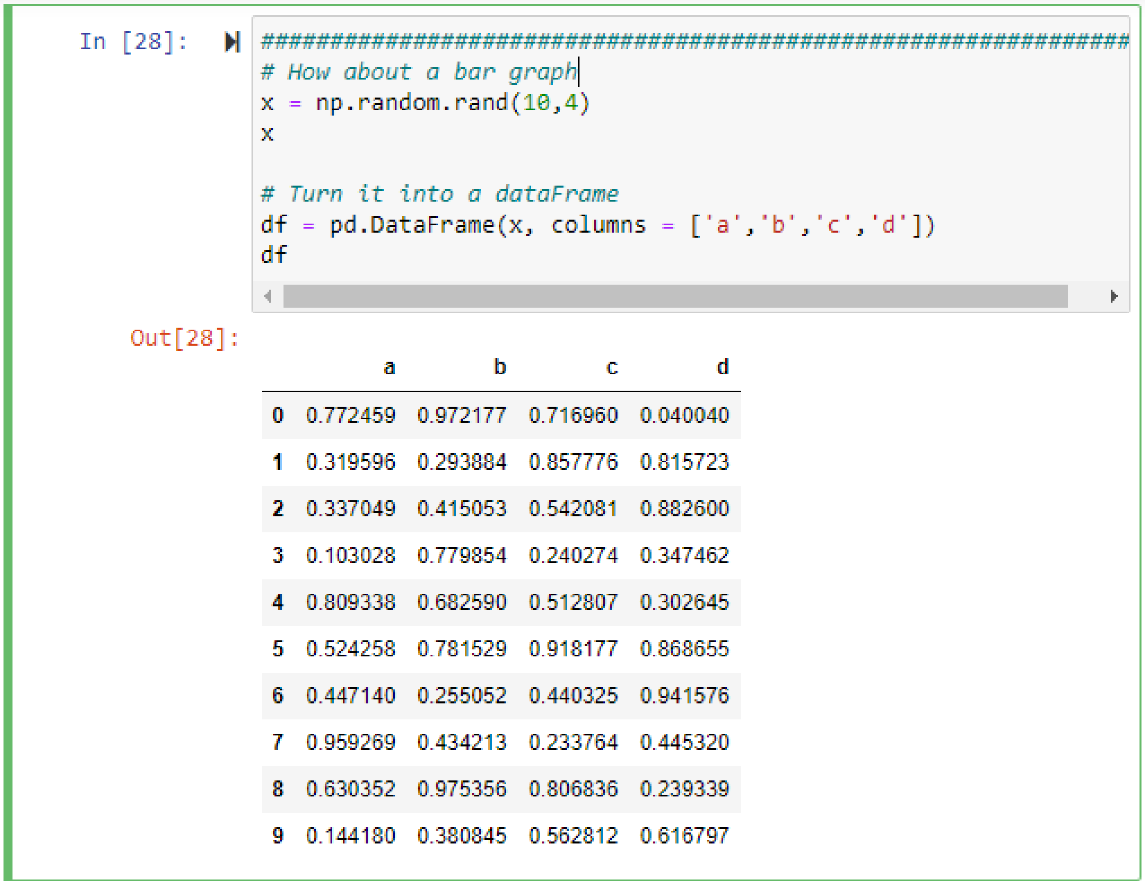
```
#Plot scatter plot
car_sales.plot(x = "Odometer (KM)", y="Price",kind = "scatter");
```

Examples of ploting

Example 1

```
# How about a bar graph
x = np.random.rand(10,4)
x

# Turn it into a dataFrame
df = pd.DataFrame(x, columns = ['a','b','c','d'])
```

```
df
```



In [28]:

```
##################################################################
# How about a bar graph
x = np.random.rand(10,4)
x

# Turn it into a dataFrame
df = pd.DataFrame(x, columns = ['a','b','c','d'])
df
```

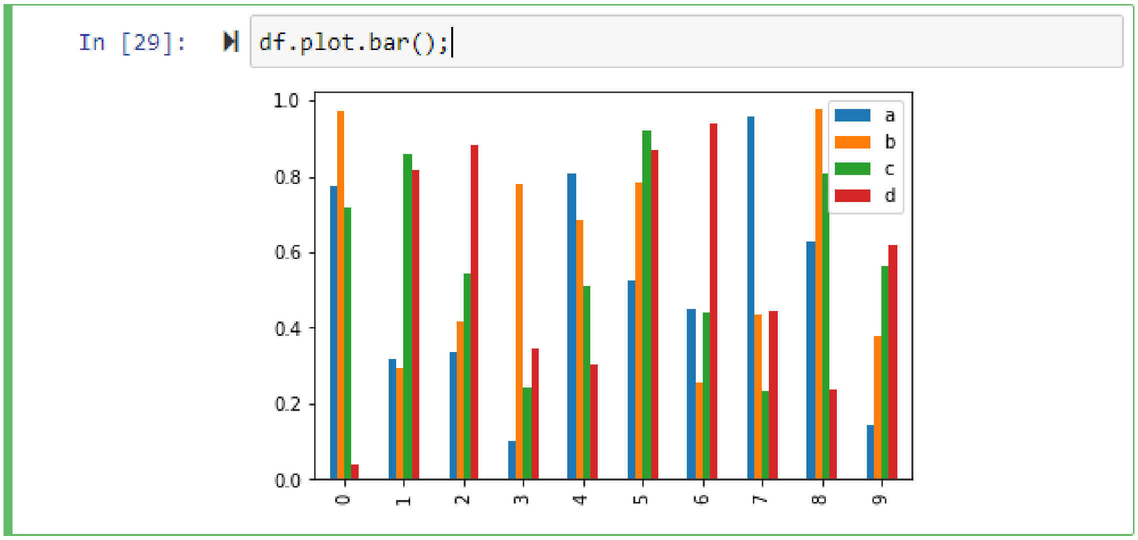
Out[28]:

	a	b	c	d
0	0.772459	0.972177	0.716960	0.040040
1	0.319596	0.293884	0.857776	0.815723
2	0.337049	0.415053	0.542081	0.882600
3	0.103028	0.779854	0.240274	0.347462
4	0.809338	0.682590	0.512807	0.302645
5	0.524258	0.781529	0.918177	0.868655
6	0.447140	0.255052	0.440325	0.941576
7	0.959269	0.434213	0.233764	0.445320
8	0.630352	0.975356	0.806836	0.239339
9	0.144180	0.380845	0.562812	0.616797

```
df.plot.bar();
```

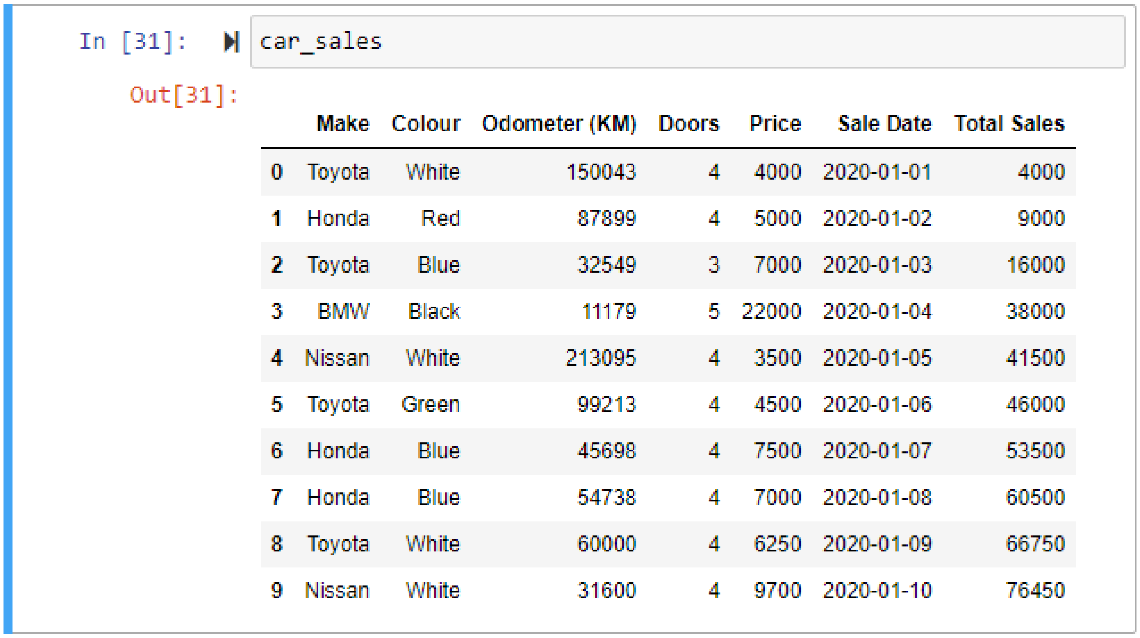
Same output

```
df.plot(kind="bar");
```



```
In [29]:  df.plot.bar();
```

