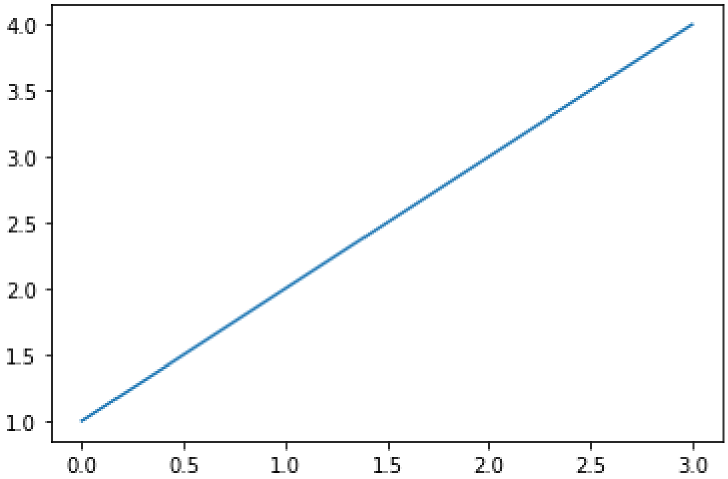
Matplotlib Template

In [2]:
```
# Matplotlib, similar to MATLAB, each pyplot creates/changes the figure

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

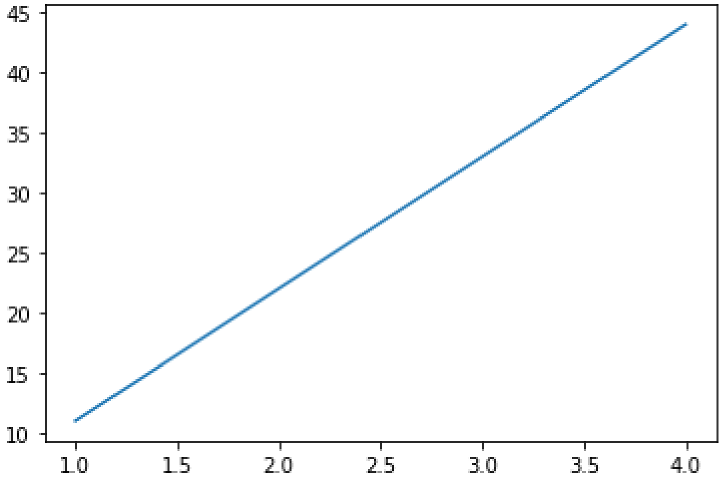
In [4]:
```
# one dimentional
plt.plot([1,2,3,4])nm
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x114255390>]
```

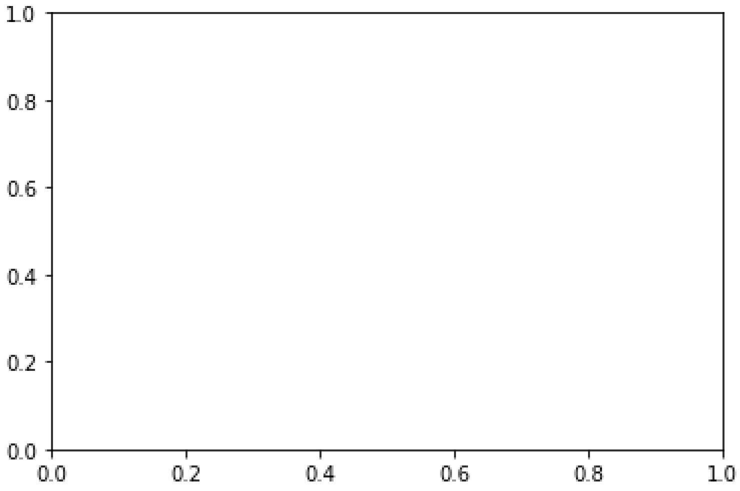


In [5]:
```
#two dimentional
x=[1,2,3,4]
y=[11,22,33,44]
plt.plot(x,y)
```

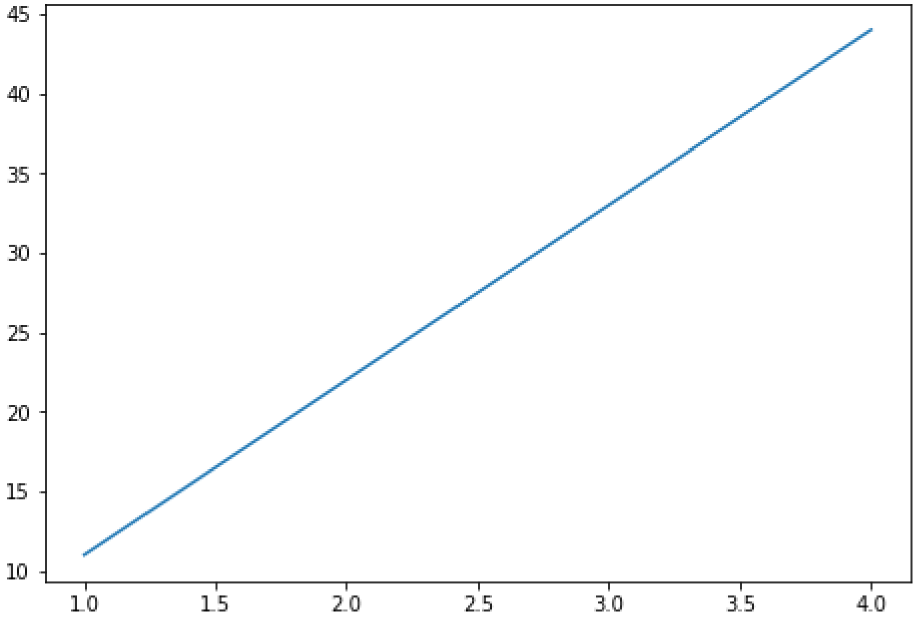
Out[5]:
```
[<matplotlib.lines.Line2D at 0x116578f10>]
```



In [6]:
```
# 1st method
fig=plt.figure() # creates a figure
ax=fig.add_subplot() # adds some axes
plt.show()
```



In [7]:
```
#2nd method
fig=plt.figure() # creates a figure
ax=fig.add_axes([1,1,1,1]) # add manual axes
ax.plot(x,y) # add some data
plt.show()
```

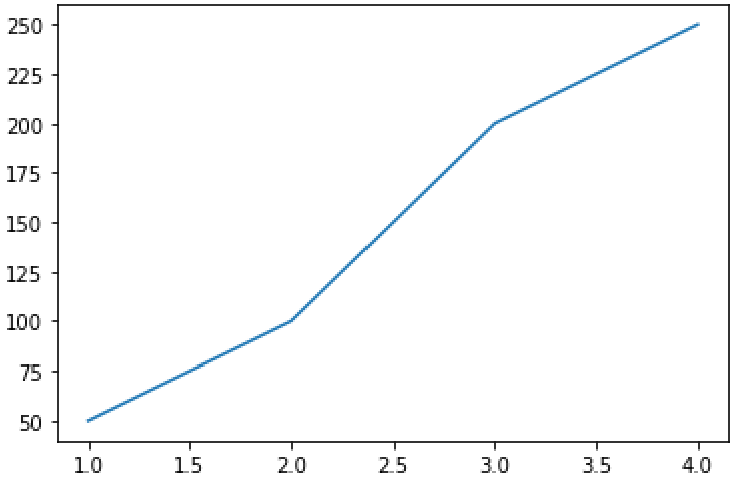


In [10]:

```
# 3rd method using subplots
fig, ax=plt.subplots()
ax.plot(x,[50,100,200,250])
```

Out[10]:

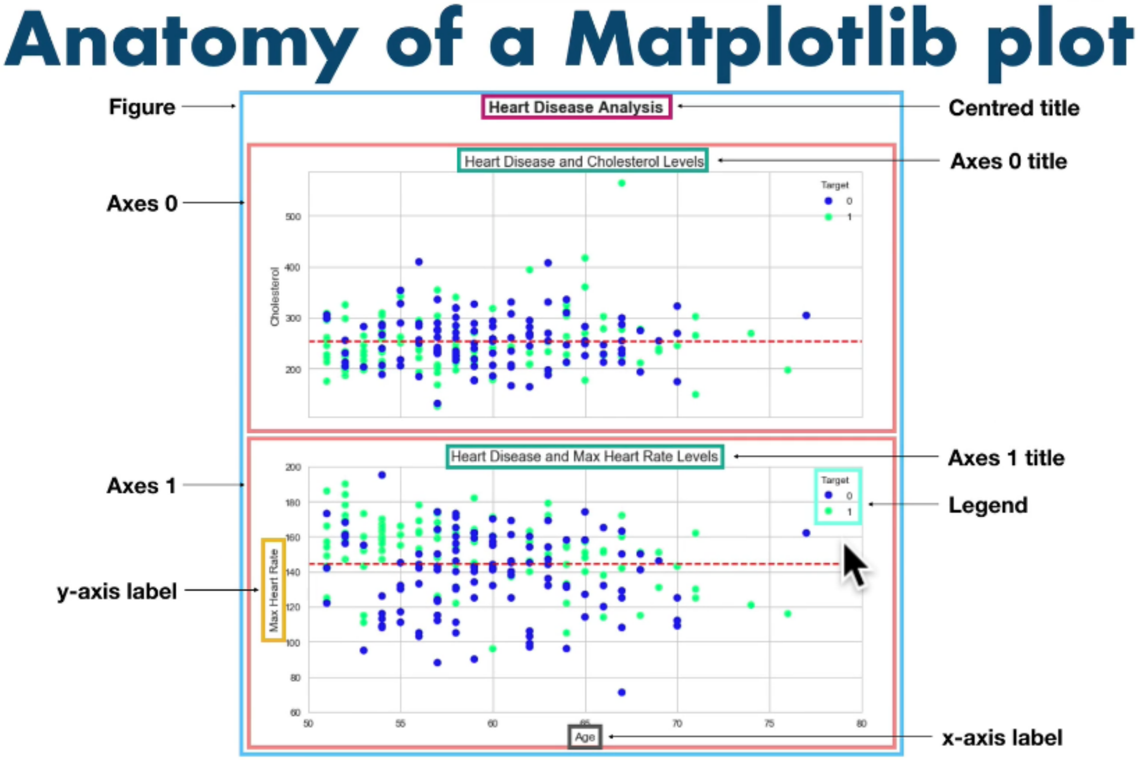
```
[<matplotlib.lines.Line2D at 0x11666bb50>]
```



In [19]:

