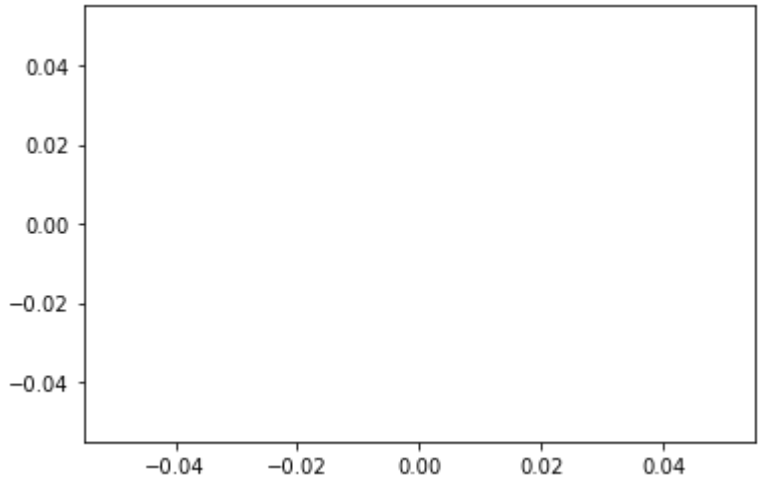
Matplotlib

Matplotlib example workflow

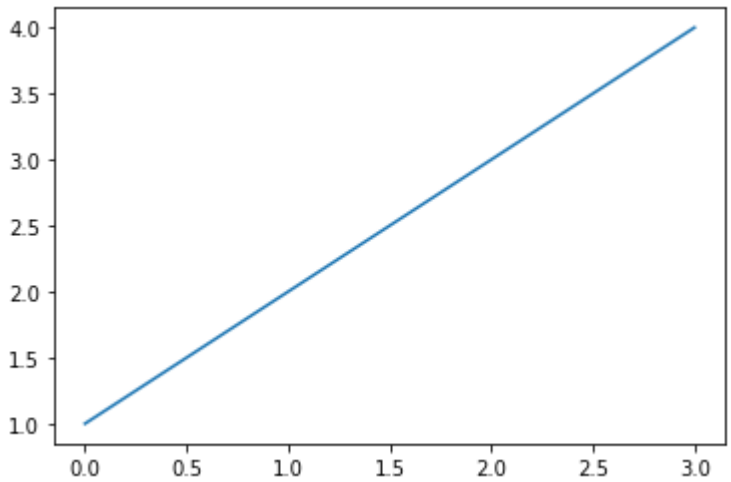
1. Import matplotlib and get it ready for plotting in Jupyter

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

In [2]:
```
plt.plot(); # Put semicolon so there is no op bracket or use plt.show()
```

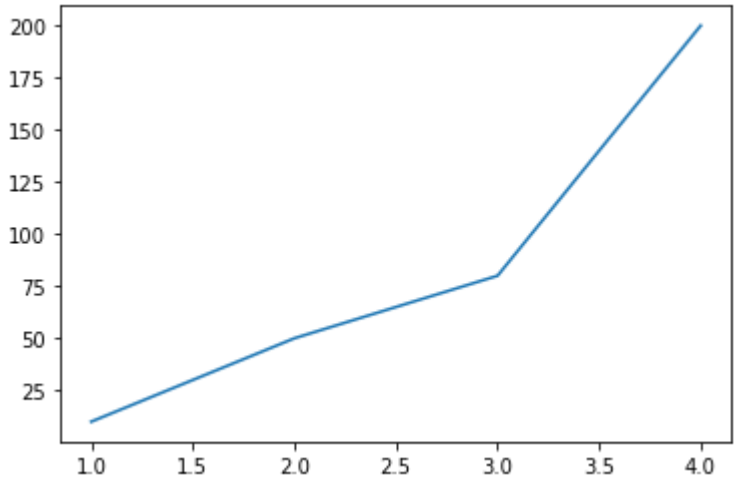


In [3]:
```
plt.plot([1,2,3,4]);
```



2. Prepare data

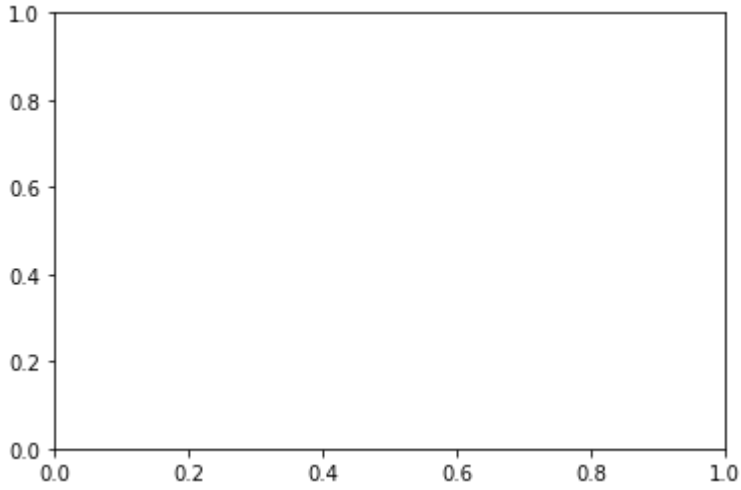
In [4]:
```
# Step 2
x = [1,2,3,4]
y = [10,50,80,200]
plt.plot(x,y); # Stateless plotting
```