Example 2

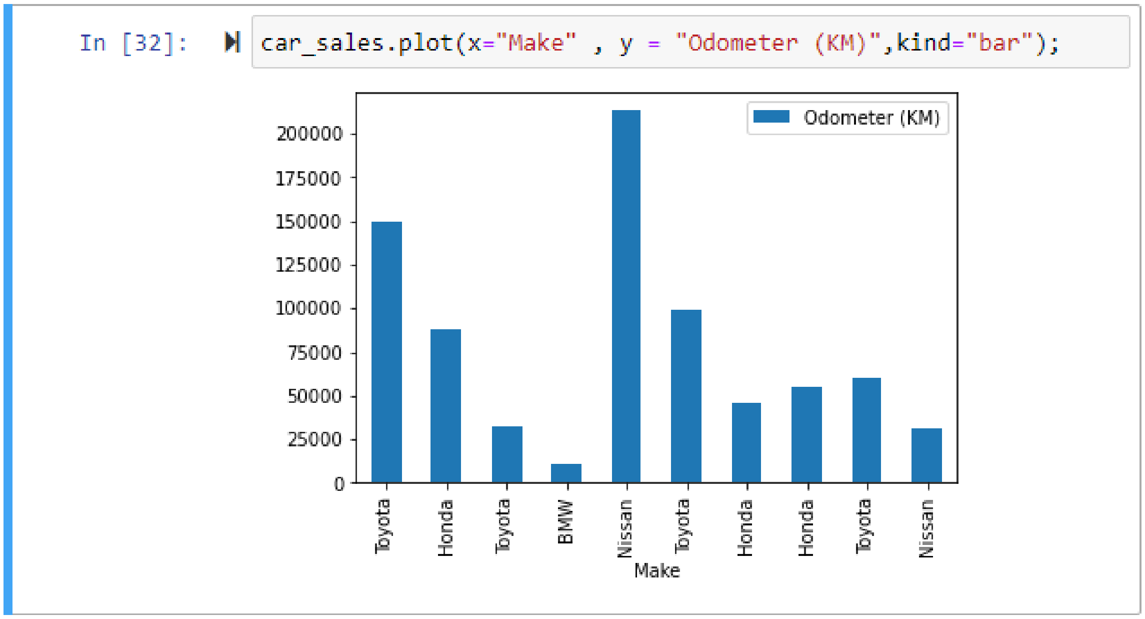


```
In [31]:  car_sales
```

Out[31]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	4000	2020-01-01	4000
1	Honda	Red	87899	4	5000	2020-01-02	9000
2	Toyota	Blue	32549	3	7000	2020-01-03	16000
3	BMW	Black	11179	5	22000	2020-01-04	38000
4	Nissan	White	213095	4	3500	2020-01-05	41500
5	Toyota	Green	99213	4	4500	2020-01-06	46000
6	Honda	Blue	45698	4	7500	2020-01-07	53500
7	Honda	Blue	54738	4	7000	2020-01-08	60500
8	Toyota	White	60000	4	6250	2020-01-09	66750
9	Nissan	White	31600	4	9700	2020-01-10	76450

```
car_sales.plot(x="Make" , y = "Odometer (KM)",kind="bar");
```



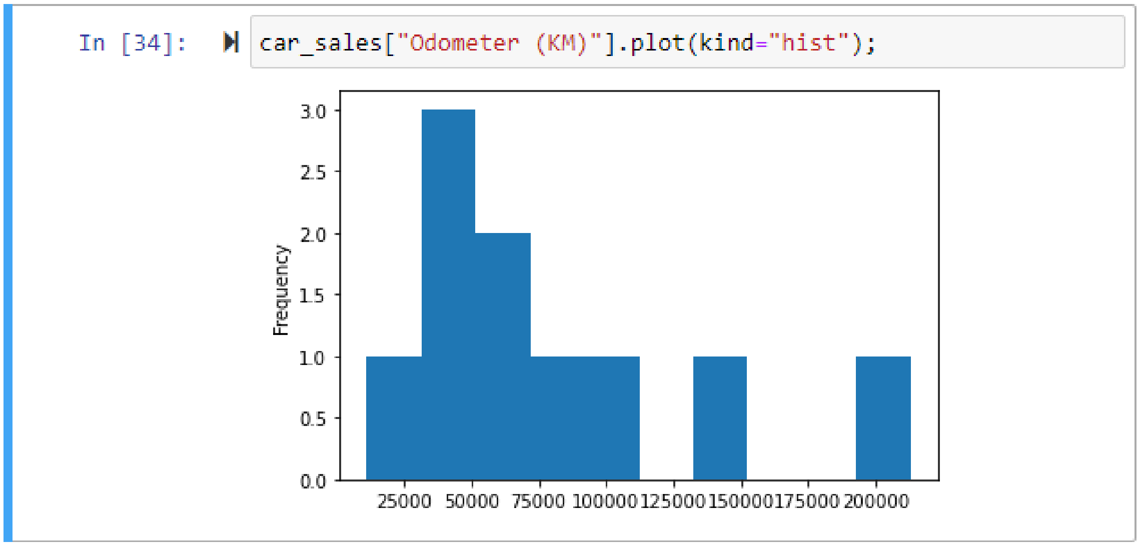
```
In [32]:  car_sales.plot(x="Make" , y = "Odometer (KM)",kind="bar");
```

Example 3

```
# how about histogram
car_sales["Odometer (KM)"].plot.hist();
```

Same output

```
car_sales["Odometer (KM)"].plot(kind="hist");
```



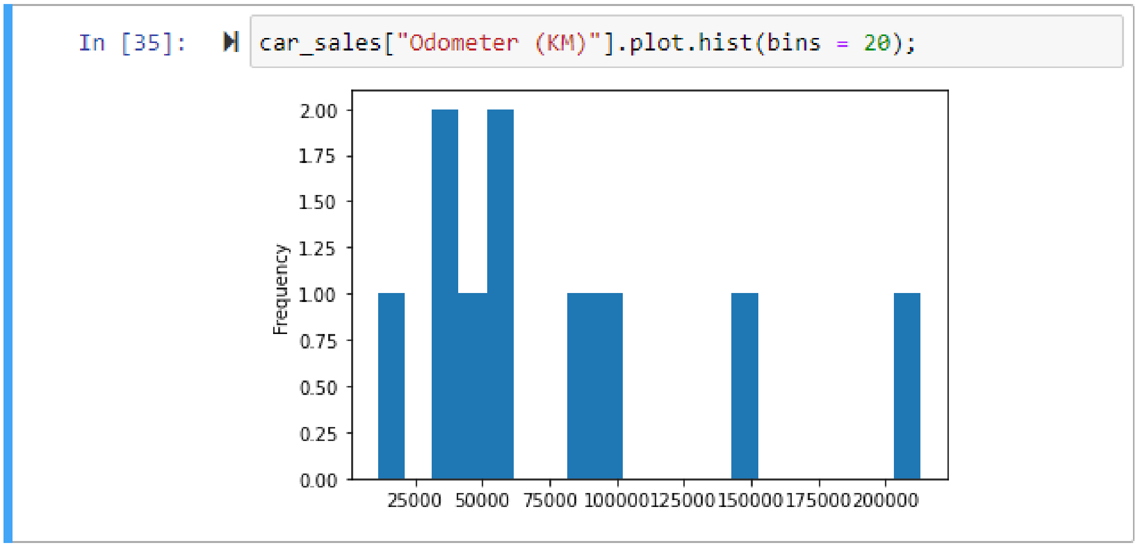
```
In [34]:  car_sales["Odometer (KM)"].plot(kind="hist");
```

Bins in histogram

The towers or bars of a histogram are called bins. The height of each bin shows how many values from that data fall into that range

The default value of the number of bins to be created in a histogram is 10.

```
car_sales["Odometer (KM)"].plot.hist(bins = 20);
```



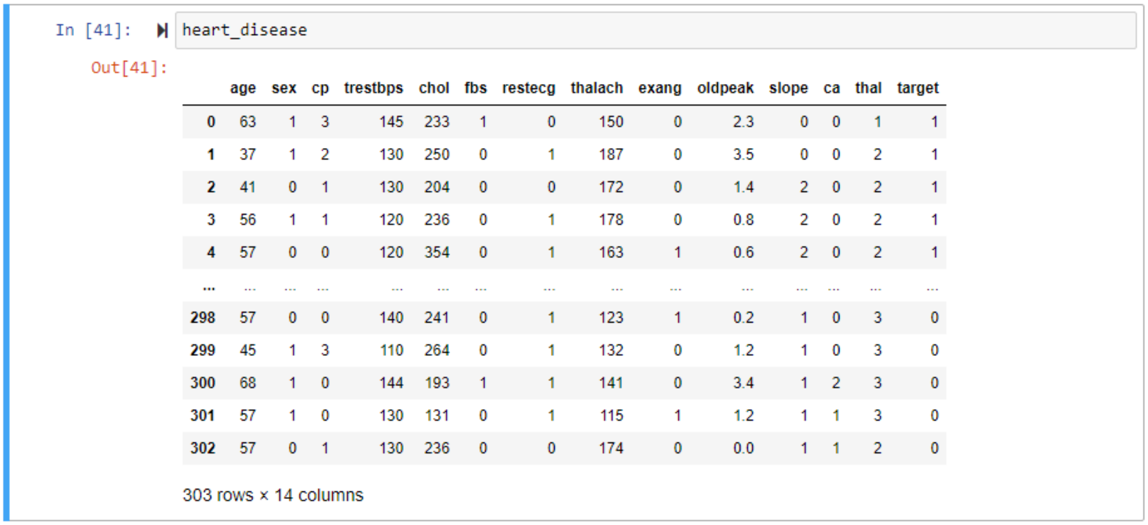
```
In [35]:  car_sales["Odometer (KM)"].plot.hist(bins = 20);
```

Which one should you use? (pyplot vs matplotlib OO method?)