```
from IPython.display import Image
Image(filename="label.png")
```

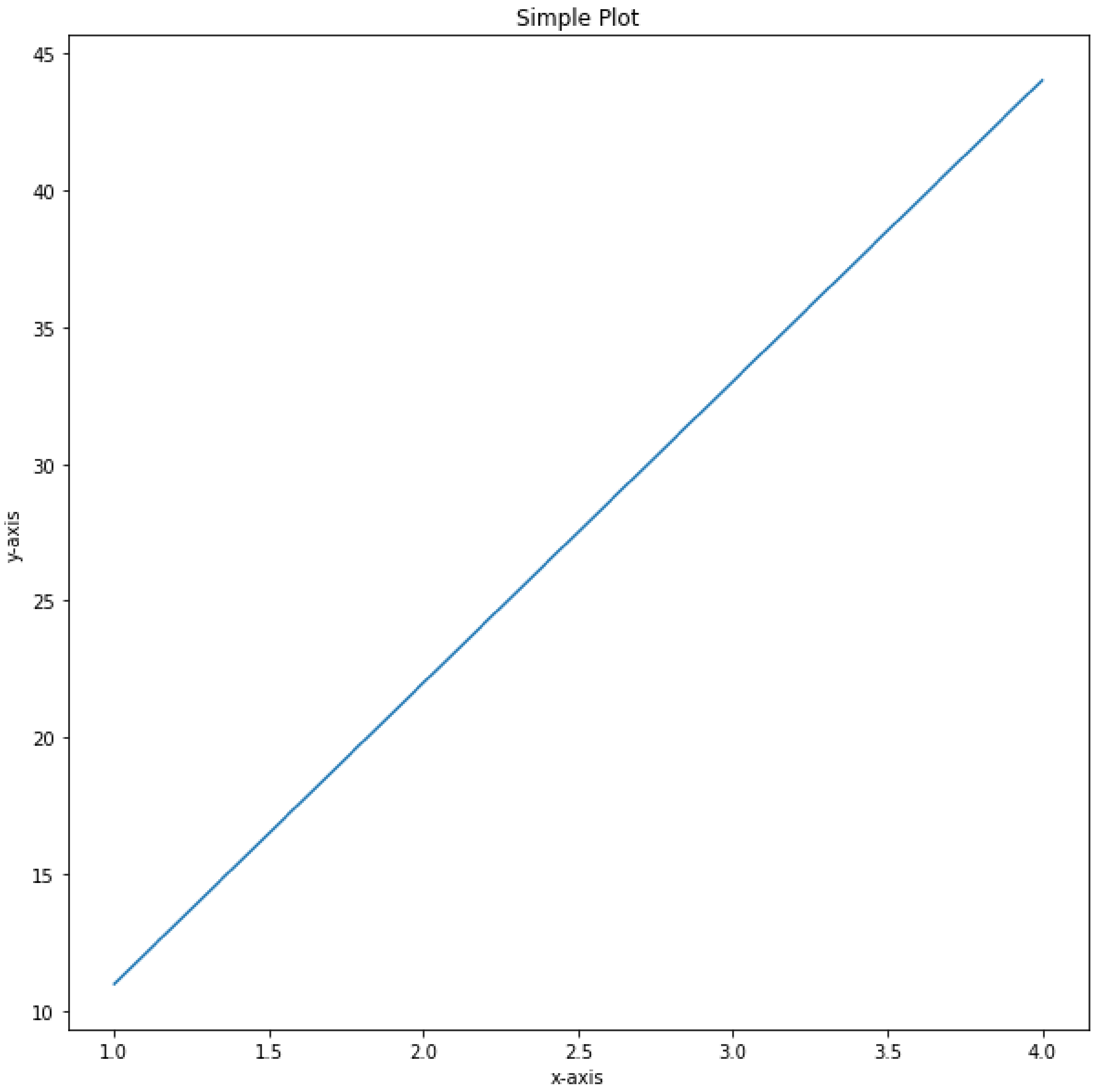
Out[19]:



Matpplotlib Example Workflow

In [21]:

```
#0. import matplotlib and get it ready for plotting in jupyter
%matplotlib inline
import matplotlib.pyplot as plt

#1. Prepare data
x=[1,2,3,4]
y=[11,22,33,44]

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

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

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

#5. Save and show
fig.savefig("../env/sample-plot.png")
```



Make Figures with NumPy Arrays

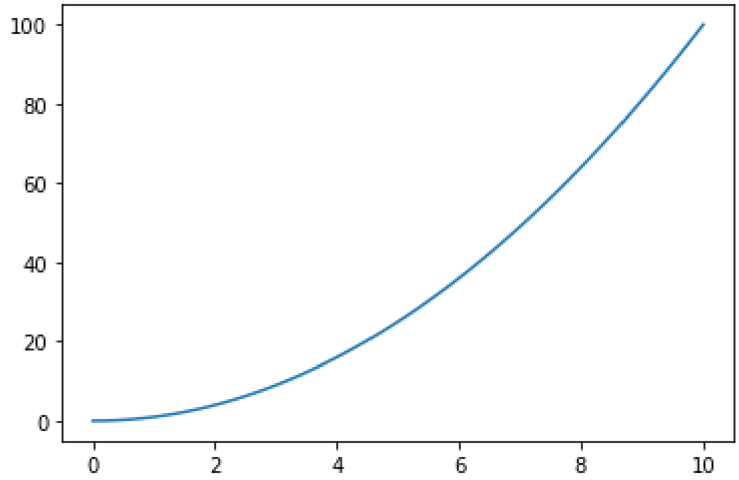
*Line plot, Scatter plot, Bar plot, Histogram, Subplots

In [26]:
```
#Create data
x=np.linspace(0,10,100)
x[:20]
```

Out[26]:
```
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])
```

In [27]:
```
#Line Plot the data
fig,ax=plt.subplots()
ax.plot(x,x**2)
```

Out[27]:
```
[<matplotlib.lines.Line2D at 0x116a3a6d0>]
```

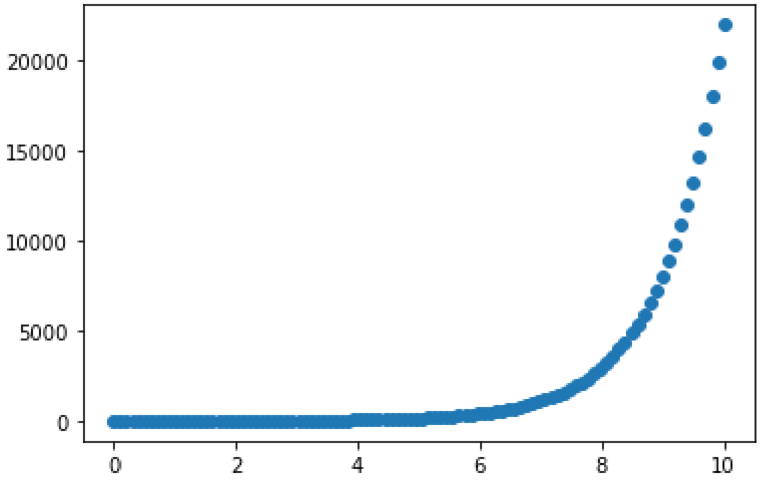


In [28]:

```
#Scatter plot the data
fig, ax = plt.subplots()
ax.scatter(x,np.exp(x))
```

Out[28]:

```
<matplotlib.collections.PathCollection at 0x1170e3590>
```

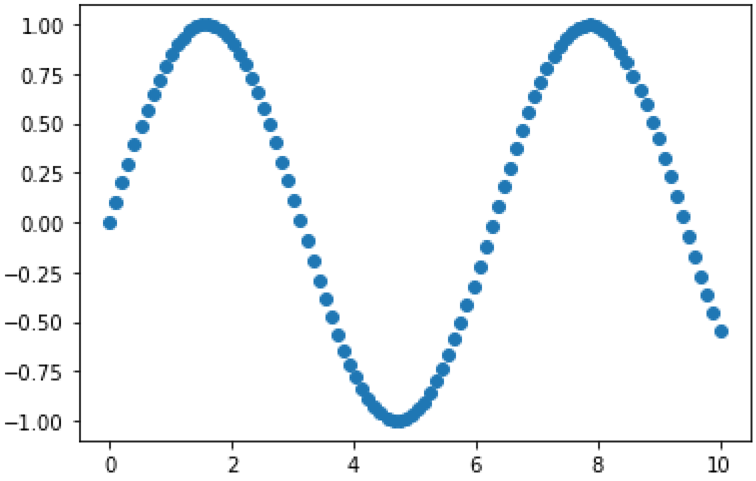


In [29]:

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

Out[29]:

```
<matplotlib.collections.PathCollection at 0x116afe890>
```

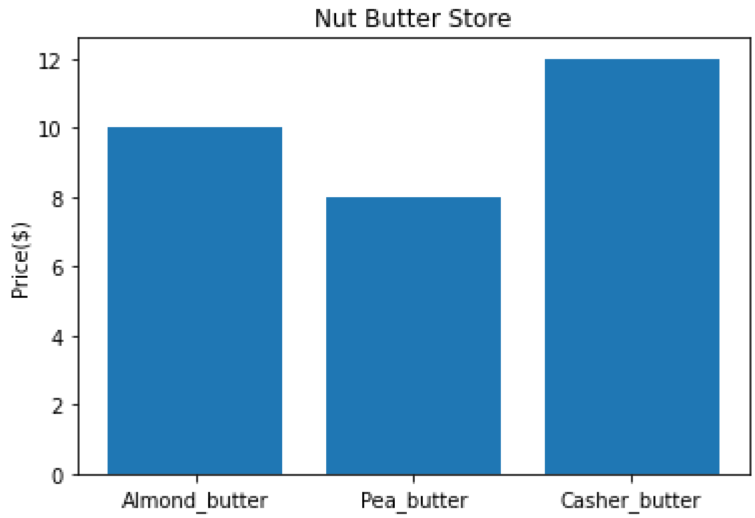


In [36]:

```
#Make a plot from dictionary
nut_butter_prices={"Almond_butter":10,
                   "Pea_butter":8,
                   "Casher_butter":12}
fig,ax=plt.subplots()
ax.bar(nut_butter_prices.keys(), nut_butter_prices.values()) # cannot wr
ite x = and y=, only use keys and values
ax.set(title="Nut Butter Store",ylabel="Price($)")
```

Out[36]:

```
[Text(0, 0.5, 'Price($)'), Text(0.5, 1.0, 'Nut Butter Store')]
```

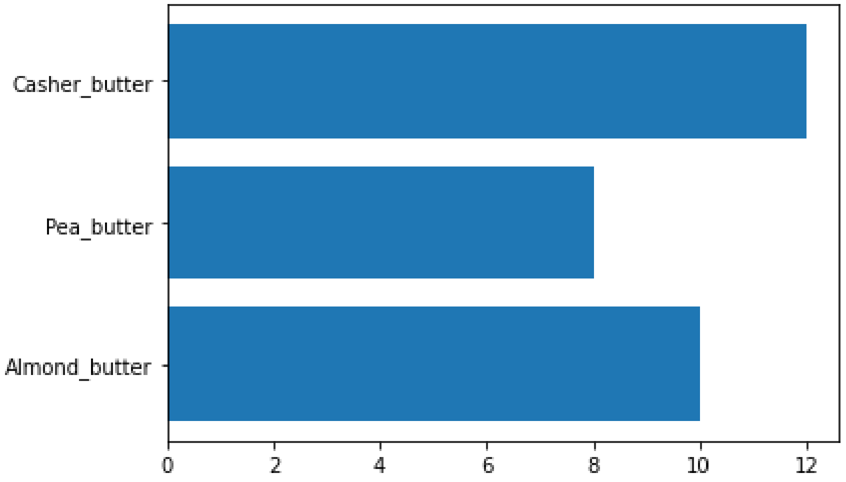


In [39]:

```
# use barh for horizontal bar
fig, ax=plt.subplots()
ax.barh(list(nut_butter_prices.keys()),list(nut_butter_prices.values()))
```

Out[39]:

```
<BarContainer object of 3 artists>
```



In [42]:

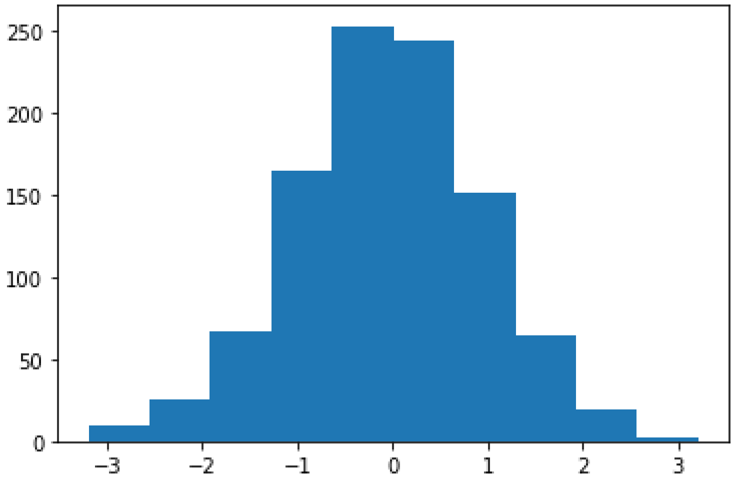
```
#Histgram - normal distribution
x=np.random.randn(1000)
x[0:10]
```

Out[42]:

```
array([ 0.61430745,  0.39826045,  0.56001557,  0.63465597,  1.06212589,
        0.04309148,  0.27076322, -1.01957777,  0.95035368, -0.1575151
2])
```

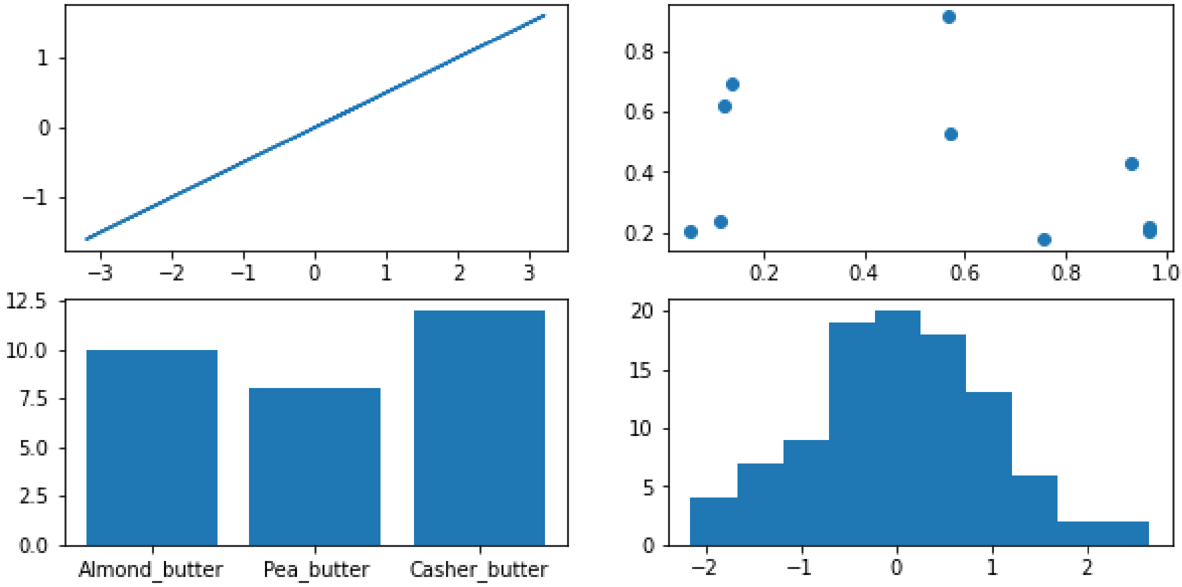
In [43]:
```
fig,ax =plt.subplots()
ax.hist(x) # normal distribution
```

Out[43]:
```
(array([  9.,  26.,  67., 165., 253., 244., 151.,  64.,  19.,   2.]),
 array([-3.19573406, -2.55458329, -1.91343252, -1.27228175, -0.6311309
7,
         0.0100198 ,  0.65117057,  1.29232135,  1.93347212,  2.5746228
9,
         3.21577367]),
 <a list of 10 Patch objects>)
```



In [50]:
```
#subplot option 1
fig, ((ax1,ax2),(ax3,ax4)) = plt.subplots(nrows=2,ncols=2,figsize=(10,5
))

#plot to each different axis
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(100))
```

Out[50]:
```
(array([ 4.,  7.,  9., 19., 20., 18., 13.,  6.,  2.,  2.]),
 array([-2.1519516 , -1.67119091, -1.19043021, -0.70966952, -0.2289088
3,
         0.25185186,  0.73261256,  1.21337325,  1.69413394,  2.1748946
4,
         2.65565533]),
 <a list of 10 Patch objects>)
```

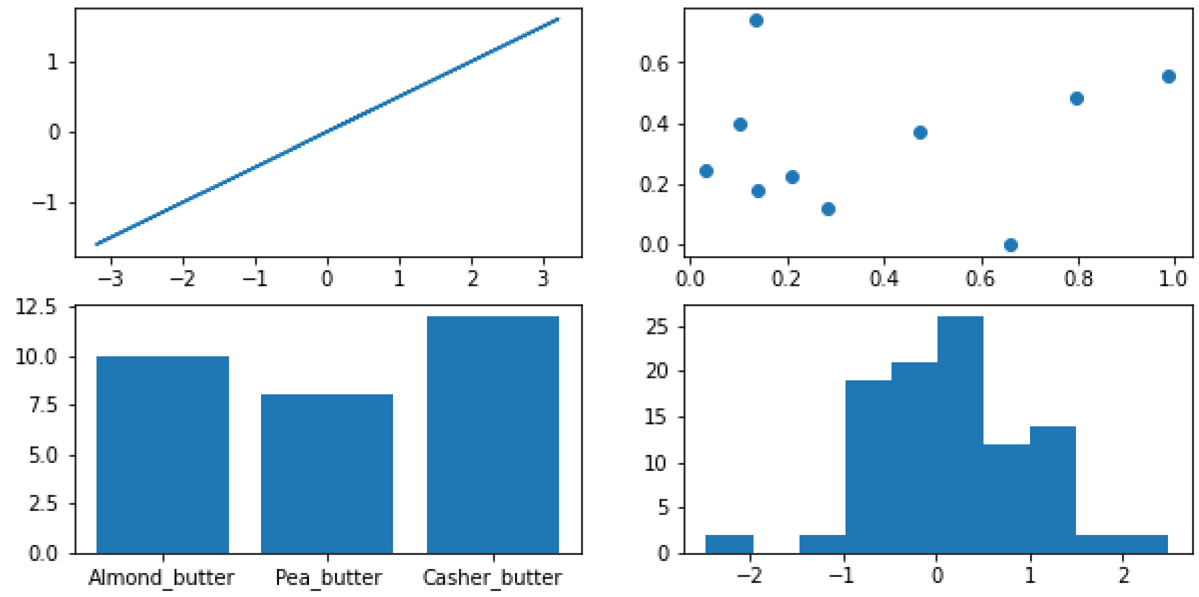


In [55]:

```
#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(100))
```

Out[55]:

```
(array([ 2.,  0.,  2., 19., 21., 26., 12., 14.,  2.,  2.]),
 array([-2.46306709, -1.96823277, -1.47339846, -0.97856414, -0.4837298
3,
         0.01110448,  0.5059388 ,  1.00077311,  1.49560743,  1.9904417
4,
         2.48527606]),
 <a list of 10 Patch objects>)
```



Plotting from Pandas DataFrames

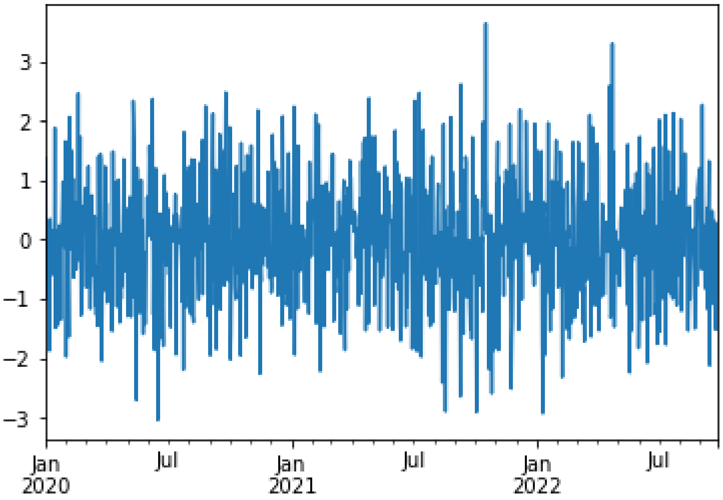
In [57]:
```
#make a dataframe
car_sales=pd.read_csv("car-sales.csv")
car_sales
```

Out[57]:

	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

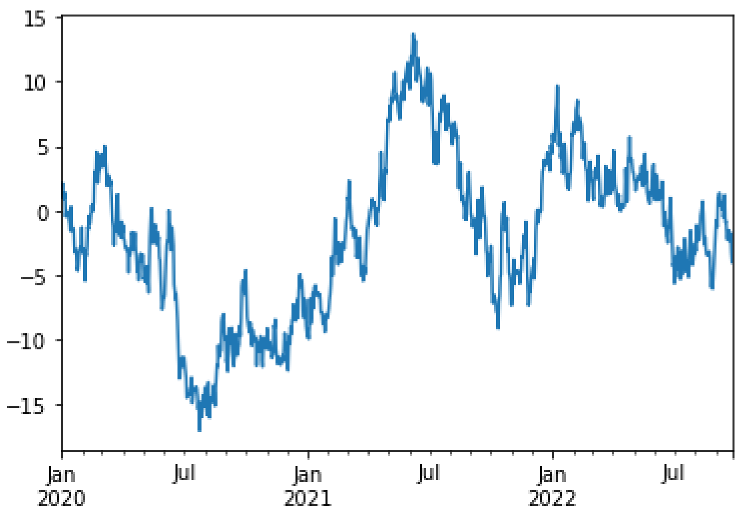
In [65]:
```
ts = pd.Series(np.random.randn(1000),index=pd.date_range("1/1/2020",peri
ods=1000))
ts.plot()
```

Out[65]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x118940490>
```



In [66]:
```
# ts.cumsum() adding rows, accumulated sum
ts=ts.cumsum()
ts.plot()
```

Out[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1189eb050>
```



In [71]:
```
#Using REGEX: https://regexone.com/
car_sales["Price"]=car_sales["Price"].str.replace('[\$\,\.]','')
type(car_sales["Price"][0])
```

Out[71]:
```
str
```

In [72]:
```
#Remove the last two digits from price
car_sales["Price"] = car_sales["Price"].str[:-2]
car_sales
```

Out[72]:

	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 [73]:

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

Out[73]:

	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 [74]:

```python
car_sales["Total Sales"]=car_sales["Price"].cumsum()
car_sales # why not accumulated sum? because sales price is a string, need to change to integer
```

Out[74]:

	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	40005000
2	Toyota	Blue	32549	3	7000	2020-01-03	400050007000
3	BMW	Black	11179	5	22000	2020-01-04	400050007000220000
4	Nissan	White	213095	4	3500	2020-01-05	4000500070002200003500
5	Toyota	Green	99213	4	4500	2020-01-06	40005000700022000035004500
6	Honda	Blue	45698	4	7500	2020-01-07	400050007000220000350045007500
7	Honda	Blue	54738	4	7000	2020-01-08	4000500070002200003500450075007000
8	Toyota	White	60000	4	6250	2020-01-09	40005000700022000035004500750070006250
9	Nissan	White	31600	4	9700	2020-01-10	400050007000220000350045007500700062509700

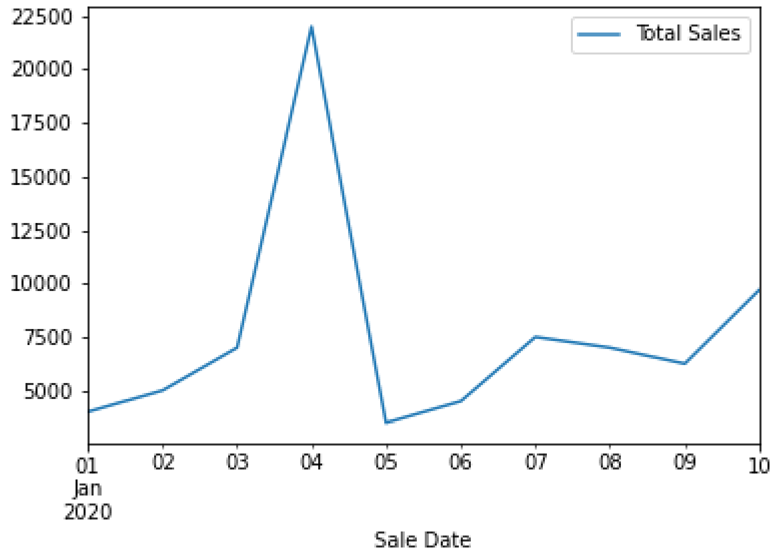
In [76]:
```
# how to add a total number?
car_sales["Total Sales"]=car_sales["Price"].astype(int).cumsum()  # didn't re-assign so astype won't change the "Price" type in later use
car_sales
```

Out[76]:

	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

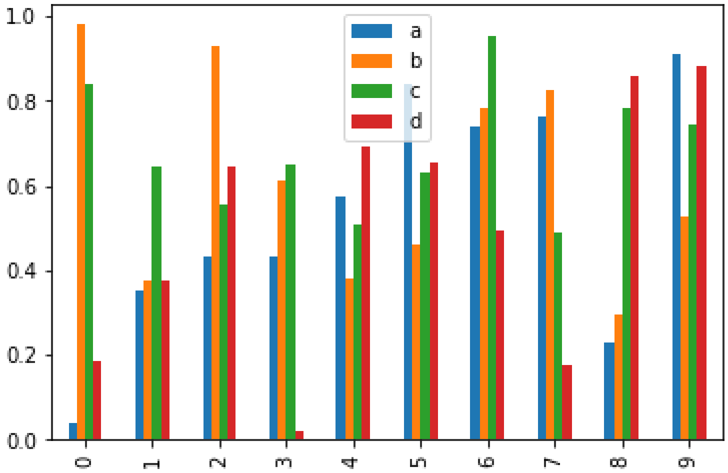
In [87]:
```
# re-assign the price type
car_sales["Price"]=car_sales["Price"].astype(int)
car_sales.plot(x="Sale Date",y="Total Sales")
```

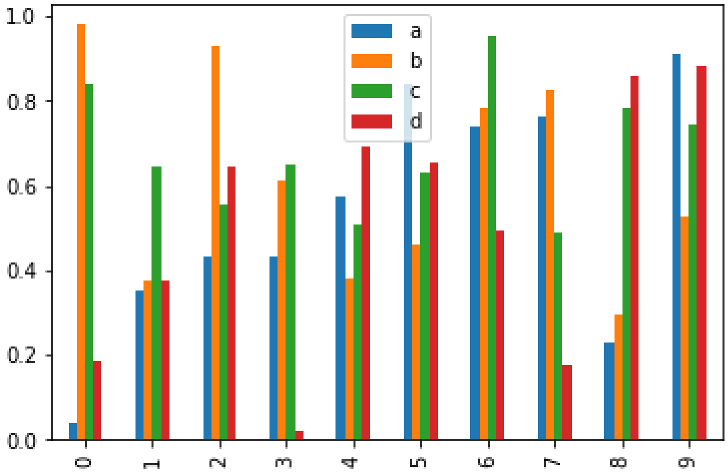
Out[87]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11982fb90>
```



In [88]:
```
# bar graph
x=np.random.rand(10,4) #10 rows and 4 columns
df=pd.DataFrame(x,columns=['a','b','c','d'])
df.plot.bar()
df.plot(kind="bar") # same as above, 2 approaches
```

Out[88]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x119a33f10>
```



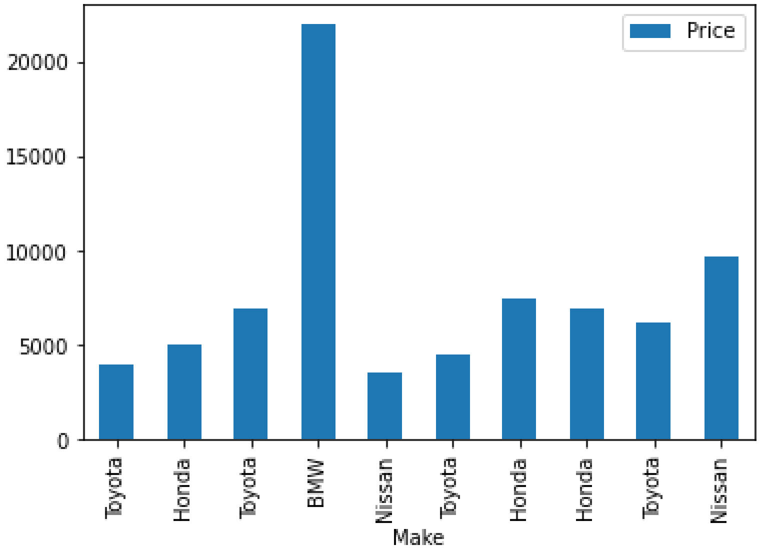


In [89]:

```
car_sales.plot(x="Make",y="Price",kind="bar")
```

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x119b241d0>
```

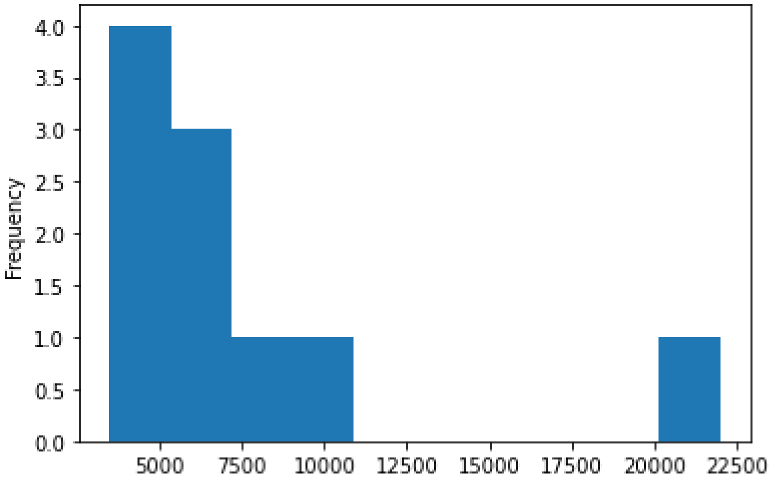


In [94]:

```
car_sales["Price"].plot(kind="hist")
```

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x119b6e4d0>
```

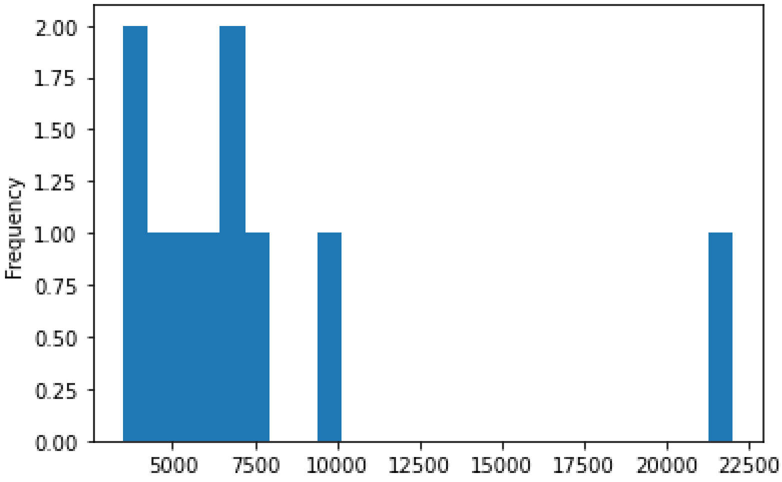


In [95]:

```
#change distribution bins
car_sales["Price"].plot.hist(bins=25)
```

Out[95]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x119eef490>
```



Practice: heart-disease.csv

In [97]:

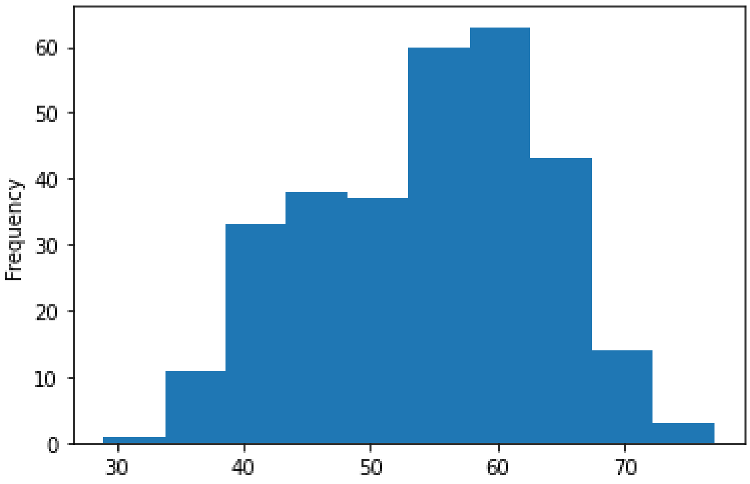
```
heart_disease=pd.read_csv("heart_disease.csv")
heart_disease.head()
```

Out[97]:

	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

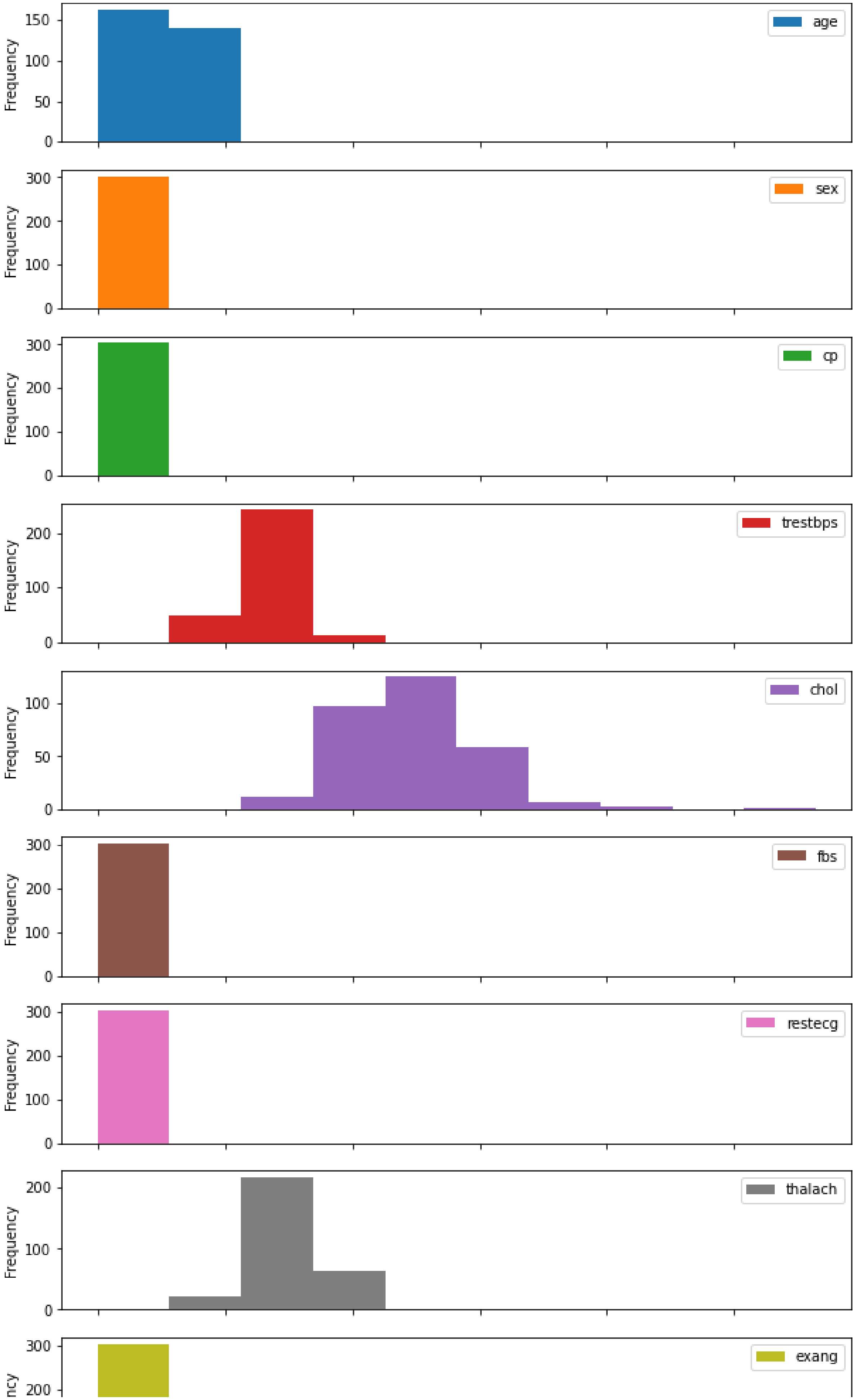
```
In [102]: heart_disease["age"].plot.hist(bins=10) # the more bins, the more precis
          e grouping it is
```

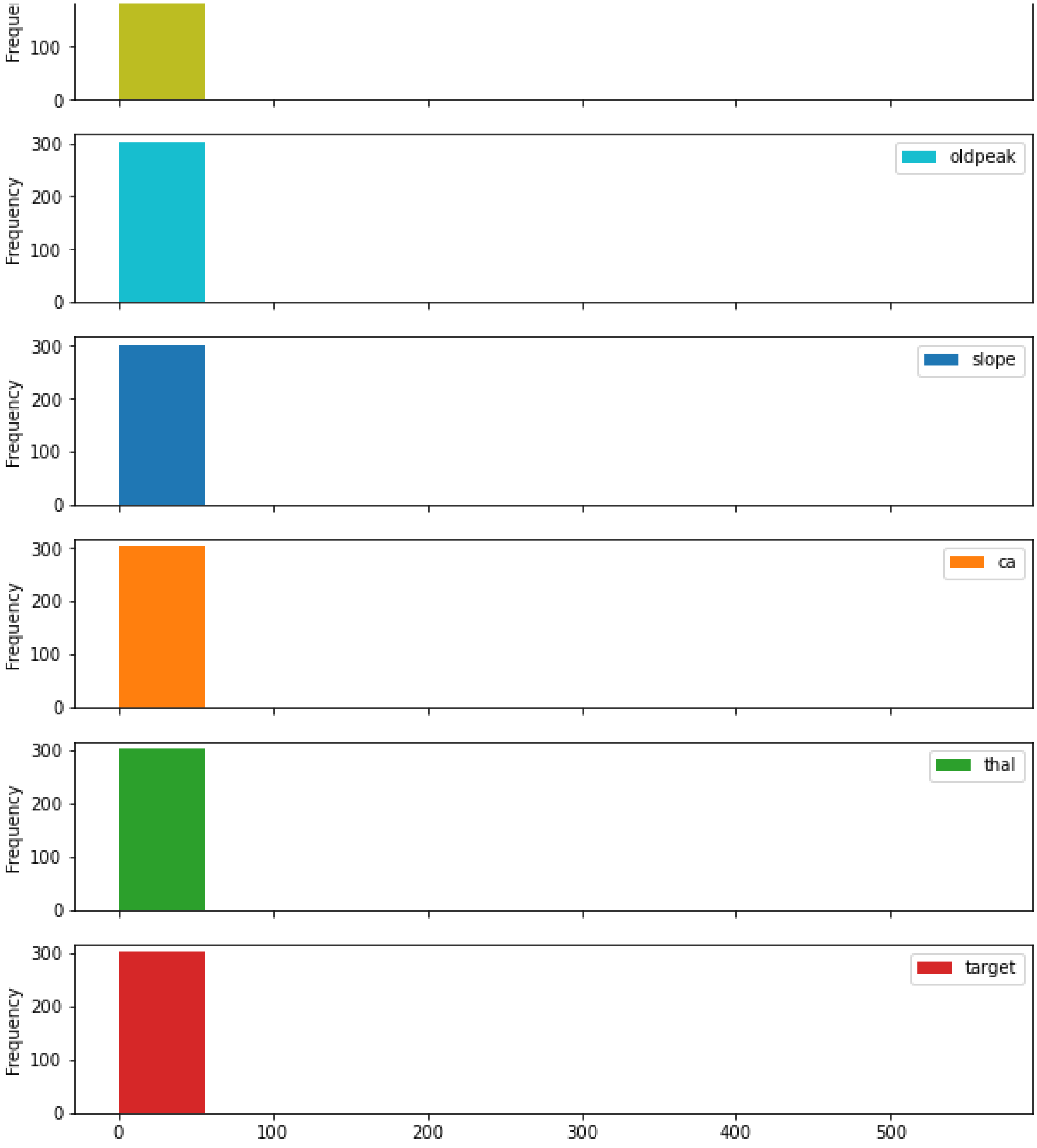
```
Out[102]: <matplotlib.axes._subplots.AxesSubplot at 0x119c205d0>
```



```
In [105]: heart_disease.plot.hist(figsize =(10,30),subplots=True)
```

```
Out[105]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x11a203d90>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11a98efd0>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11aa40e90>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11ab8ca90>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11abc1e50>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11ac06250>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11ac3d690>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11ac73990>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11ac739d0>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11acaae90>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11ad26550>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11ad5d910>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11ad95cd0>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x11adda0d0>],
                dtype=object)
```


Frequency
150
100
50
0
age
Frequency
300
200
100
0
sex
Frequency
300
200
100
0
cp
Frequency
200
100
0
trestbps
Frequency
100
50
0
chol
Frequency
300
200
100
0
fbs
Frequency
300
200
100
0
restecg
Frequency
200
100
0
thalach
ncy
300
200
exang



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

when plot quickly, use pyplot; when plot more advanced, use OO method

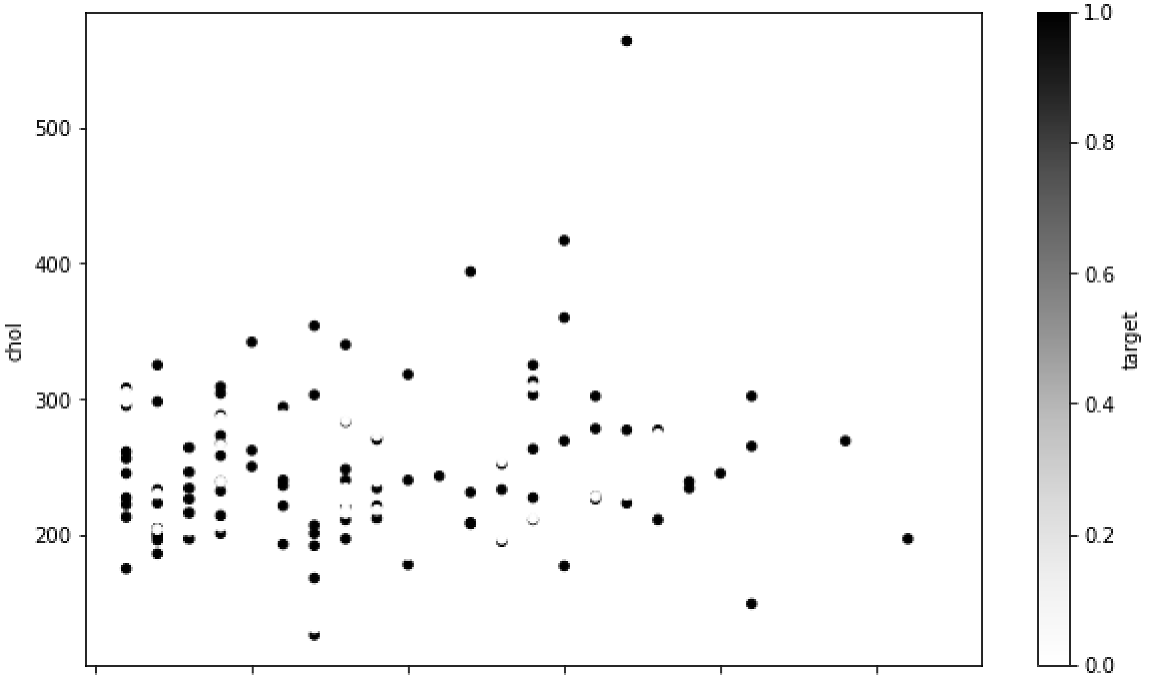
In [106]:
```
over_50=heart_disease[heart_disease["age"]>50]
over_50.head()
```

Out[106]:

	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
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
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1

In [110]:
```
#Pyplot Method
over_50.plot(kind='scatter',x='age',y='chol', c='target',figsize=(10,6))
```