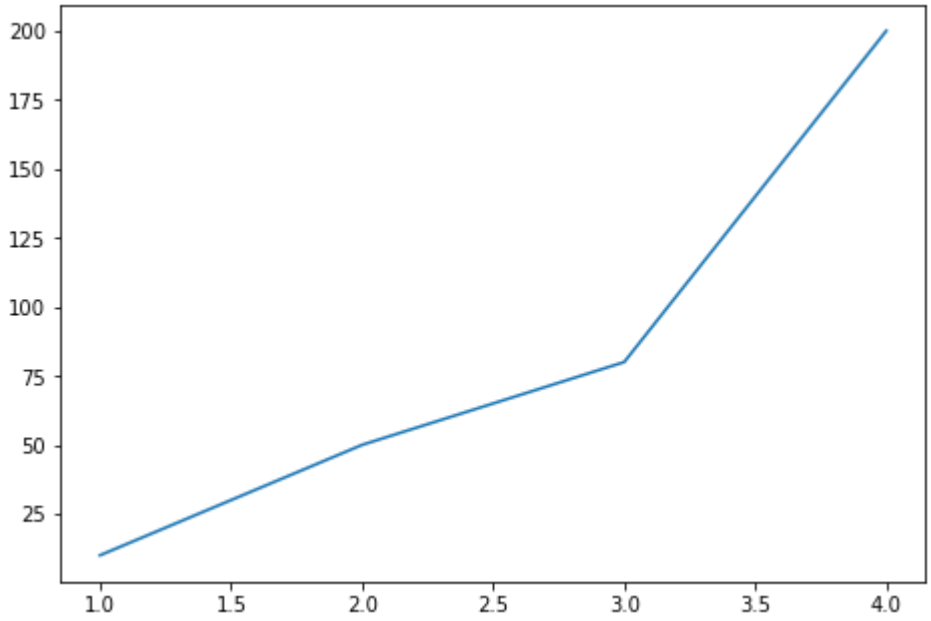
In general, object-oriented interface is used over the pyplot interface.

3. Setup plot & plot data

In [5]:
```
# 1st Method of plotting
fig = plt.figure() # creates a figure
ax = fig.add_subplot() # adds some axes
plt.show()
```

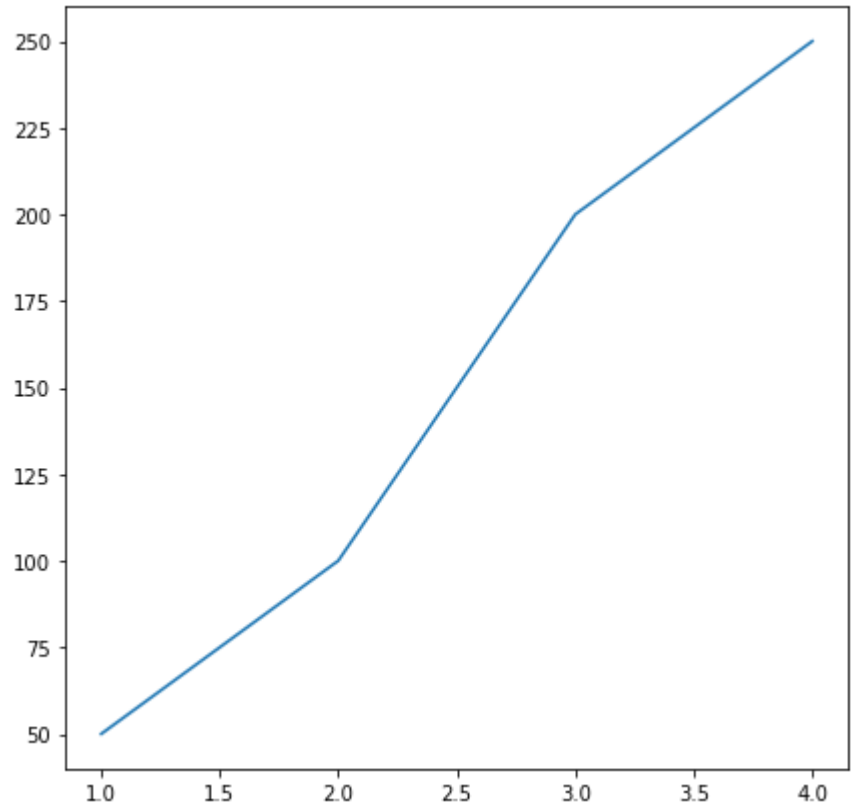


In [6]:
```
# 2nd method
fig = plt.figure()
ax = fig.add_axes([1,1,1,1])
ax.plot(x, y)
plt.show()
```



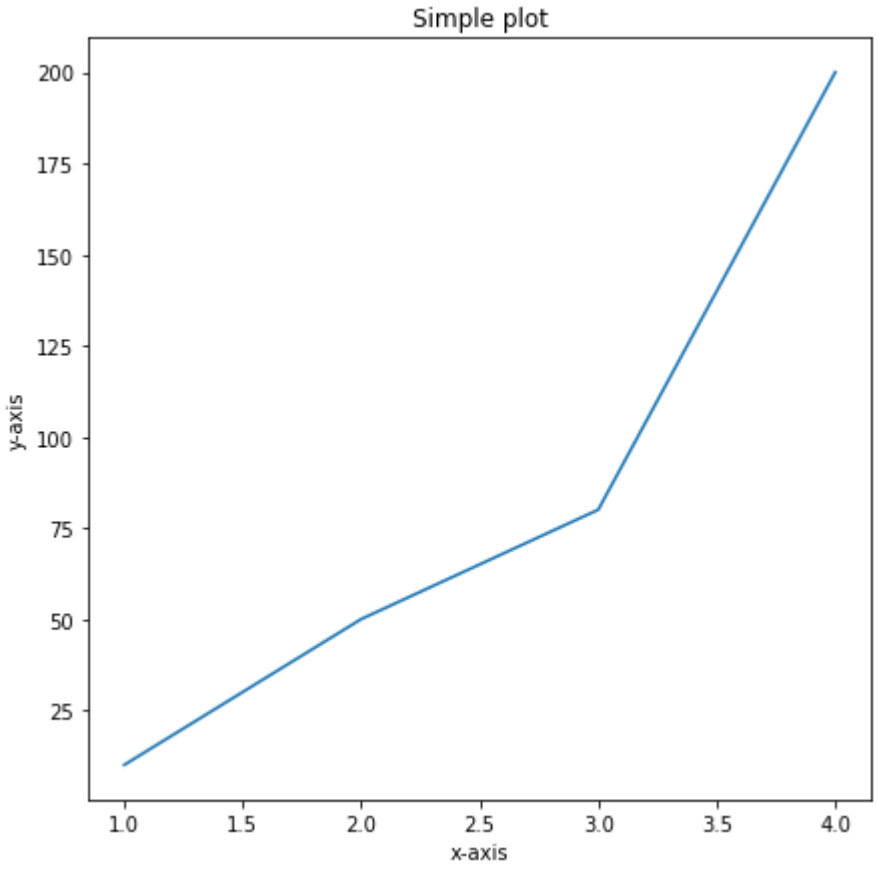
In [7]:
```
# 3rd method (Recommended)
fig, ax = plt.subplots(figsize = (7,7)) # creats a figure and a set of subplots
ax.plot(x, [50, 100, 200, 250]);
type(fig), type(ax)
```