- when plotting something quickly, okay to use the pyplot method.
- when plotting something more advanced, use the OO method.

```
heart_disease = pd.read_csv("11.2 heart-disease.csv")
```

```
heart_disease
```

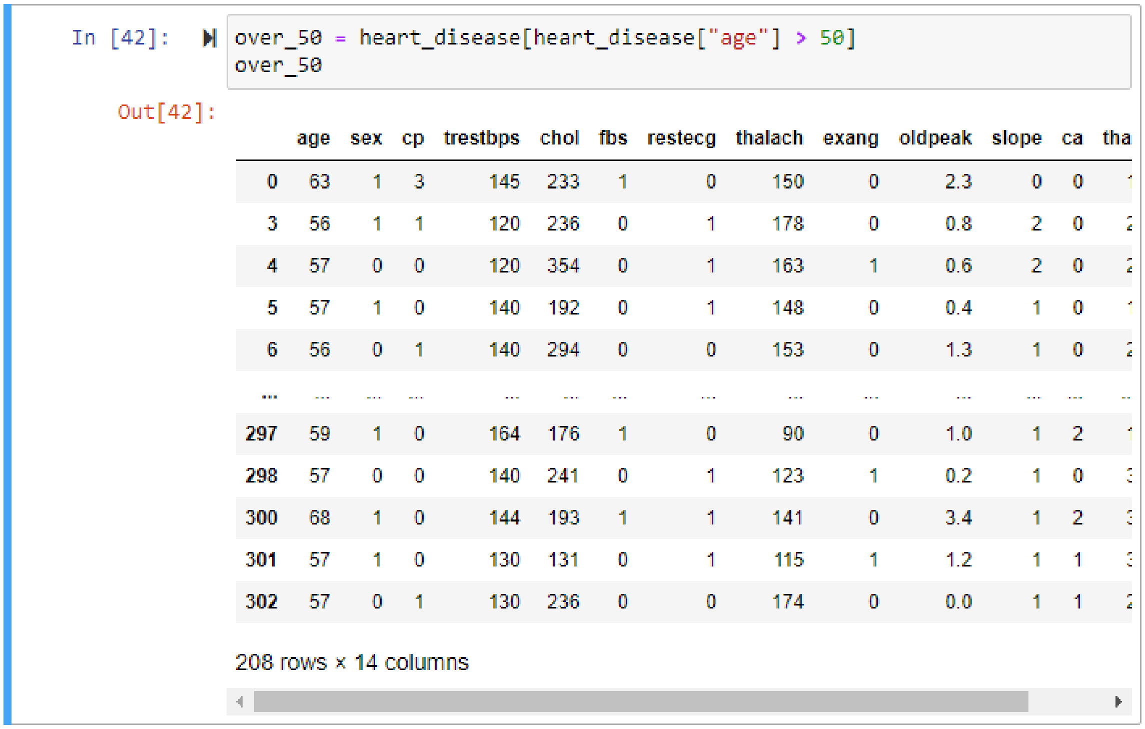


```
In [41]:  heart_disease
Out[41]:
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal  target
  0   63    1   3       145   233    1        0      150      0      2.3      0   0     1       1
  1   37    1   2       130   250    0        1      187      0      3.5      0   0     2       1
  2   41    0   1       130   204    0        0      172      0      1.4      2   0     2       1
  3   56    1   1       120   236    0        1      178      0      0.8      2   0     2       1
  4   57    0   0       120   354    0        1      163      1      0.6      2   0     2       1
...  ...  ...  ...      ...   ...  ...      ...      ...    ...      ...    ...  ...   ...     ...
298   57    0   0       140   241    0        1      123      1      0.2      1   0     3       0
299   45    1   3       110   264    0        1      132      0      1.2      1   0     3       0
300   68    1   0       144   193    1        1      141      0      3.4      1   2     3       0
301   57    1   0       130   131    0        1      115      1      1.2      1   1     3       0
302   57    0   1       130   236    0        0      174      0      0.0      1   1     2       0

303 rows × 14 columns
```

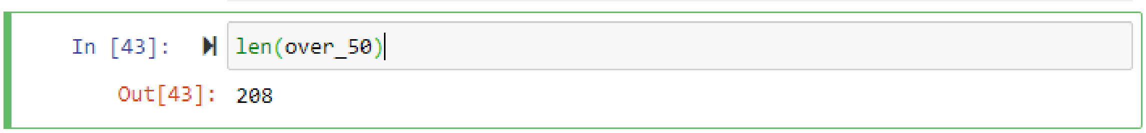
Filtering

```
over_50 = heart_disease[heart_disease["age"] > 50]
over_50
```



```
In [42]:  over_50 = heart_disease[heart_disease["age"] > 50]
          over_50
Out[42]:
```

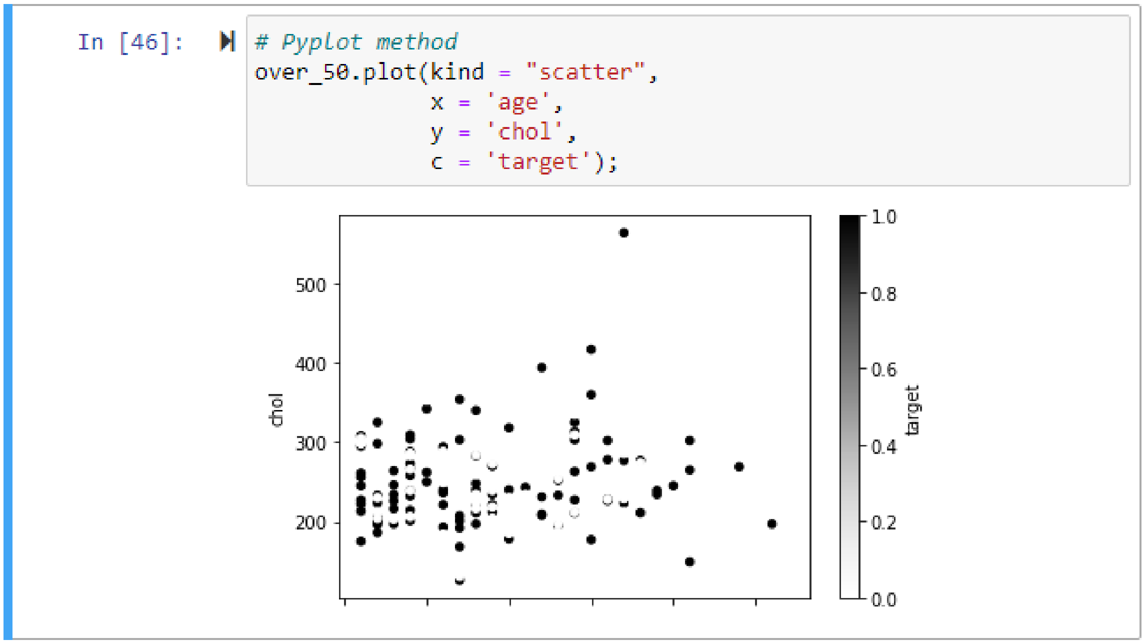
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  tha
  0   63    1   3       145   233    1        0      150      0      2.3      0   0    1
  3   56    1   1       120   236    0        1      178      0      0.8      2   0    2
  4   57    0   0       120   354    0        1      163      1      0.6      2   0    2
  5   57    1   0       140   192    0        1      148      0      0.4      1   0    1
  6   56    0   1       140   294    0        0      153      0      1.3      1   0    2
...  ...  ...  ...      ...   ...  ...      ...      ...    ...      ...    ...  ...  ...
297   59    1   0       164   176    1        0       90      0      1.0      1   2    1
298   57    0   0       140   241    0        1      123      1      0.2      1   0    3
300   68    1   0       144   193    1        1      141      0      3.4      1   2    3
301   57    1   0       130   131    0        1      115      1      1.2      1   1    3
302   57    0   1       130   236    0        0      174      0      0.0      1   1    2

208 rows × 14 columns
```



```
In [43]:  len(over_50)
Out[43]:  208
```

Pyplot method

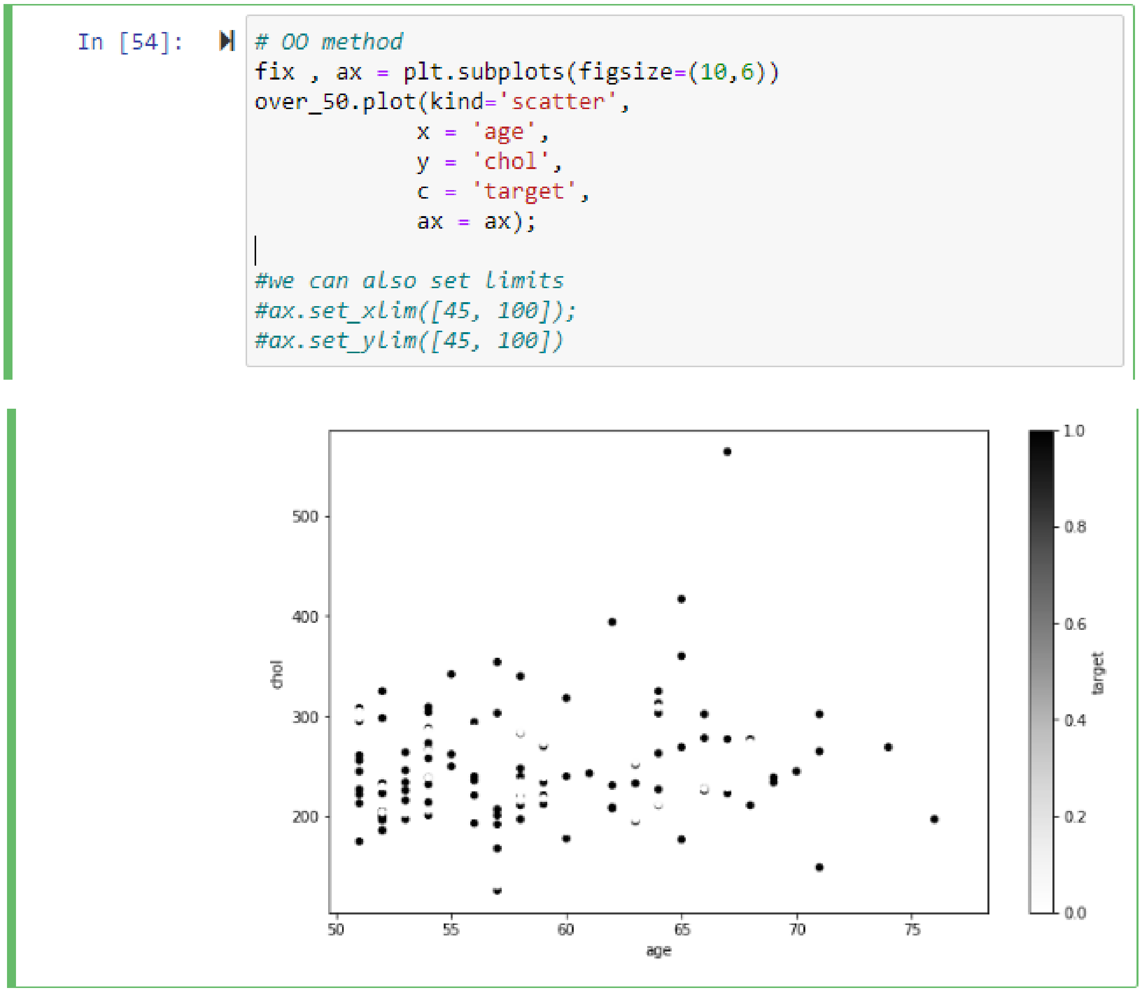
```
# Pyplot method
over_50.plot(kind = "scatter",
             x = 'age',
             y = 'chol',
             c = 'target');
```