Out[110]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11bd88750>
```

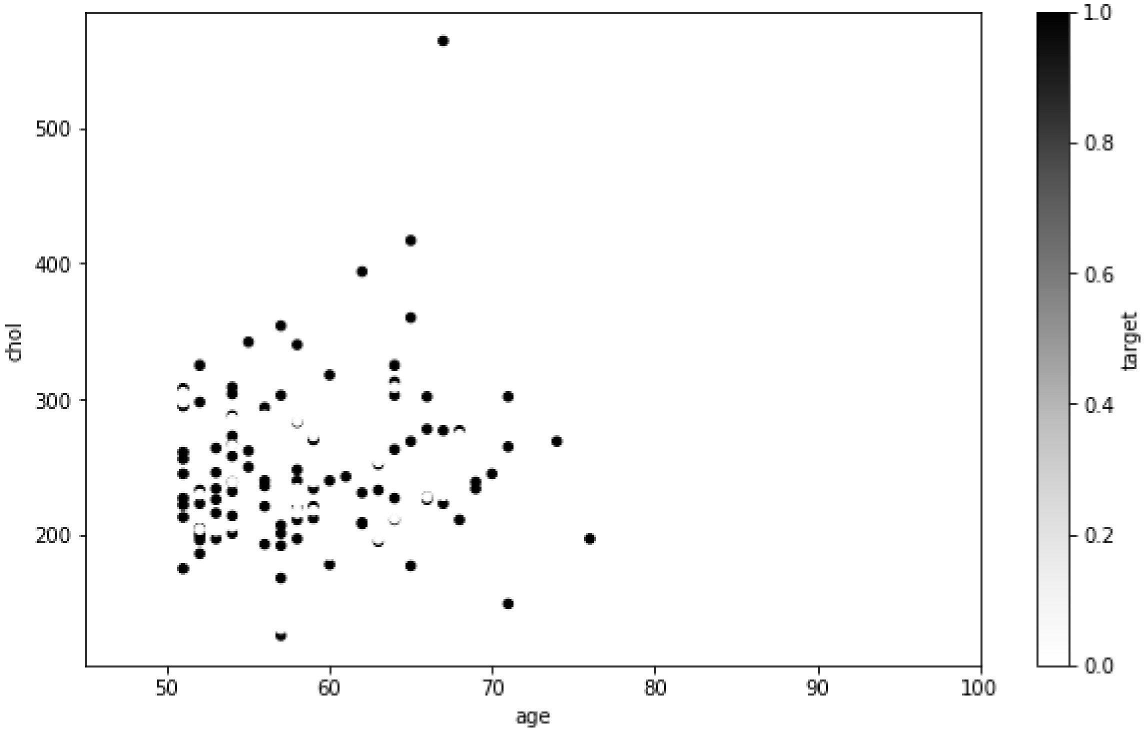


In [111]:

```
#Object Oriented Method mix with PyPlot
fix, ax=plt.subplots(figsize=(10,6))
over_50.plot(kind='scatter',x='age',y='chol',c='target',ax=ax)
ax.set_xlim([45,100]) # narrow the axis
```

Out[111]:

```
(45.0, 100.0)
```

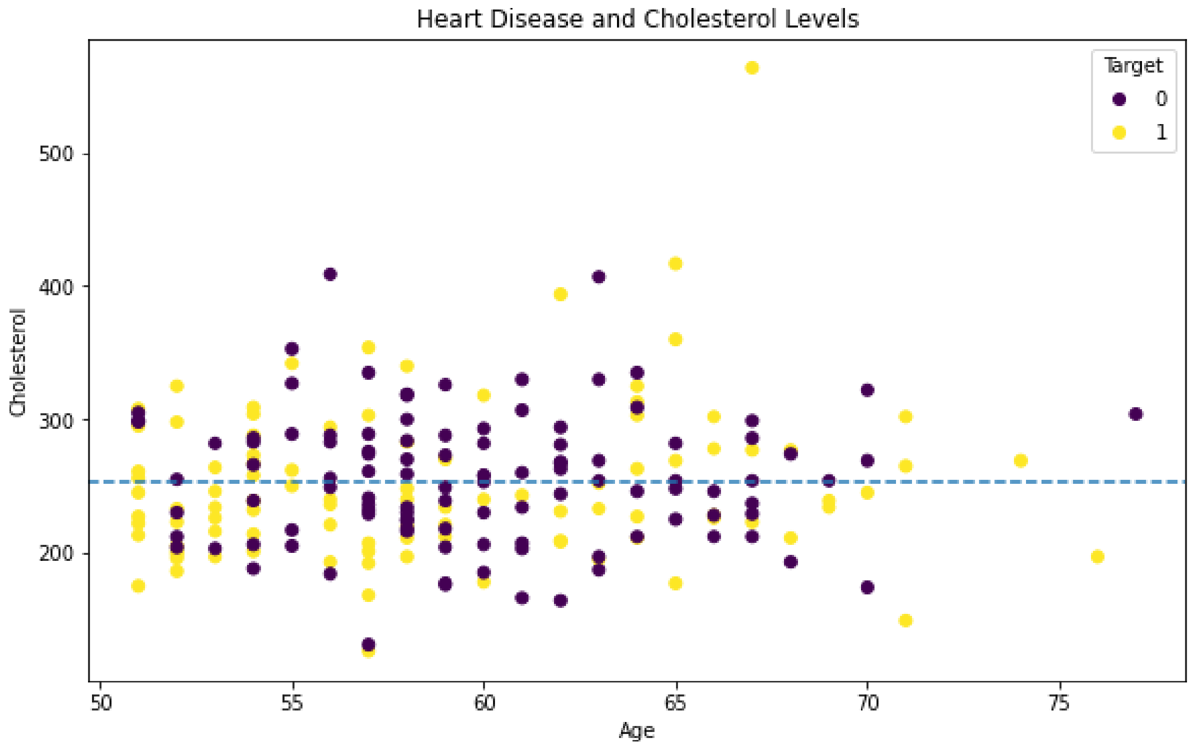


In [125]:

```
## OO method from scrach
fix, 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 legend
ax.legend(*scatter.legend_elements(),title="Target") #set up a legend fo
r c

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

Out[125]:

```
<matplotlib.lines.Line2D at 0x11c5c5950>
```



In [126]:

```
over_50.head()
```

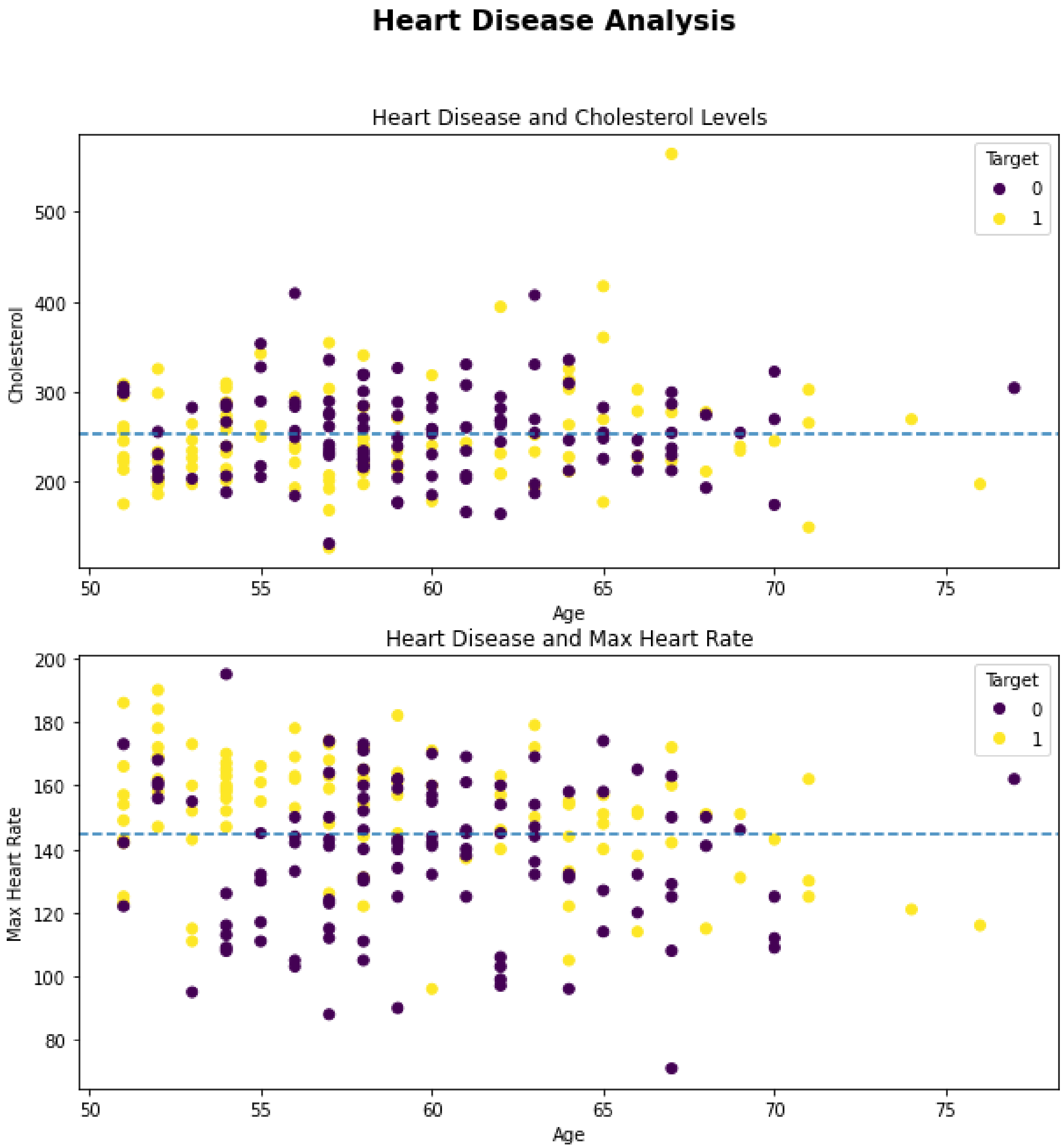
Out[126]:

	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
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
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1

In [144]:

```
#subplot of chol, age, thalach
fig,(ax0,ax1) = plt.subplots(nrows=2,ncols=1, figsize=(10,10))
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', ylabel='Cholesterol')

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

#add a meanline
ax0.axhline(y=over_50["chol"].mean(),linestyle="--")

#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 legend to ax1
ax1.legend(*scatter.legend_elements(),title="Target")

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

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

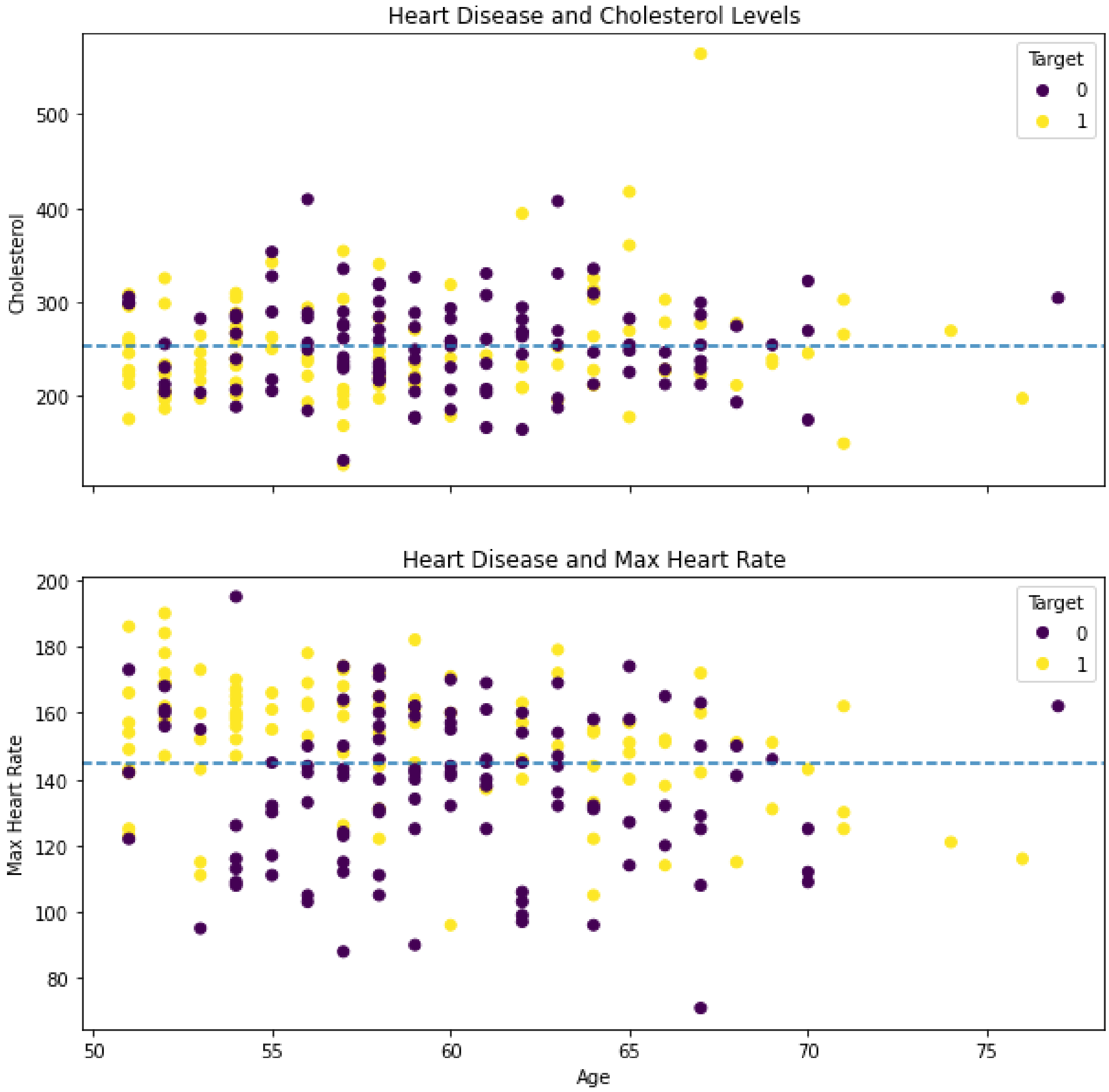
```
Out[144]: Text(0.5, 0.98, 'Heart Disease Analysis')
```



In [143]:

```
## How to share one axis at the bottom? - same code as above, comment changes

fig,(ax0,ax1) = plt.subplots(nrows=2,ncols=1, figsize=(10,10),sharex=True)  # add sharex=True
scatter=ax0.scatter(x=over_50["age"],y=over_50["chol"],c=over_50["target"])


ax0.set(title="Heart Disease and Cholesterol Levels", ylabel='Cholesterol') # remove the x=age in the ax0


ax0.legend(*scatter.legend_elements(),title="Target")


ax0.axhline(y=over_50["chol"].mean(),linestyle="--")


scatter=ax1.scatter(x=over_50['age'],y=over_50['thalach'],c=over_50['target'])


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


ax1.legend(*scatter.legend_elements(),title="Target")


ax1.axhline(y=over_50['thalach'].mean(),linestyle='--')
```

```
Out[143]: <matplotlib.lines.Line2D at 0x11f7f5fd0>
```



Customization of Plots Styles

In [145]:

```
#What styles are available?
plt.style.available
```

Out[145]:

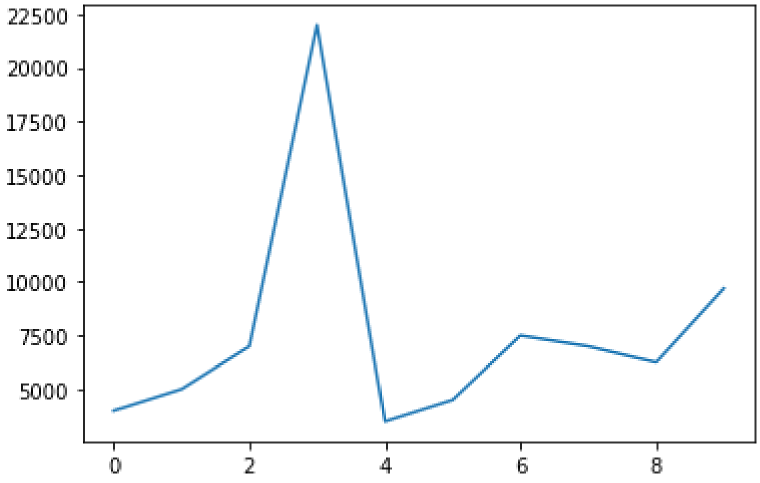
```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',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

In [147]:

```
car_sales["Price"].plot()
```

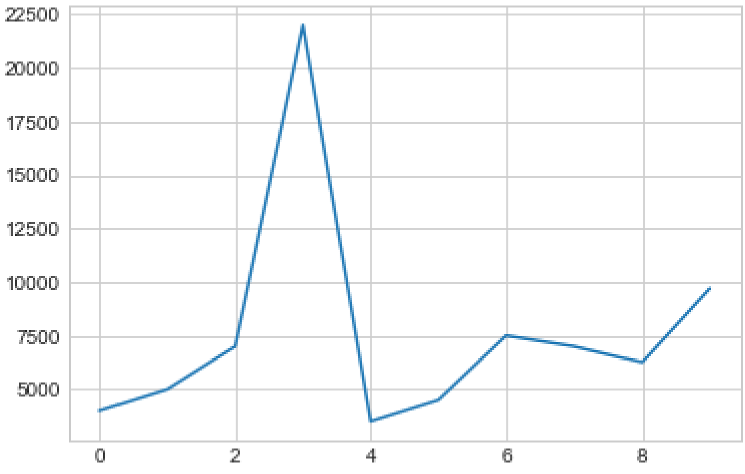
Out[147]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11fadc590>
```



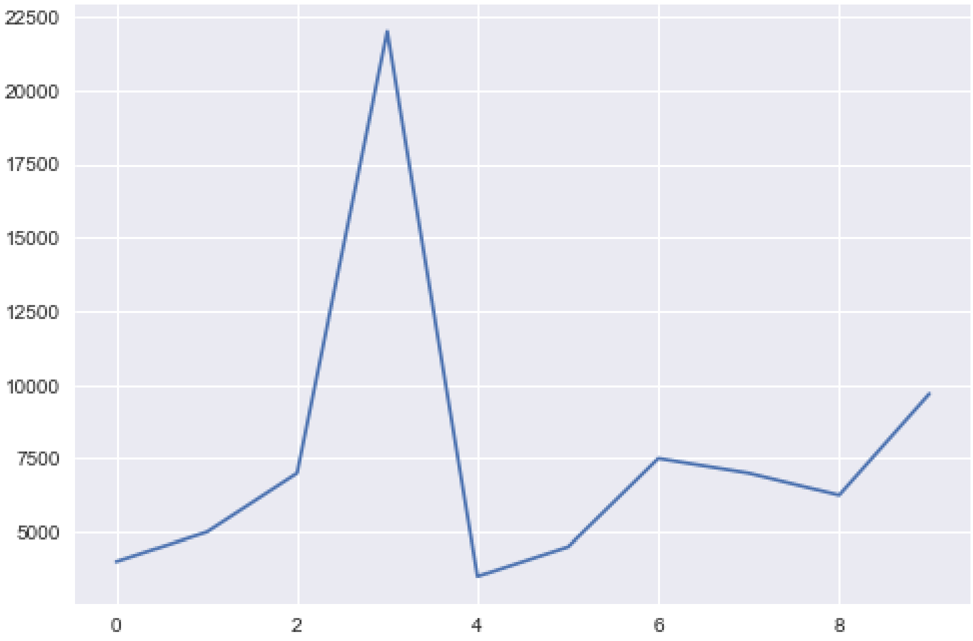
```
In [150]: plt.style.use('seaborn-whitegrid')
          car_sales["Price"].plot()
```

```
Out[150]: <matplotlib.axes._subplots.AxesSubplot at 0x11d789d90>
```



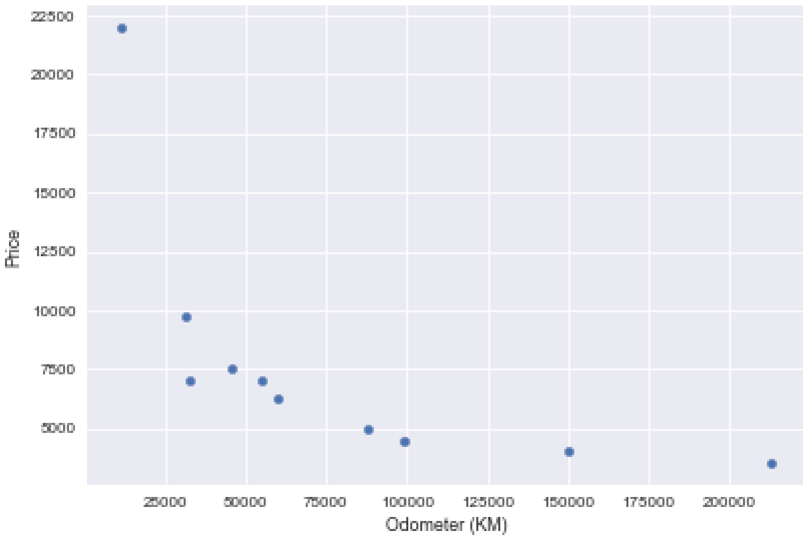
```
In [151]: plt.style.use('seaborn')
          car_sales["Price"].plot()
```

```
Out[151]: <matplotlib.axes._subplots.AxesSubplot at 0x11e21c190>
```



In [155]:
```
plt.style.use('seaborn-paper')
car_sales.plot(x="Odometer (KM)", y="Price", kind="scatter")
```

Out[155]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11eb23e10>
```



In [156]:
```
#create data
x=np.random.randn(10,4)
x
```

Out[156]:
```
array([[-0.49010756, -0.91078814, -0.65665894,  0.3616924 ],
       [ 1.51812785, -1.64945007,  2.36129995, -0.71282619],
       [-0.20869268, -0.90125836, -0.61503112,  0.22644629],
       [ 0.40994387,  0.04486931, -1.47825941, -0.14885771],
       [-0.99691935,  0.1560044 ,  0.58091192,  1.66523667],
       [-1.04117363, -0.63105782,  0.03962856,  1.04024624],
       [ 1.27646758, -1.10128692, -0.63668165, -0.80587473],
       [ 0.62775285,  1.98998486,  1.338314  , -0.23576596],
       [ 1.58547927,  0.9713256 ,  0.52774973, -0.22667659],
       [-2.16767903,  0.83576231, -0.23758773,  0.83787207]])
```

In [160]:

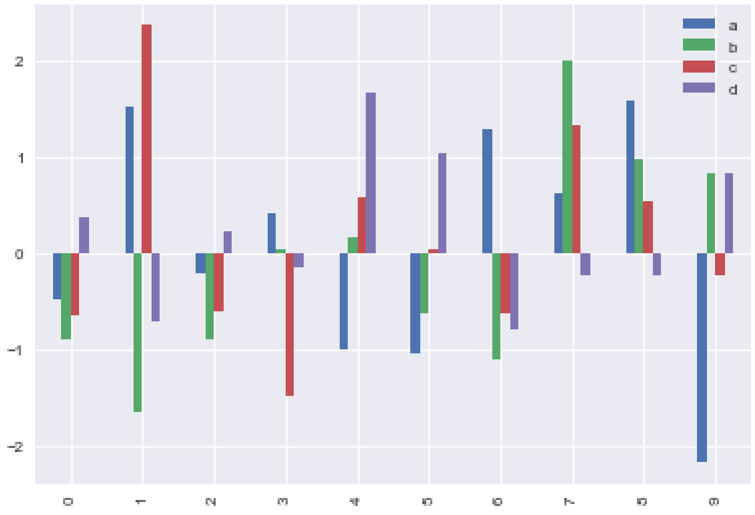
```
df=pd.DataFrame(x,columns=['a','b','c','d'])
df
```