Out[7]: (matplotlib.figure.Figure, matplotlib.axes._subplots.AxesSubplot)



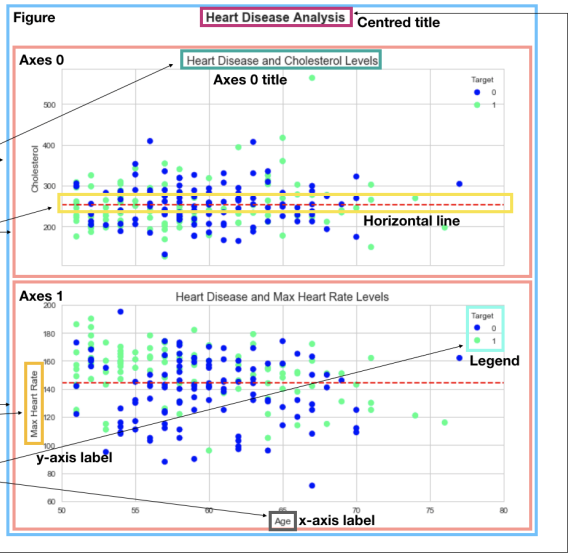
4. Customize plot, Save & Show (You save the whole figure)

In [8]:
```
fig, ax = plt.subplots(figsize = (7,7)) # (width, height)
ax.set(title="Simple plot", xlabel="x-axis", ylabel="y-axis")
ax.plot(x,y);
fig.savefig("sample-plot.png")
```