```
In [46]:  # Pyplot method
          over_50.plot(kind = "scatter",
                       x = 'age',
                       y = 'chol',
                       c = 'target');
```

OO method (Object-Oriented) mixed with pyplot method

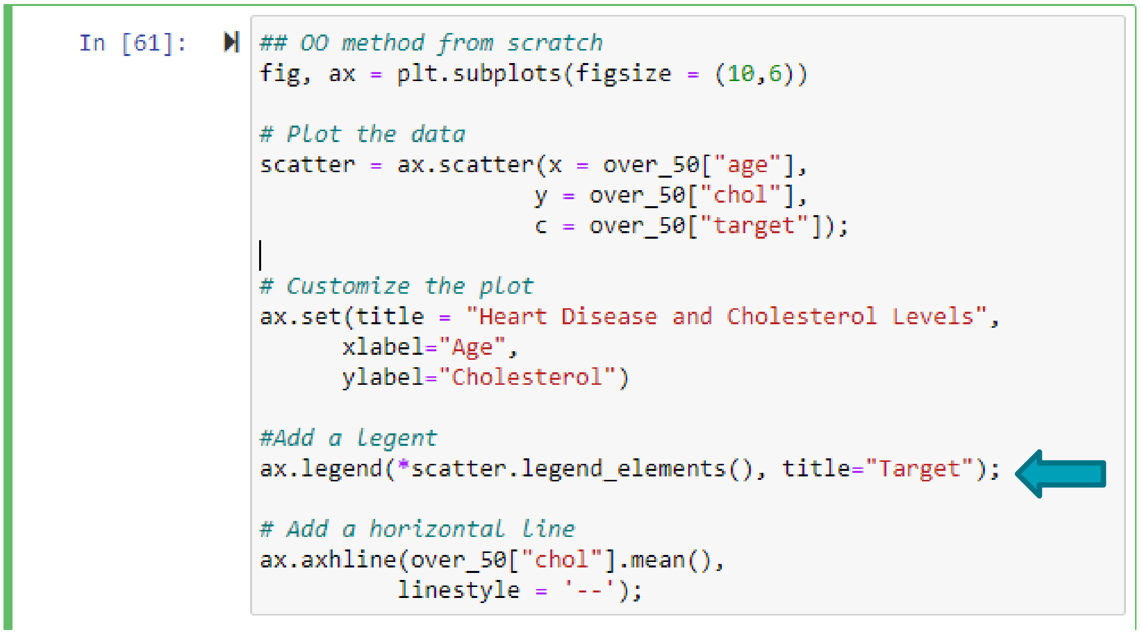
```
# OO method
fix , ax = plt.subplots(figsize=(10,6))
over_50.plot(kind='scatter',
            x = 'age',
            y = 'chol',
            c = 'target',
            ax = ax);

#we can also set limits
#ax.set_xlim([45, 100]);
#ax.set_ylim([45, 100])
```



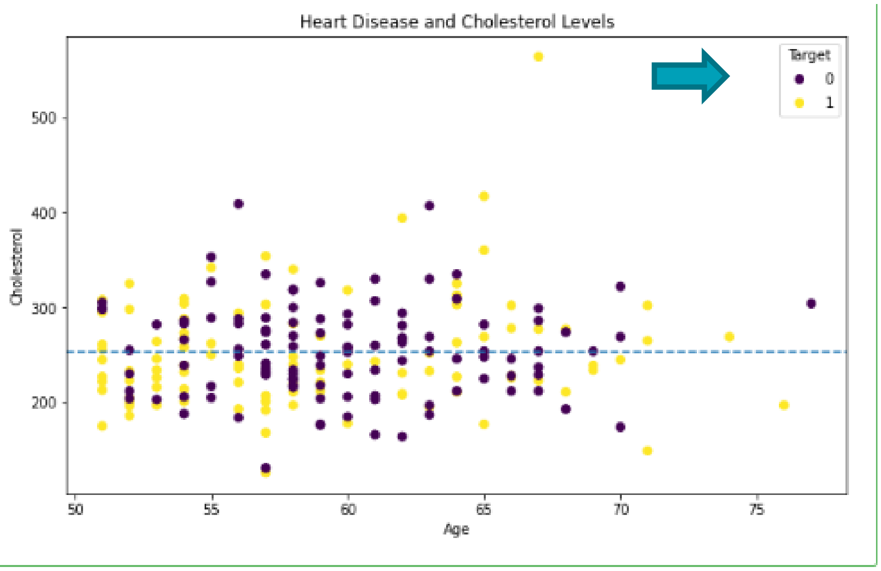
```
In [54]:  # OO method
           fix , ax = plt.subplots(figsize=(10,6))
           over_50.plot(kind='scatter',
                        x = 'age',
                        y = 'chol',
                        c = 'target',
                        ax = ax);

           #we can also set limits
           #ax.set_xlim([45, 100]);
           #ax.set_ylim([45, 100])
```

OO From Scratch

```
## OO method from scratch
fig, ax = plt.subplots(figsize = (10,6))

# Plot the data
scatter = ax.scatter(x = over_50["age"],
                     y = over_50["chol"],
                    c = over_50["target"]);

# Customize the plot
ax.set(title = "Heart Disease and Cholesterol Levels",
      xlabel="Age",
      ylabel="Cholesterol")

#Add a legent
ax.legend(*scatter.legend_elements(), title="Target");

# Add a horizontal line
ax.axhline(over_50["chol"].mean(),
          linestyle = '--');
```

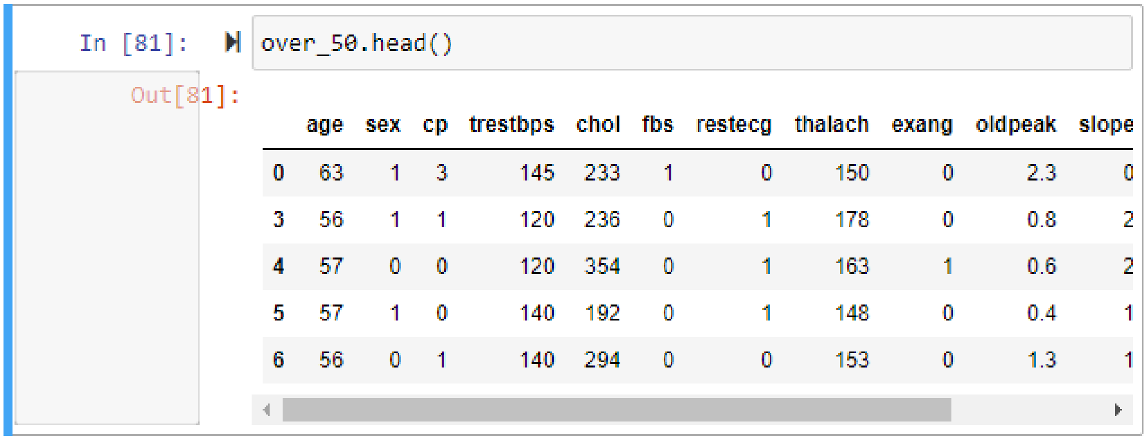


```
## OO method from scratch
fig, ax = plt.subplots(figsize = (10,6))

# Plot the data
scatter = ax.scatter(x = over_50["age"],
                    y = over_50["chol"],
                    c = over_50["target"]);

# Customize the plot
ax.set(title = "Heart Disease and Cholesterol Levels",
      xlabel="Age",
      ylabel="Cholesterol")

#Add a legent
ax.legend(*scatter.legend_elements(), title="Target");