Out[160]:

	a	b	c	d
0	-0.490108	-0.910788	-0.656659	0.361692
1	1.518128	-1.649450	2.361300	-0.712826
2	-0.208693	-0.901258	-0.615031	0.226446
3	0.409944	0.044869	-1.478259	-0.148858
4	-0.996919	0.156004	0.580912	1.665237
5	-1.041174	-0.631058	0.039629	1.040246
6	1.276468	-1.101287	-0.636682	-0.805875
7	0.627753	1.989985	1.338314	-0.235766
8	1.585479	0.971326	0.527750	-0.226677
9	-2.167679	0.835762	-0.237588	0.837872

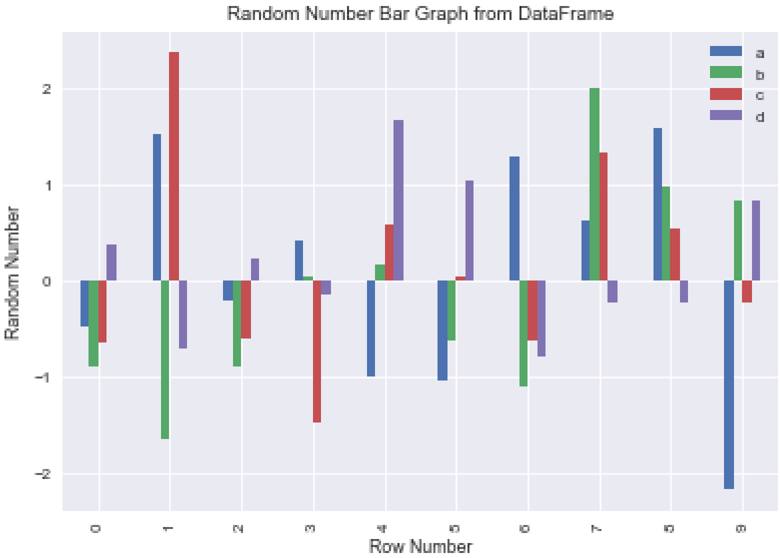
In [161]:

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



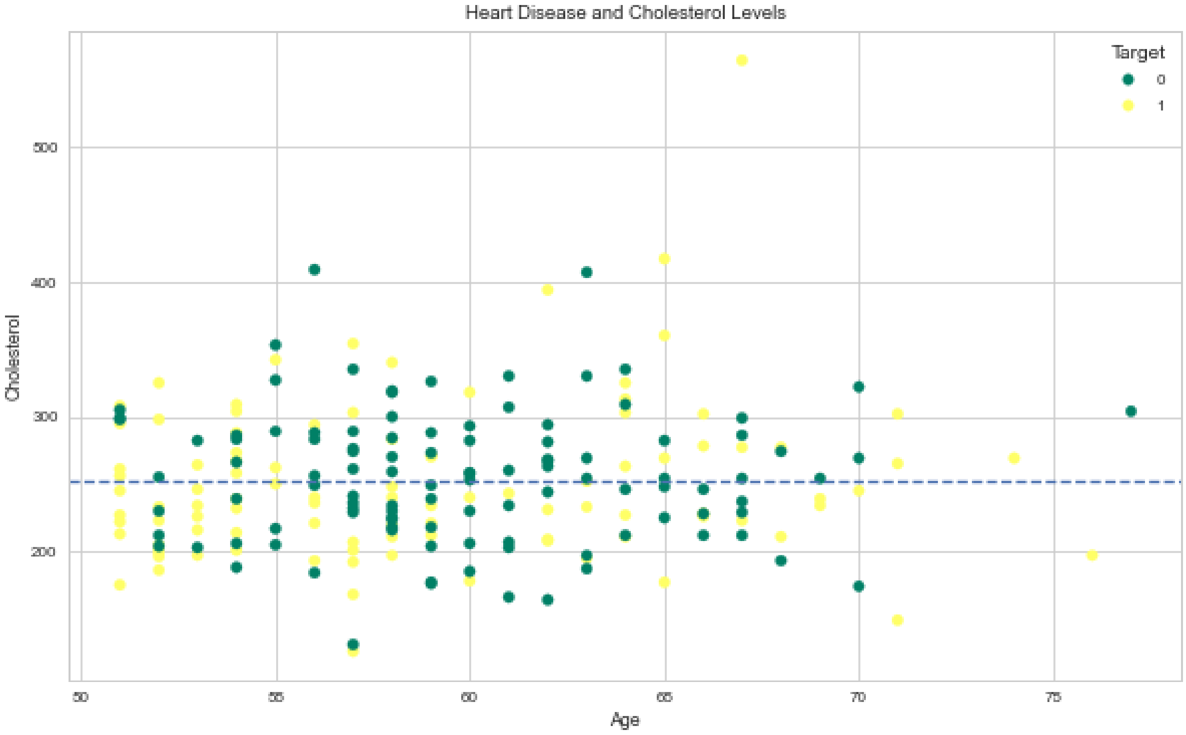
In [166]:

```
#Customize out 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 Numbe
r",ylabel='Random Number')
ax.legend().set_visible(True) # legend has () after it
```



```
In [172]: ## same code as above "OO method from scrach" changed commented
          plt.style.use('seaborn-whitegrid') #change style

          fix, ax=plt.subplots(figsize=(10,6))

          #cmap can change color theme, check onlien for available
          scatter=ax.scatter(x=over_50['age'],y=over_50['chol'],c=over_50['target'
          ],cmap="summer")
          ax.set(title="Heart Disease and Cholesterol Levels",
                 xlabel="Age",
                 ylabel="Cholesterol")

          ax.legend(*scatter.legend_elements(),title="Target")

          ax.axhline(over_50['chol'].mean(),linestyle='--')
```

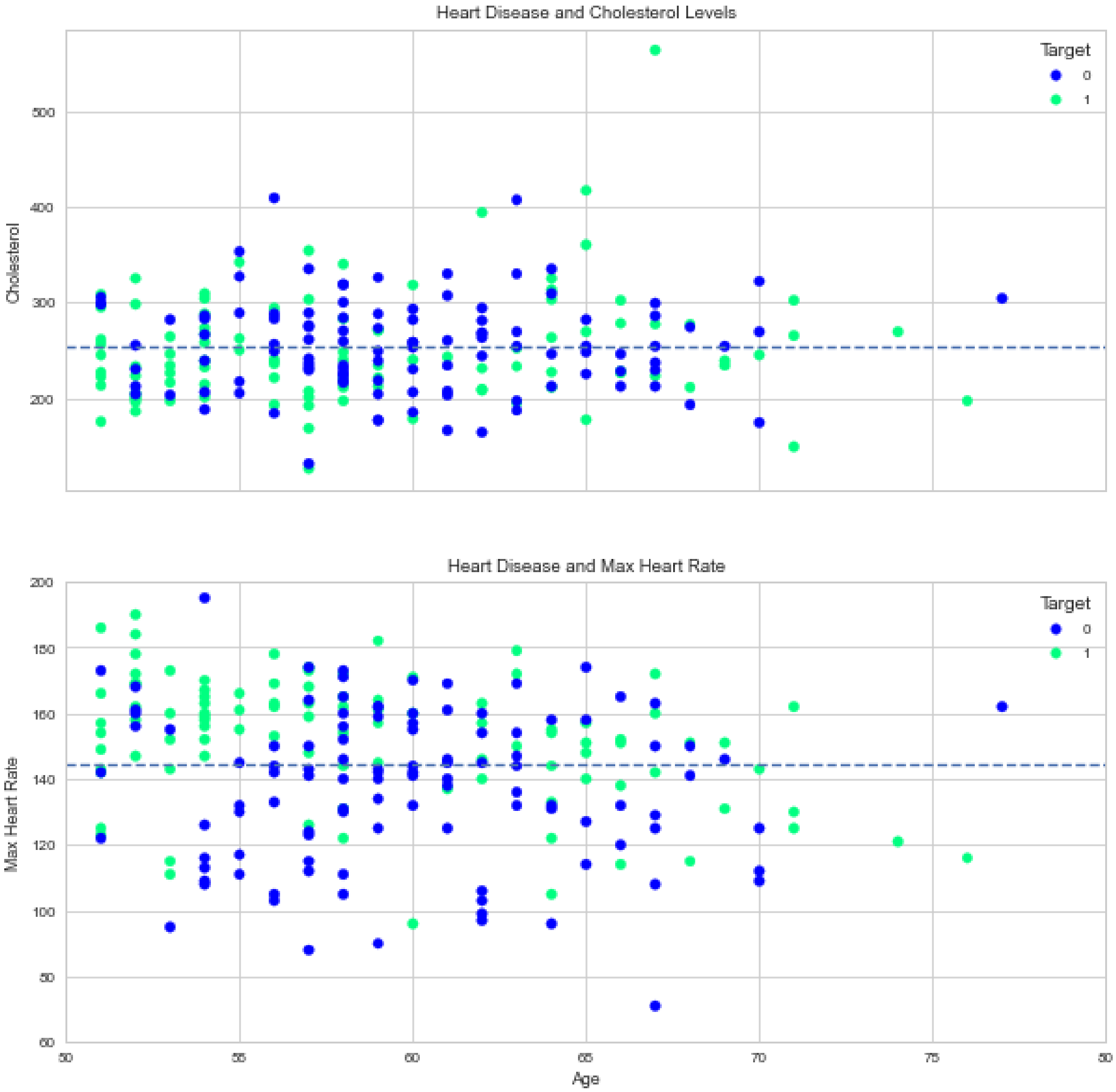
```
Out[172]: <matplotlib.lines.Line2D at 0x11cbe2790>
```



In [180]:

```
## same code as above, changed commented
plt.style.use('seaborn-whitegrid') #change style

fig,(ax0,ax1) = plt.subplots(nrows=2,ncols=1, figsize=(10,10),sharex=True)  # add sharex=True

scatter=ax0.scatter(x=over_50["age"],y=over_50["chol"],c=over_50["target"],cmap="winter")
ax0.set(title="Heart Disease and Cholesterol Levels", ylabel='Cholesterol') # remove the x=age in the ax0
ax0.set_xlim([50,80])# these can remove the double lines at the edge, see above

ax0.legend(*scatter.legend_elements(),title="Target")


ax0.axhline(y=over_50["chol"].mean(),linestyle="--")


scatter=ax1.scatter(x=over_50['age'],y=over_50['thalach'],c=over_50['target'],cmap = "winter")


ax1.set(title="Heart Disease and Max Heart Rate",xlabel="Age",ylabel="Max Heart Rate")
ax1.set_xlim([50,80]) # these can remove the double lines at the edge
ax1.set_ylim([60,200])# these can remove the double lines at the edge

ax1.legend(*scatter.legend_elements(),title="Target")


ax1.axhline(y=over_50['thalach'].mean(),linestyle='--')
```

```
Out[180]: <matplotlib.lines.Line2D at 0x11d7dcd10>
```



```
In [181]: #how to explore the figures?
          # - download the images
          # - save fig method
          fig.savefig("heart-disease.png")
```

```
In [ ]:
```