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');
```



Making figures with NumPy arrays

Baseline plots:

- Line plot
- Scatter plot
- Bar plot
- Histogram
- Subplot

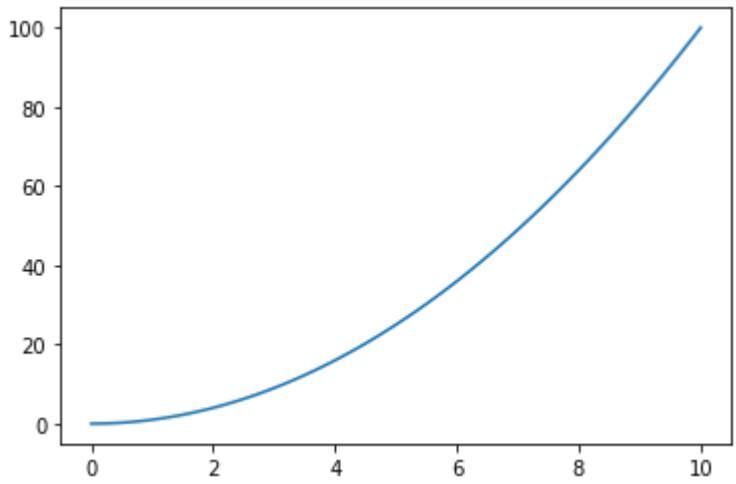
In [9]:
```
import numpy as np
```

In [10]:
```
# Create some data
x = np.linspace(0, 10, 100) # Start, Stop, Num ; Returns evenly spaced numbers ov
x
```

Out[10]:
```
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.        ])
```

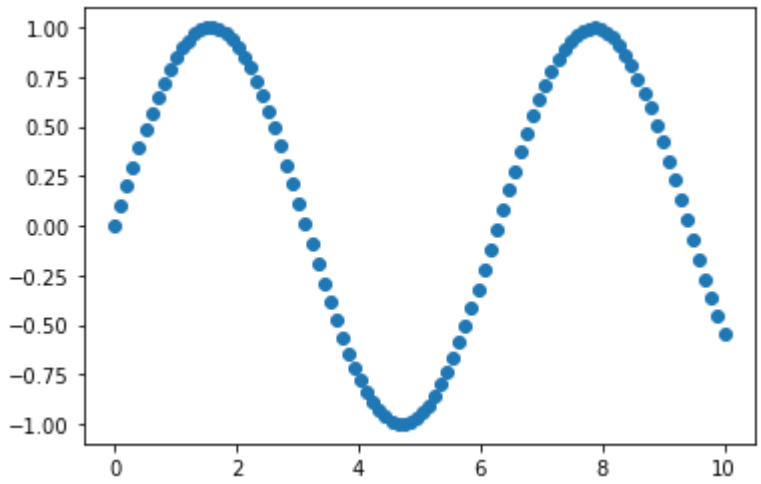
Line plot

In [11]:
```
# Plot the data and create a line plot (default plot)
fig, ax = plt.subplots()
ax.plot(x, x**2);
```



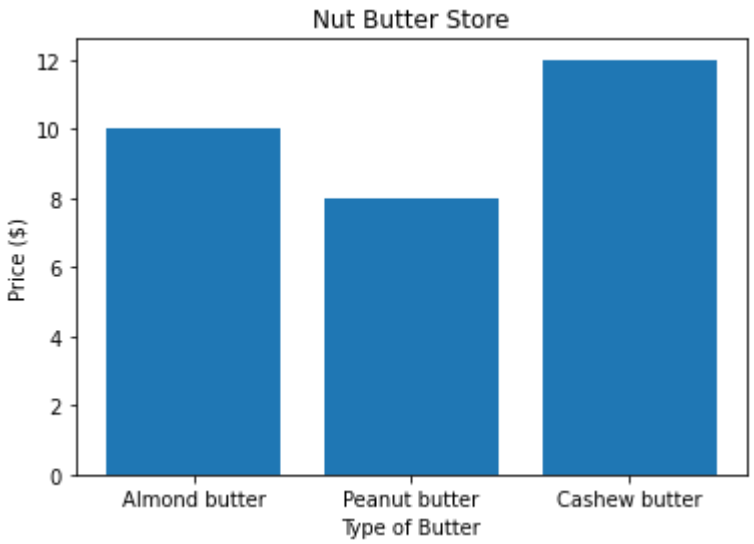
Scatter plot

In [12]:
```
# Use same data to make a scatter plot
fig, ax = plt.subplots()
ax.scatter(x, np.sin(x));
```

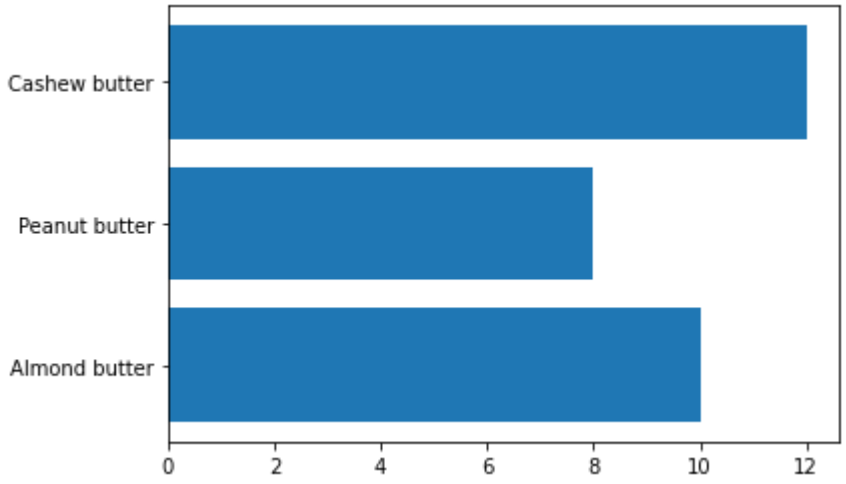


Bar plot

In [13]:
```
# Make a bar 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()); # x, height
ax.set(title="Nut Butter Store", xlabel="Type of Butter", ylabel="Price ($)");
```

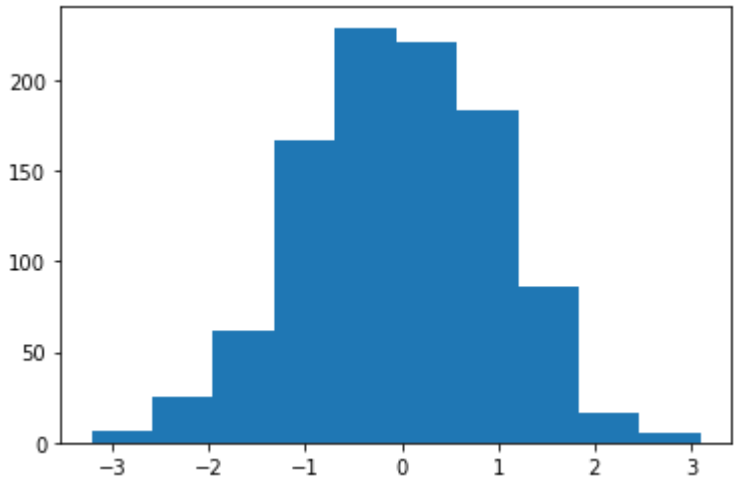


In [14]:
```
# Horizontal
fig, ax = plt.subplots()