# Add a horizontal line
ax.axhline(over_50["chol"].mean(),
          linestyle = '--');
```



Heart disease example on OO Method

data

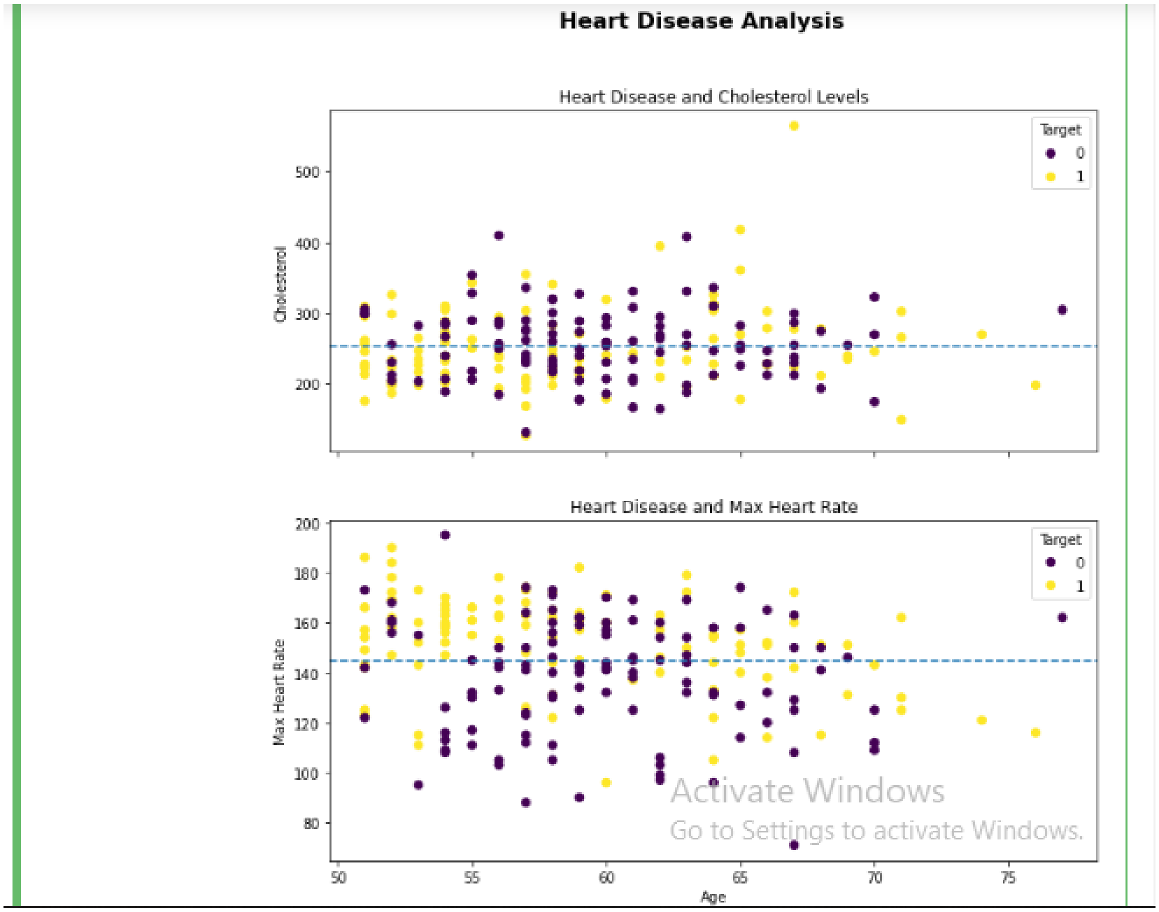


```
In [81]:  over_50.head()
```

Out[81]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	63	1	3	145	233	1	0	150	0	2.3	0
3	56	1	1	120	236	0	1	178	0	0.8	2
4	57	0	0	120	354	0	1	163	1	0.6	2
5	57	1	0	140	192	0	1	148	0	0.4	1
6	56	0	1	140	294	0	0	153	0	1.3	1

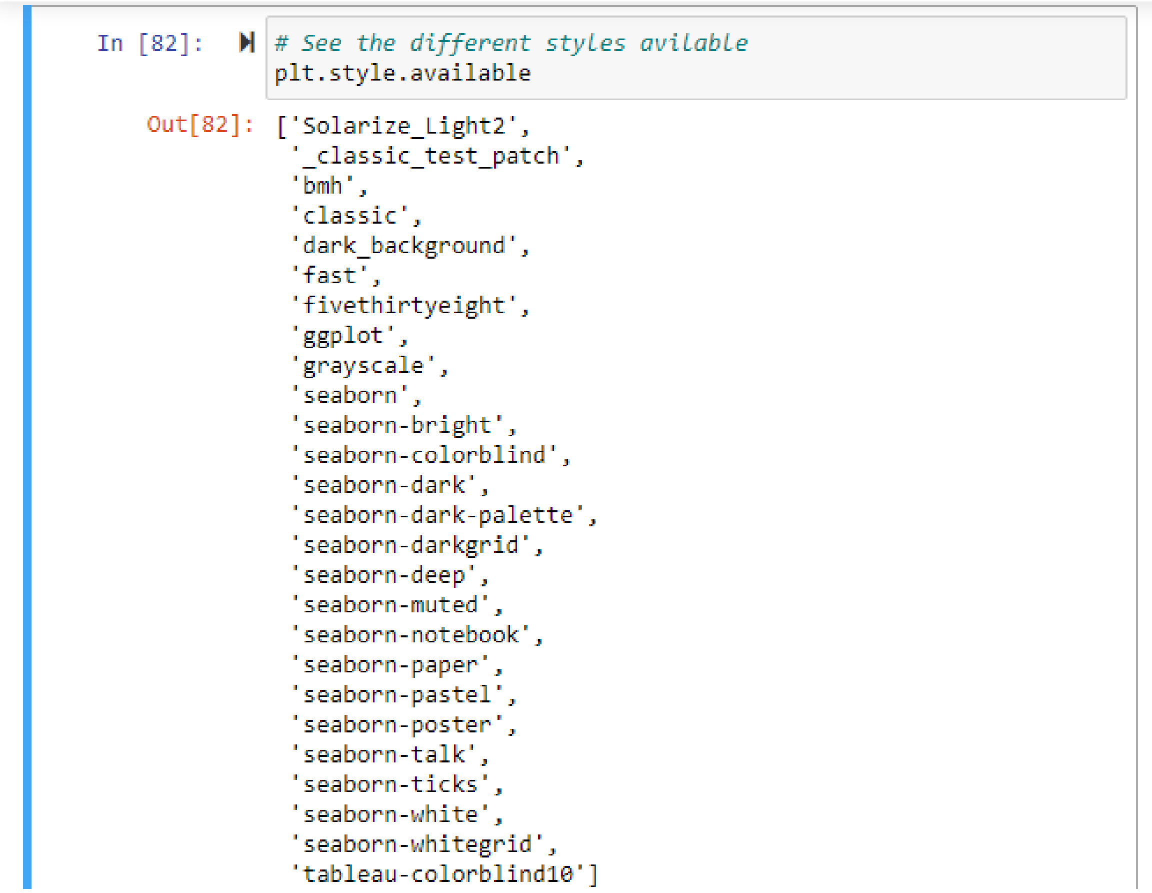
```
# Subplot of chol, age, thalach
fig, (ax0, ax1) = plt.subplots(nrows= 2 ,
                               ncols = 1,
                               figsize = (10,10),
                               sharex=True)


# Add data to ax0
scatter = ax0.scatter(x = over_50["age"],
                    y = over_50["chol"],
                    c = over_50["target"]);

#Customize ax0
ax0.set(title = "Heart Disease and Cholesterol Levels",
     #xlabel="Age", because of sharex=True

      ylabel="Cholesterol")

#Add a legent to ax0
ax0.legend(*scatter.legend_elements(), title="Target");

# Add a horizontal line
ax0.axhline(over_50["chol"].mean(),
          linestyle = '--');




### ax1
# Add data to ax1
scatter = ax1.scatter(x = over_50["age"],
                    y = over_50["thalach"],
                    c = over_50["target"]);

#Customize ax1
ax1.set(title = "Heart Disease and Max Heart Rate",
      xlabel="Age",
      ylabel="Max Heart Rate")

#Add a legent to ax1
ax1.legend(*scatter.legend_elements(), title="Target");

# Add a horizontal line at ax1
ax1.axhline(y = over_50["thalach"].mean(),
          linestyle = '--');

#Add a title to the figure
fig.suptitle("Heart Disease Analysis", fontsize = 16,
fontweight="bold");
```



Customizing Matplotlib plots and getting stylish

See the different styles available

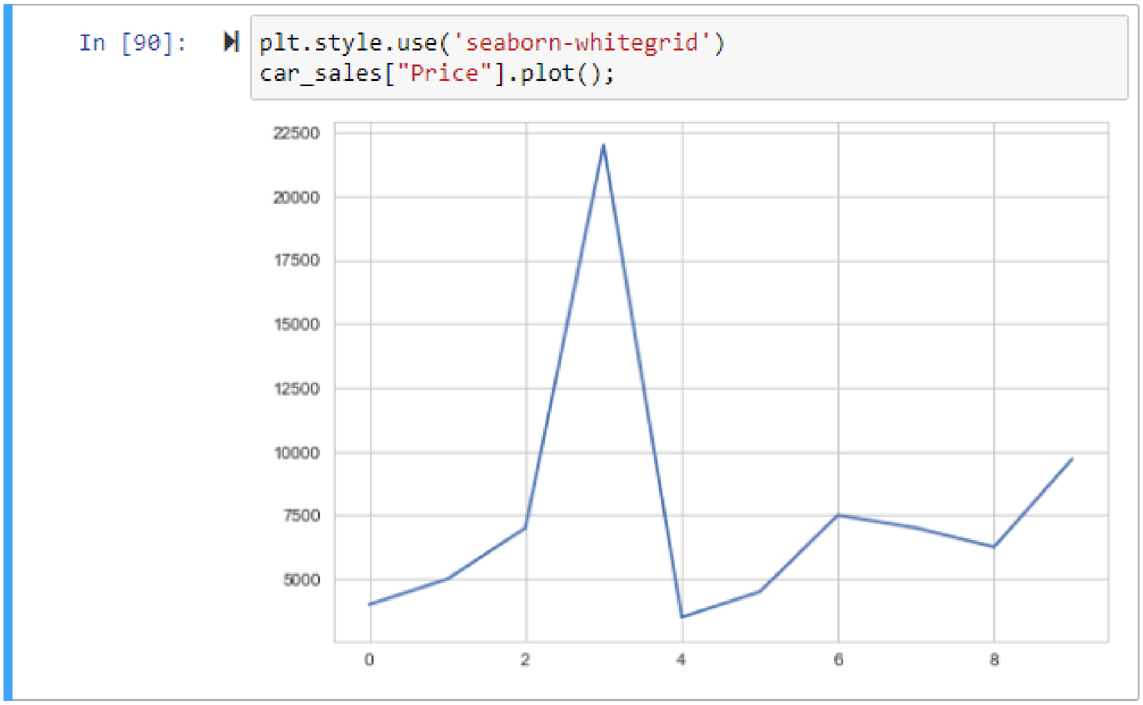
```
# See the different styles available
plt.style.available
```