ax.barh(list(nut_butter_prices.keys()), list(nut_butter_prices.values()));
        # Values needs to be converted to list for barh
```



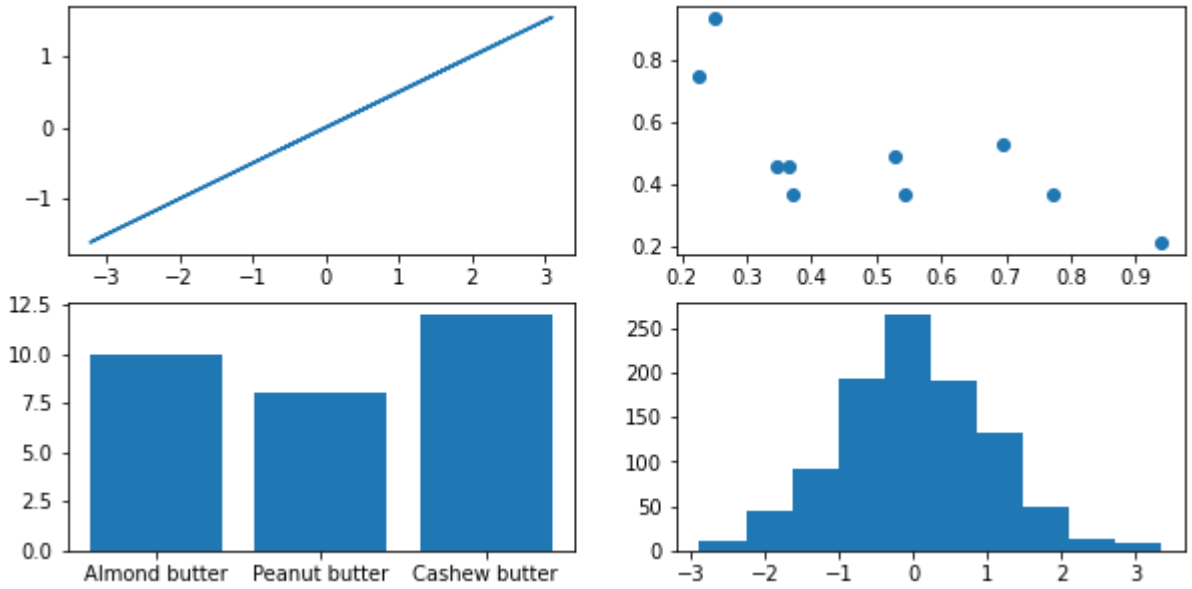
Histogram

In [15]:
```
# Make some data for histogram and plot it
x = np.random.randn(1000)
fig, ax = plt.subplots()
ax.hist(x);
```

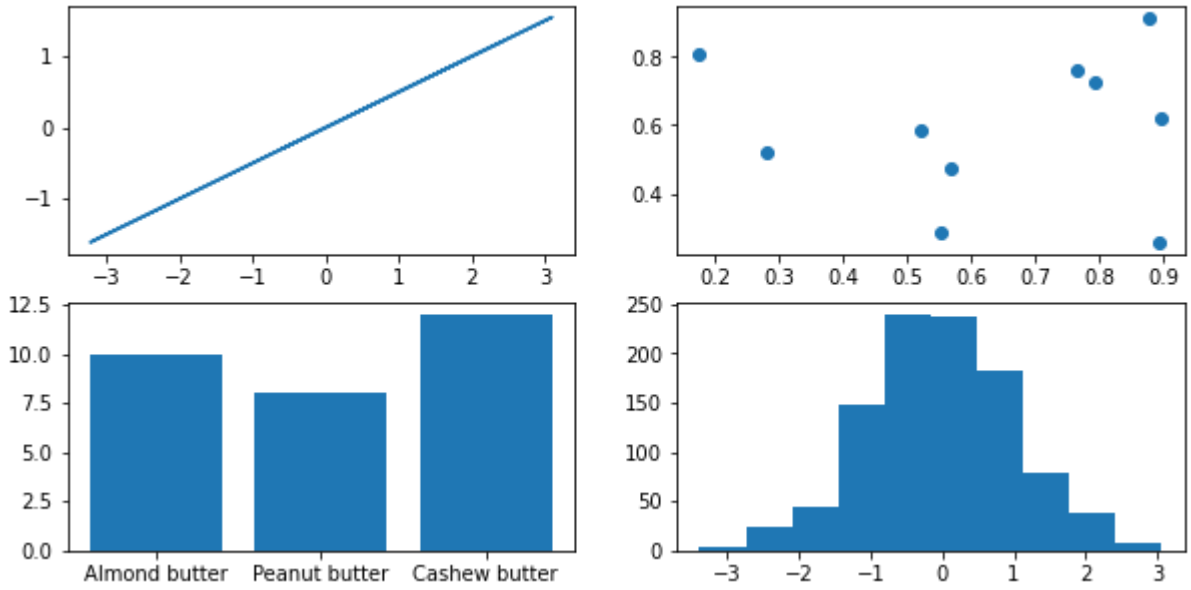


Subplot

In [16]:
```
# First option
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(1000));
```



In [17]:
```
# Second option
fix, 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));
```



Plotting from Pandas DataFrames

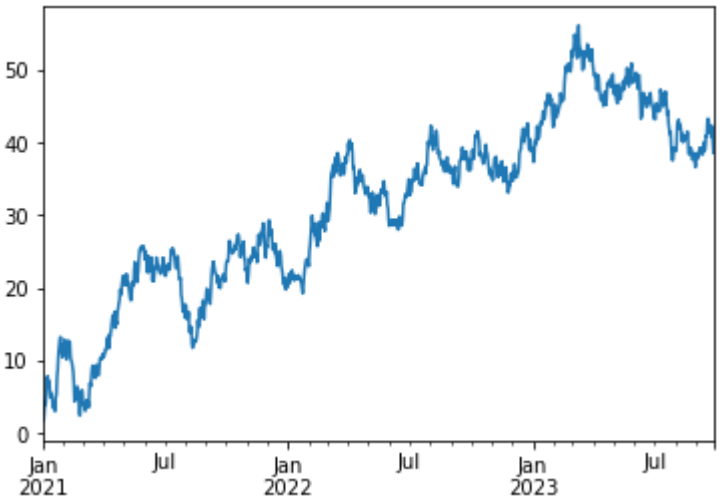
In [18]:
```
import pandas as pd
```

In [19]:
```
# Make a dataframe
car_sales = pd.read_csv("car-sales.csv")
car_sales
```

Out[19]:

	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 [20]:
```
ts = pd.Series(np.random.randn(1000),
               index=pd.date_range("1/1/2021", periods=1000))
ts = ts.cumsum() # Return cumulative sum over Dataframe or Series axis
ts.plot();
```



In [21]:
```
# Plotting from DataFrame
car_sales["Price"] = car_sales["Price"].str.replace('[\$\,\.]', '') # convert pri
car_sales
```

```
C:\Users\sonar\AppData\Local\Temp/ipykernel_8944/3396562050.py:2: FutureWarnin
g: The default value of regex will change from True to False in a future versio
n.
  car_sales["Price"] = car_sales["Price"].str.replace('[\$\,\.]', '') # convert
price into int
```

Out[21]:

	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 [22]:
```
# Remove last 2 zeros, currently price is a string
car_sales["Price"] = car_sales["Price"].str[:-1]
car_sales
```

Out[22]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	40000
1	Honda	Red	87899	4	50000
2	Toyota	Blue	32549	3	70000
3	BMW	Black	11179	5	220000
4	Nissan	White	213095	4	35000
5	Toyota	Green	99213	4	45000
6	Honda	Blue	45698	4	75000
7	Honda	Blue	54738	4	70000
8	Toyota	White	60000	4	62500
9	Nissan	White	31600	4	97000

In [23]:
```
car_sales["Sale Date"] = pd.date_range("12/12/2021", periods=len(car_sales))
car_sales
```

Out[23]:

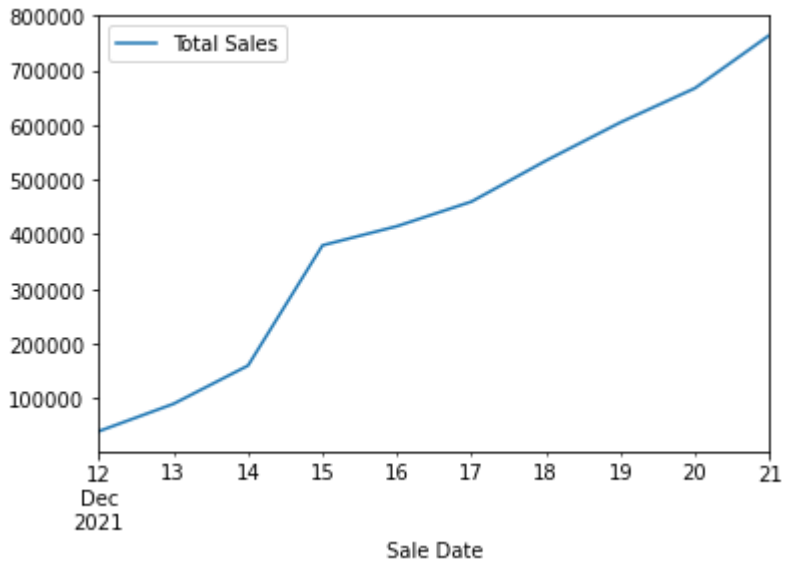
	Make	Colour	Odometer (KM)	Doors	Price	Sale Date
0	Toyota	White	150043	4	40000	2021-12-12
1	Honda	Red	87899	4	50000	2021-12-13
2	Toyota	Blue	32549	3	70000	2021-12-14
3	BMW	Black	11179	5	220000	2021-12-15
4	Nissan	White	213095	4	35000	2021-12-16
5	Toyota	Green	99213	4	45000	2021-12-17
6	Honda	Blue	45698	4	75000	2021-12-18
7	Honda	Blue	54738	4	70000	2021-12-19
8	Toyota	White	60000	4	62500	2021-12-20
9	Nissan	White	31600	4	97000	2021-12-21

In [24]:
```