```
In [82]:  # See the different styles avilable
          plt.style.available

Out[82]: ['Solarize_Light2',
          '_classic_test_patch',
          'bmh',
          'classic',
          'dark_background',
          'fast',
          'fivethirtyeight',
          'ggplot',
          'grayscale',
          'seaborn',
          'seaborn-bright',
          'seaborn-colorblind',
          'seaborn-dark',
          'seaborn-dark-palette',
          'seaborn-darkgrid',
          'seaborn-deep',
          'seaborn-muted',
          'seaborn-notebook',
          'seaborn-paper',
          'seaborn-pastel',
          'seaborn-poster',
          'seaborn-talk',
          'seaborn-ticks',
          'seaborn-white',
          'seaborn-whitegrid',
          'tableau-colorblind10']
```

1)

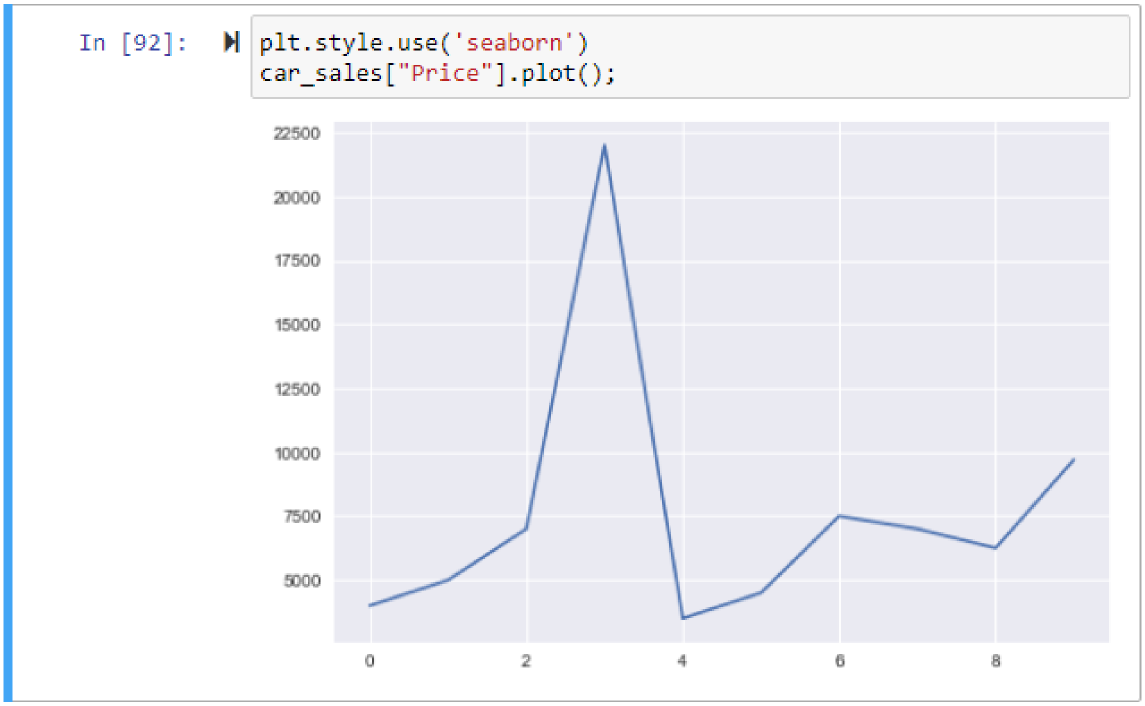
```
plt.style.use('seaborn-whitegrid')
car_sales["Price"].plot();
```



```
plt.style.use('seaborn-whitegrid')
car_sales["Price"].plot();
```

2)

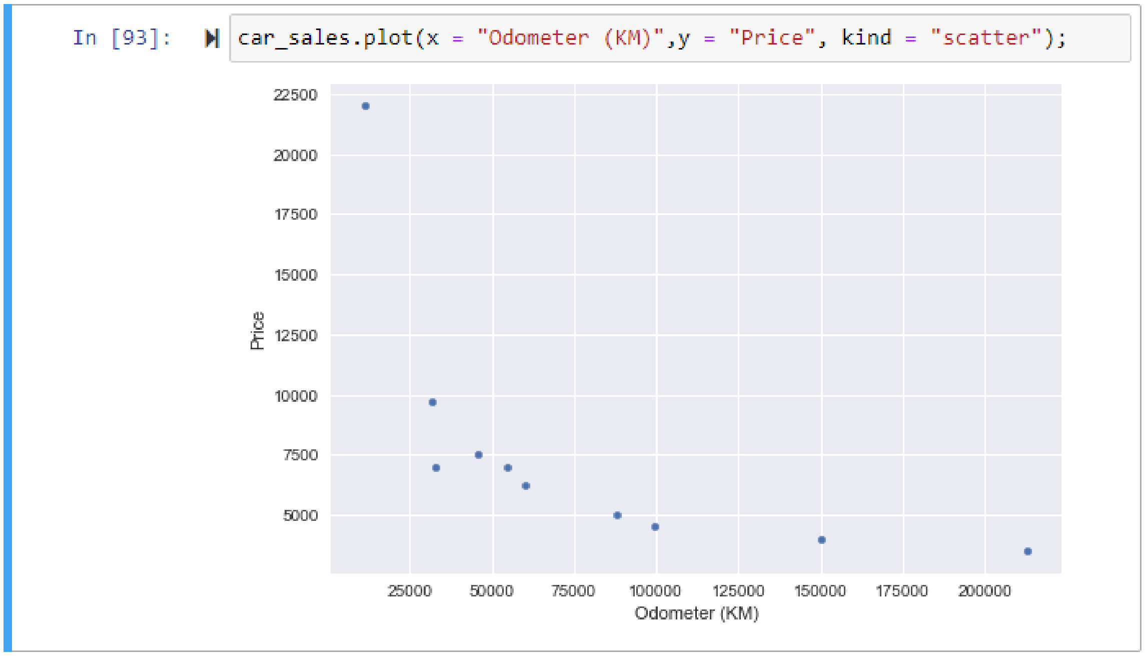
```
plt.style.use('seaborn')
car_sales["Price"].plot();
```



```
plt.style.use('seaborn')
car_sales["Price"].plot();
```

3)

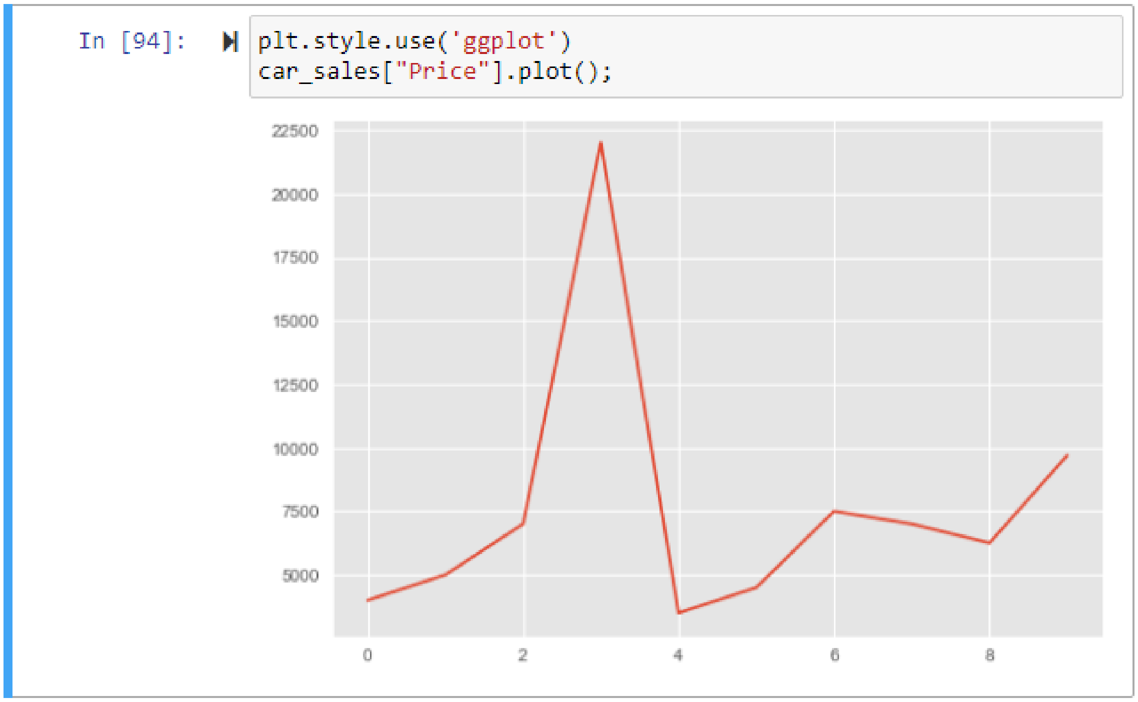
```
car_sales.plot(x = "Odometer (KM)",y = "Price", kind = "scatter");
```



```
In [93]:  car_sales.plot(x = "Odometer (KM)",y = "Price", kind = "scatter");
```

4)