car_sales["Total Sales"] = car_sales["Price"].astype(int).cumsum()
car_sales
```

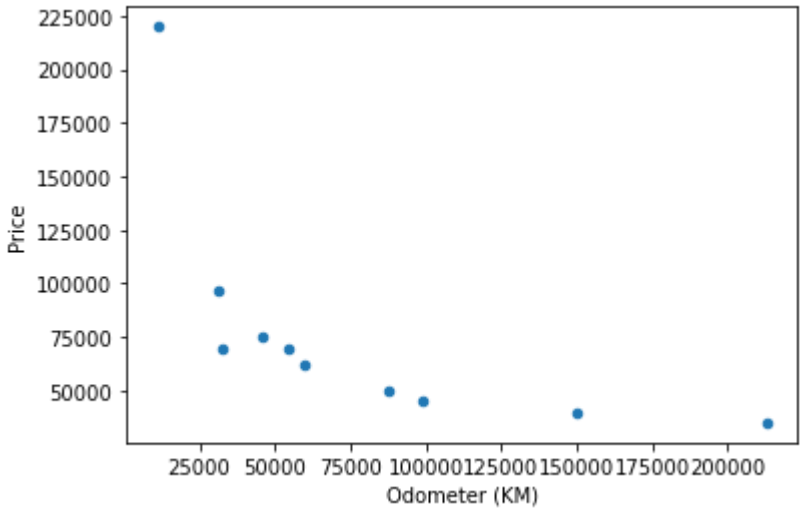
Out[24]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	40000	2021-12-12	40000
1	Honda	Red	87899	4	50000	2021-12-13	90000
2	Toyota	Blue	32549	3	70000	2021-12-14	160000
3	BMW	Black	11179	5	220000	2021-12-15	380000
4	Nissan	White	213095	4	35000	2021-12-16	415000
5	Toyota	Green	99213	4	45000	2021-12-17	460000
6	Honda	Blue	45698	4	75000	2021-12-18	535000
7	Honda	Blue	54738	4	70000	2021-12-19	605000
8	Toyota	White	60000	4	62500	2021-12-20	667500
9	Nissan	White	31600	4	97000	2021-12-21	764500

In [25]:
```
# Plotting total sales
car_sales.plot(x="Sale Date", y="Total Sales");
```



In [26]:
```
# Reassign price column to int
car_sales["Price"] = car_sales["Price"].astype(int)
# Plot scatter plot with price column as numeric
car_sales.plot(x="Odometer (KM)", y="Price", kind="scatter");
```

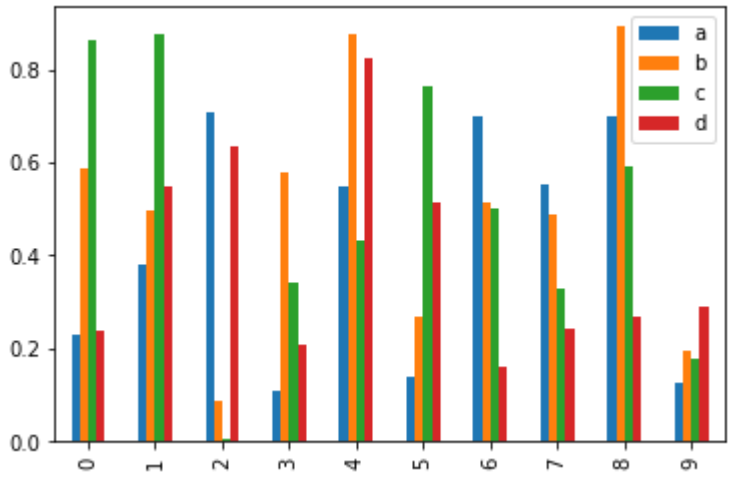


In [27]:
```
# Bar graph
x = np.random.rand(10, 4)
# Turn x into dataframe
df = pd.DataFrame(x, columns=['a', 'b', 'c', 'd'])
df
```

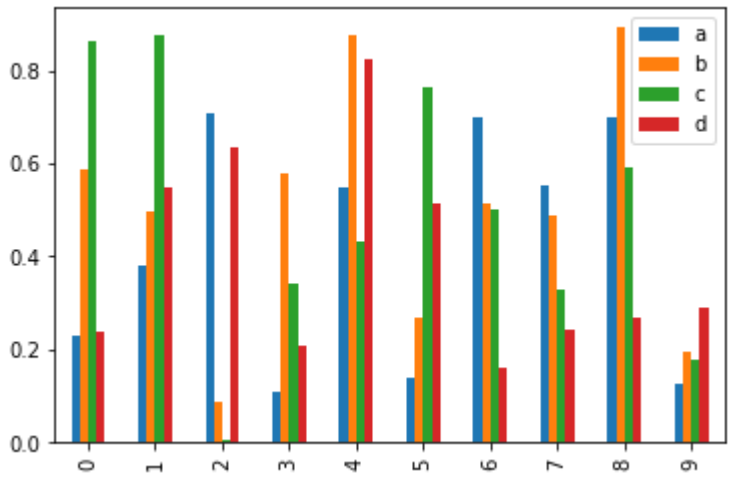
Out[27]:

	a	b	c	d
0	0.227940	0.586324	0.863546	0.239234
1	0.378600	0.495611	0.877597	0.546857
2	0.709690	0.089831	0.005476	0.634092
3	0.109496	0.577726	0.341140	0.208169
4	0.548322	0.873561	0.434316	0.823410
5	0.138942	0.267483	0.764431	0.514773