```
plt.style.use('ggplot')
car_sales["Price"].plot();
```



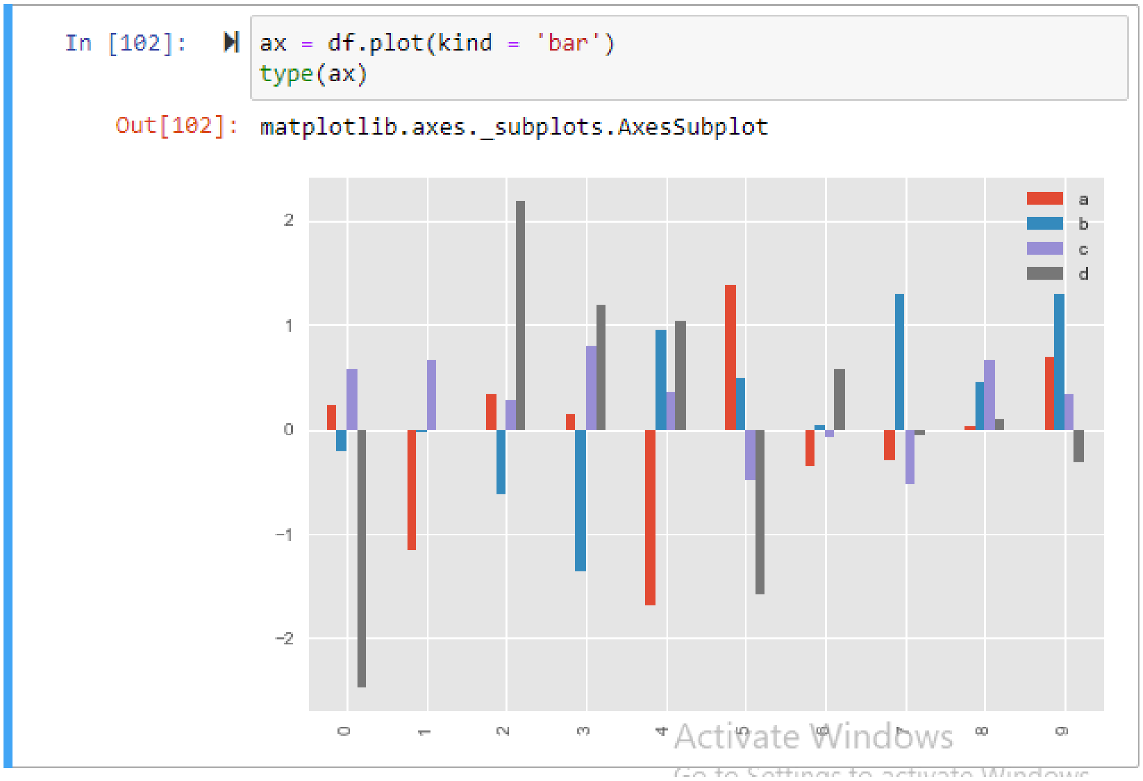
```
In [94]:  plt.style.use('ggplot')
          car_sales["Price"].plot();
```

5)

```
# create some data
x = np.random.randn(10, 4)
x
```

```
df = pd.DataFrame(x, columns=['a','b','c','d'])
```

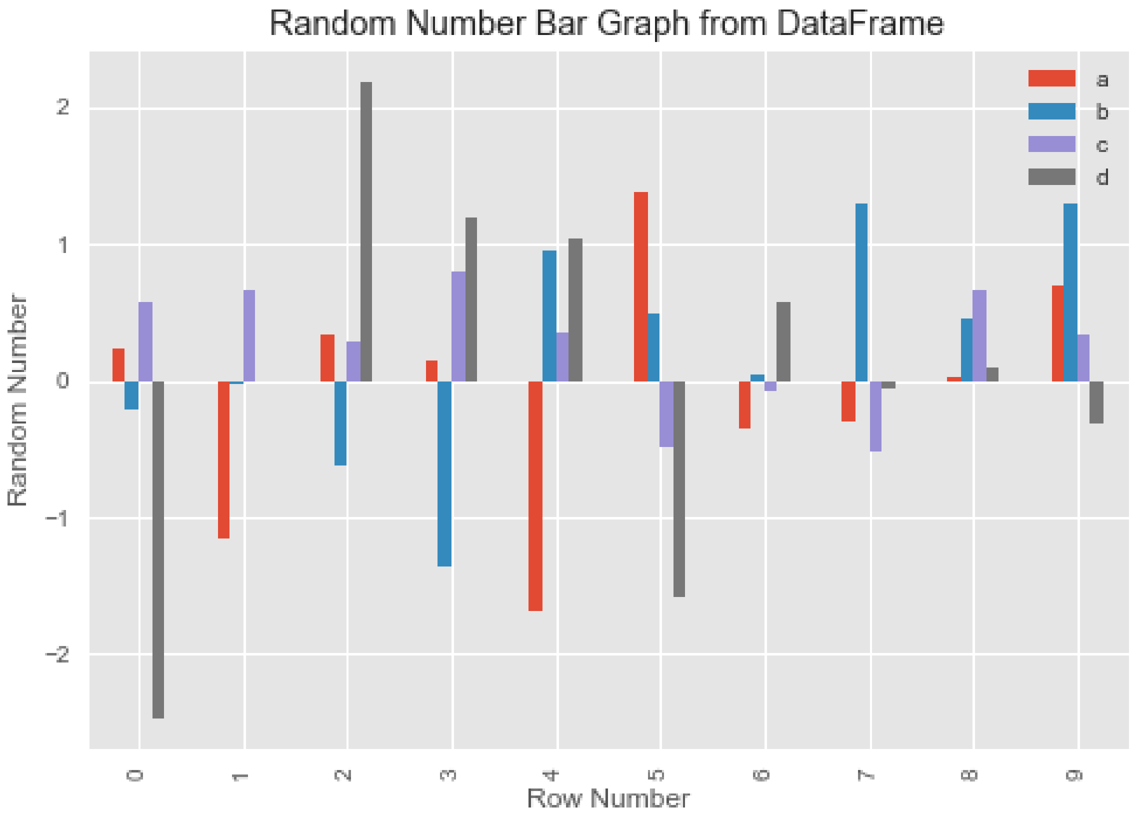
```
ax = df.plot(kind = 'bar')
type(ax)
```



```
In [102]:  ax = df.plot(kind = 'bar')
           type(ax)

Out[102]:  matplotlib.axes._subplots.AxesSubplot
```

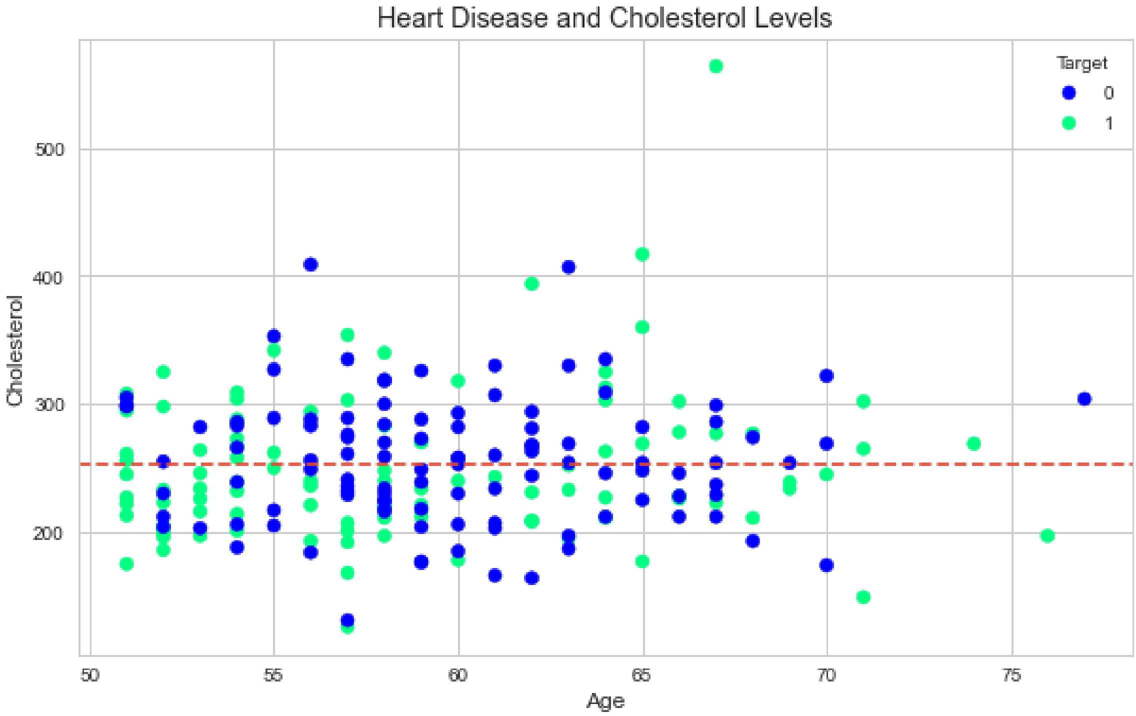
Customize our plot with the set() method

```
# Customize our plot with the set() method
ax = df.plot(kind = 'bar')
# Add some labels and a title
ax.set(title = "Random Number Bar Graph from DataFrame",
      xlabel = 'Row Number',
      ylabel = 'Random Number')

# Make the legend visible
ax.legend().set_visible(True)
```



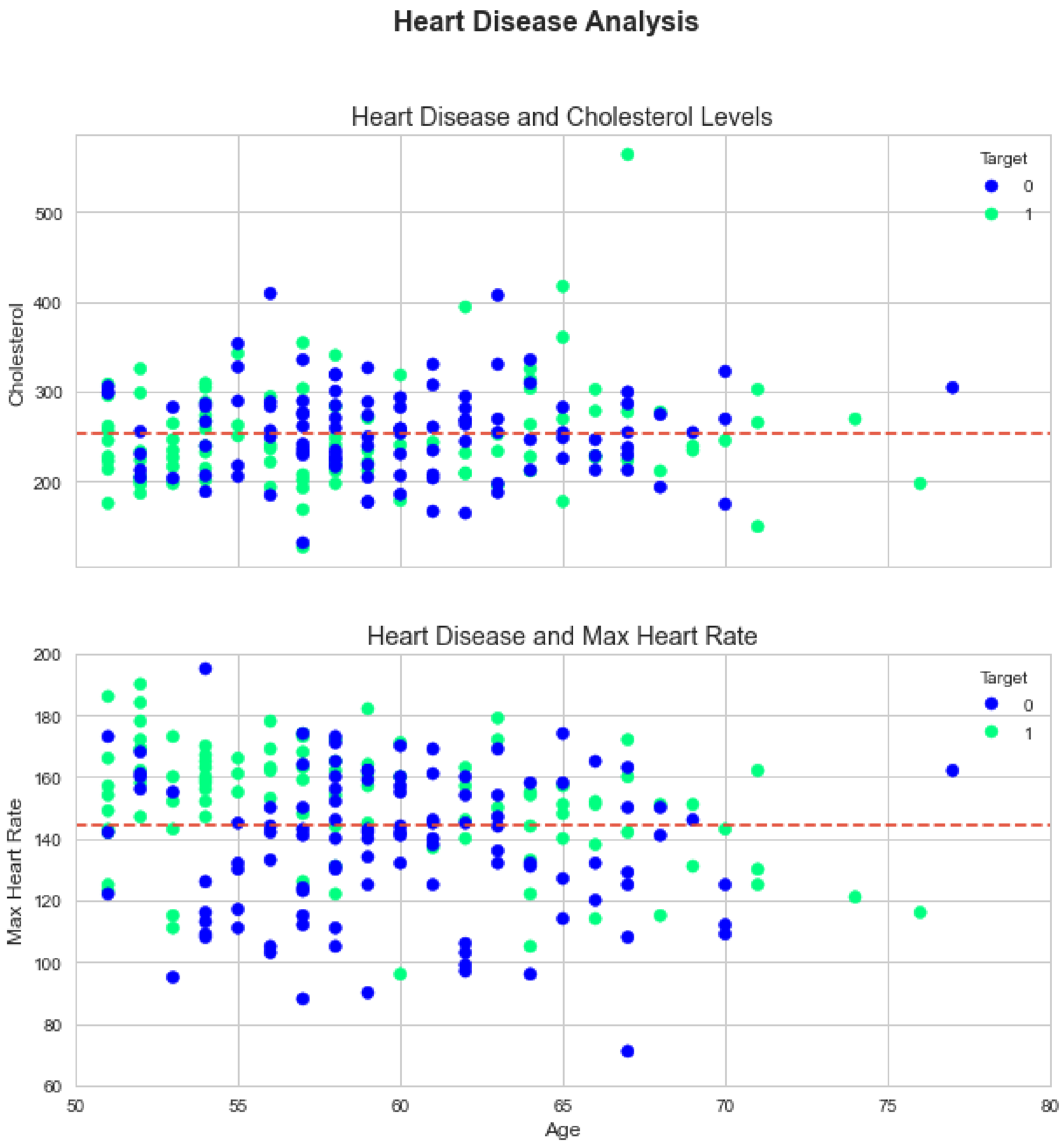
change the style again but from within an existing style.

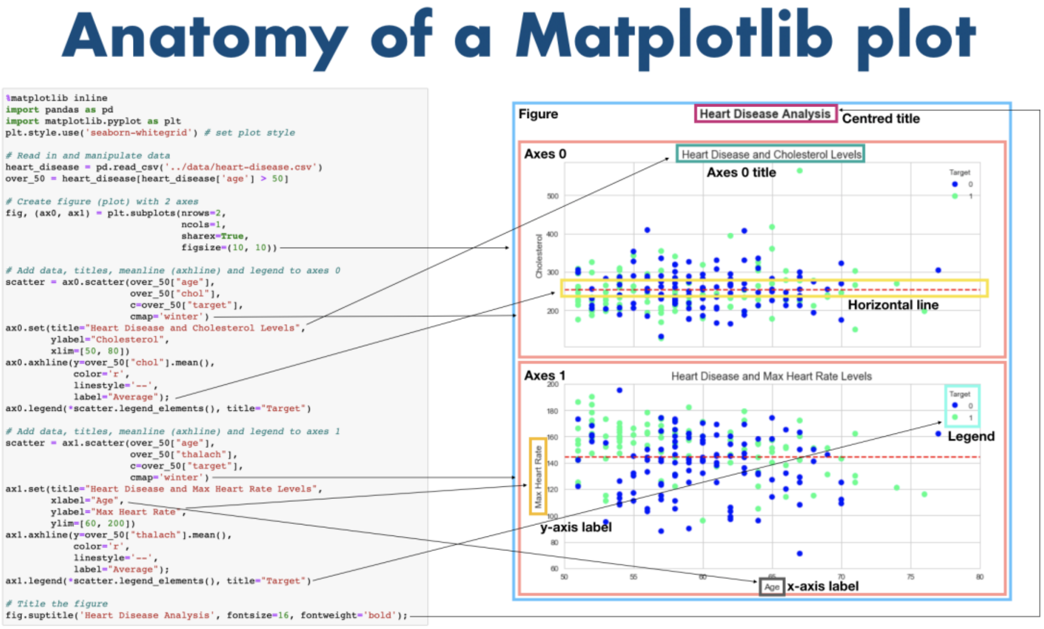
```
# Set the style
plt.style.use('seaborn-whitegrid')

## OO method from scratch
fig, ax = plt.subplots(figsize = (10,6))

# matplotlib colors tutorial for more color scheme

# Plot the data
scatter = ax.scatter(x = over_50["age"],
                     y = over_50["chol"],
                     c = over_50["target"],
                     cmap = "winter"); # this changes the color scheme
                            #"summer"
                            #"plasma'
# Customize the plot
ax.set(title = "Heart Disease and Cholesterol Levels",
      xlabel="Age",
      ylabel="Cholesterol")

#Add a legent
ax.legend(*scatter.legend_elements(), title="Target");

# Add a horizontal line
ax.axhline(over_50["chol"].mean(),
          linestyle = '--');
```



Customizing the y and x axis limitations

```
# Customizing the y and x axis limitations

# Subplot of chol, age, thalach
fig, (ax0, ax1) = plt.subplots(nrows= 2 ,
                               ncols = 1,
                               figsize = (10,10),
                               sharex=True)

# Add data to ax0
scatter = ax0.scatter(x = over_50["age"],
                    y = over_50["chol"],
                    c = over_50["target"],
                     cmap= "winter");

#Customize ax0
ax0.set(title = "Heart Disease and Cholesterol Levels",

     #xlabel="Age", because of sharex=True

      ylabel="Cholesterol")

#### change the x axis limits
ax0.set_xlim([50, 80])

#Add a legent to ax0
ax0.legend(*scatter.legend_elements(), title="Target");

# Add a horizontal line
ax0.axhline(over_50["chol"].mean(),
          linestyle = '--');




### ax1
# Add data to ax1
scatter = ax1.scatter(x = over_50["age"],
                    y = over_50["thalach"],
                    c = over_50["target"],
                     cmap= "winter");

#Customize ax1
ax1.set(title = "Heart Disease and Max Heart Rate",
      xlabel="Age",
      ylabel="Max Heart Rate")

#### Chande ax1 axis limits
ax1.set_xlim([50,80])
ax1.set_ylim([60,200])

#Add a legent to ax1
ax1.legend(*scatter.legend_elements(), title="Target");

# Add a horizontal line at ax1
ax1.axhline(y = over_50["thalach"].mean(),
          linestyle = '--');

#Add a title to the figure
fig.suptitle("Heart Disease Analysis", fontsize = 16,
fontweight="bold");
```


Heart Disease Analysis
Heart Disease and Cholesterol Levels
Cholesterol
500
400
300
200
Target
0
1
Heart Disease and Max Heart Rate
Max Heart Rate
200
180
160
140
120
100
80
60
Target
0
1
50
55
60
65
70
75
80
Age



Anatomy of a Matplotlib plot

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid') # set plot style

# Read in and manipulate data
heart_disease = pd.read_csv('../data/heart-disease.csv')
over_50 = heart_disease[heart_disease['age'] > 50]

# Create figure (plot) with 2 axes
fig, (ax0, ax1) = plt.subplots(nrows=2,
                               ncols=1,
                               sharex=True,
                               figsize=(10, 10))

# Add data, titles, meanline (axhline) and legend to axes 0
scatter = ax0.scatter(over_50["age"],
                      over_50["chol"],
                      c=over_50["target"],
                      cmap='winter')
ax0.set(title="Heart Disease and Cholesterol Levels",
        ylabel="Cholesterol",
        xlim=[50, 80])
ax0.axhline(y=over_50["chol"].mean(),
            color='r',
            linestyle='--',
            label="Average");
ax0.legend(*scatter.legend_elements(), title="Target")

# Add data, titles, meanline (axhline) and legend to axes 1
scatter = ax1.scatter(over_50["age"],
                      over_50["thalach"],
                      c=over_50["target"],
                      cmap='winter')
ax1.set(title="Heart Disease and Max Heart Rate Levels",
        xlabel="Age",
        ylabel="Max Heart Rate",
        ylim=[60, 200])
ax1.axhline(y=over_50["thalach"].mean(),
            color='r',
            linestyle='--',
            label="Average");
ax1.legend(*scatter.legend_elements(), title="Target")

# Title the figure
fig.suptitle('Heart Disease Analysis', fontsize=16, fontweight='bold');
```

Saving plots

By code

```
fig.savefig("Heart-disease-analysis-plot-saved-with-code")
```

Manually

By copy and paste