6	0.700645	0.514342	0.499251	0.160485
7	0.553449	0.489807	0.327930	0.242700
8	0.699437	0.891238	0.591480	0.268902
9	0.125929	0.195237	0.178912	0.288415

In [28]:
```
df.plot.bar();
```



In [29]:
```
# OR
df.plot(kind="bar");
```

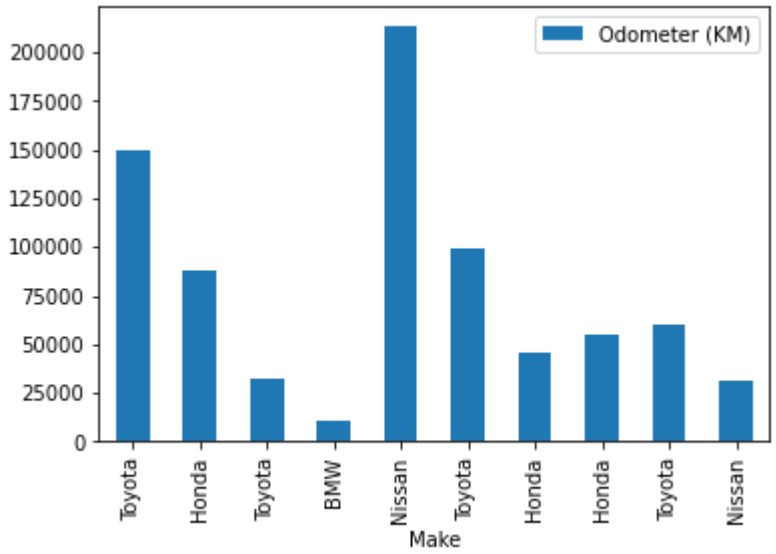


In [30]:
```
car_sales
```

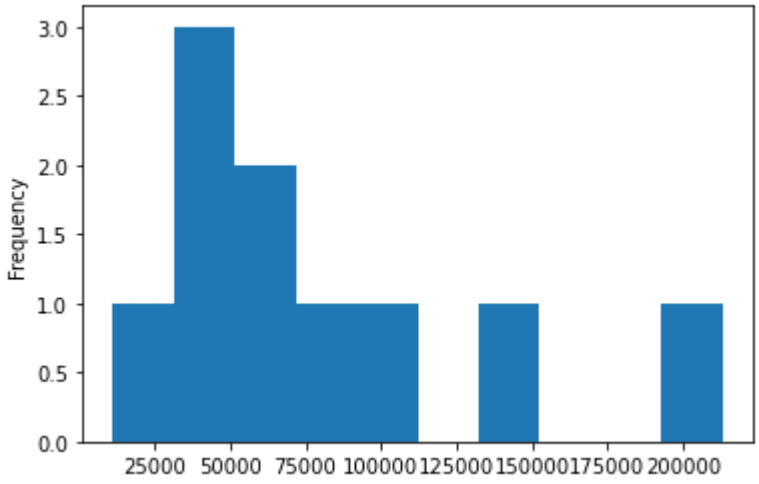
Out[30]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	40000	2021-12-12	40000
1	Honda	Red	87899	4	50000	2021-12-13	90000
2	Toyota	Blue	32549	3	70000	2021-12-14	160000
3	BMW	Black	11179	5	220000	2021-12-15	380000
4	Nissan	White	213095	4	35000	2021-12-16	415000
5	Toyota	Green	99213	4	45000	2021-12-17	460000
6	Honda	Blue	45698	4	75000	2021-12-18	535000
7	Honda	Blue	54738	4	70000	2021-12-19	605000
8	Toyota	White	60000	4	62500	2021-12-20	667500
9	Nissan	White	31600	4	97000	2021-12-21	764500

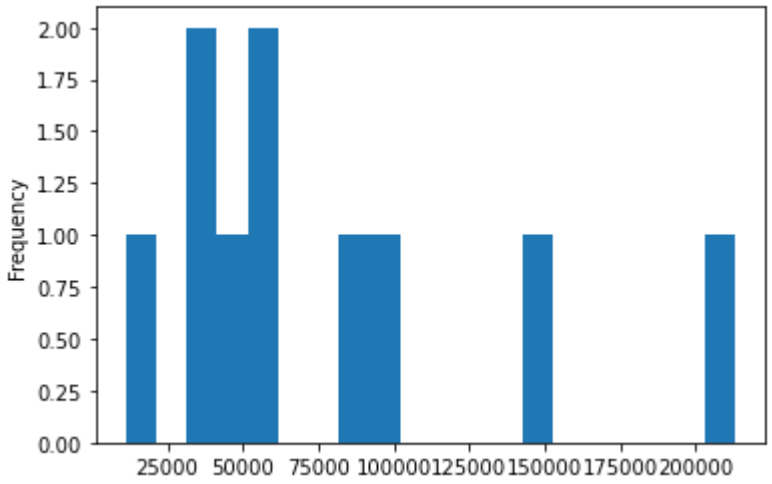
In [31]:
```
car_sales.plot(x="Make", y="Odometer (KM)", kind="bar");
```



In [32]:
```
# Histograms : Find distribution of odometer
car_sales["Odometer (KM)"].plot(kind="hist");
```



In [33]:
```
car_sales["Odometer (KM)"].plot.hist(bins=20); # Bins is 10 by default
```



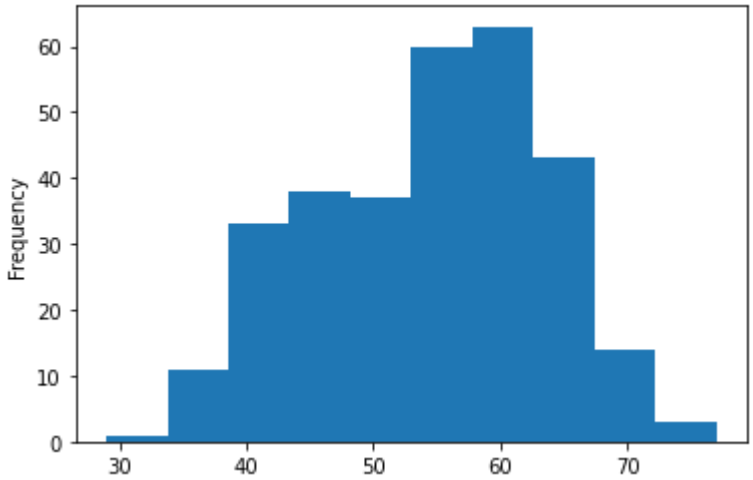
In [34]:
```
# Using another dataset
heart_disease = pd.read_csv("heart-disease.csv")
heart_disease
```

Out[34]:

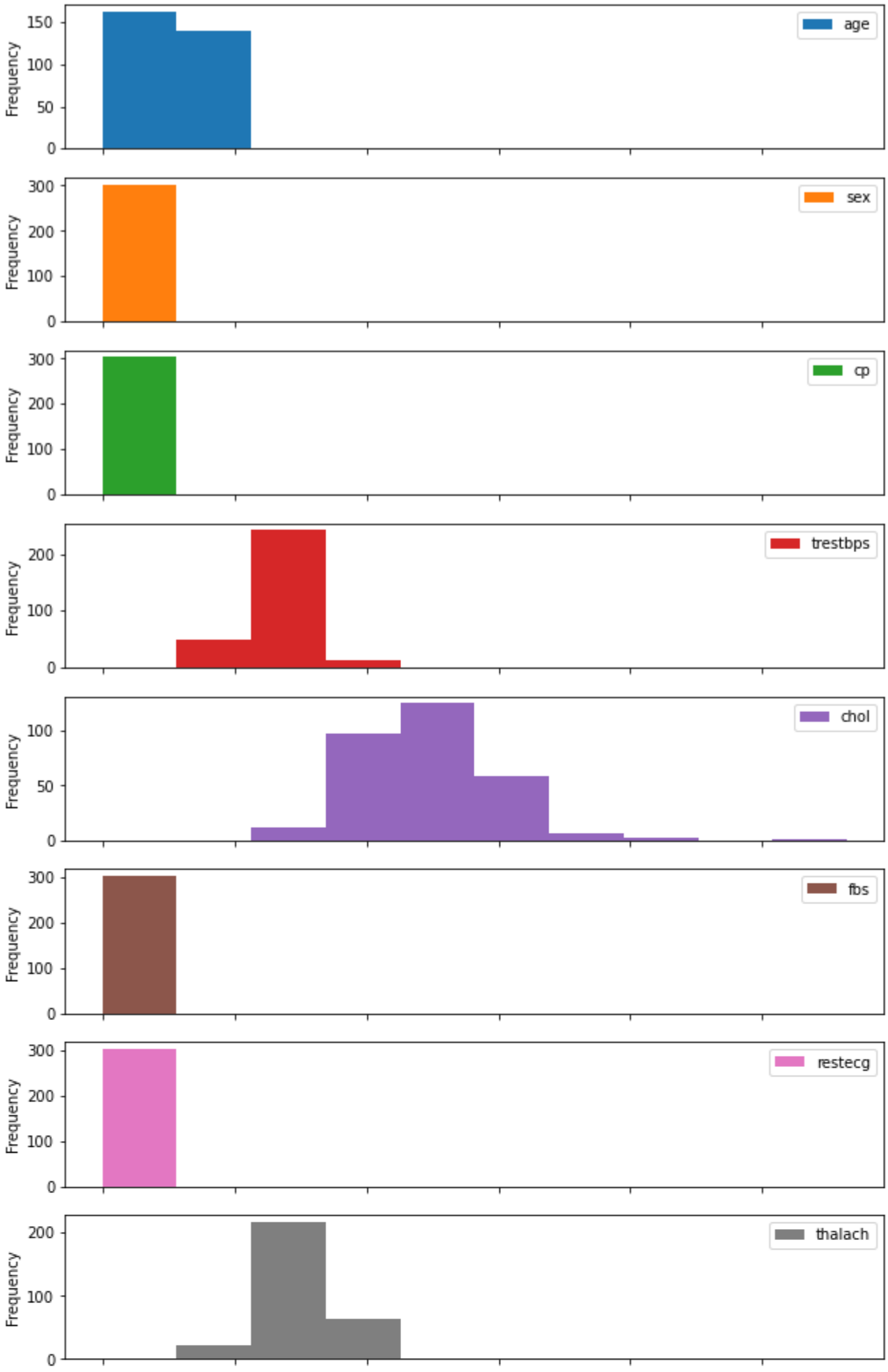
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

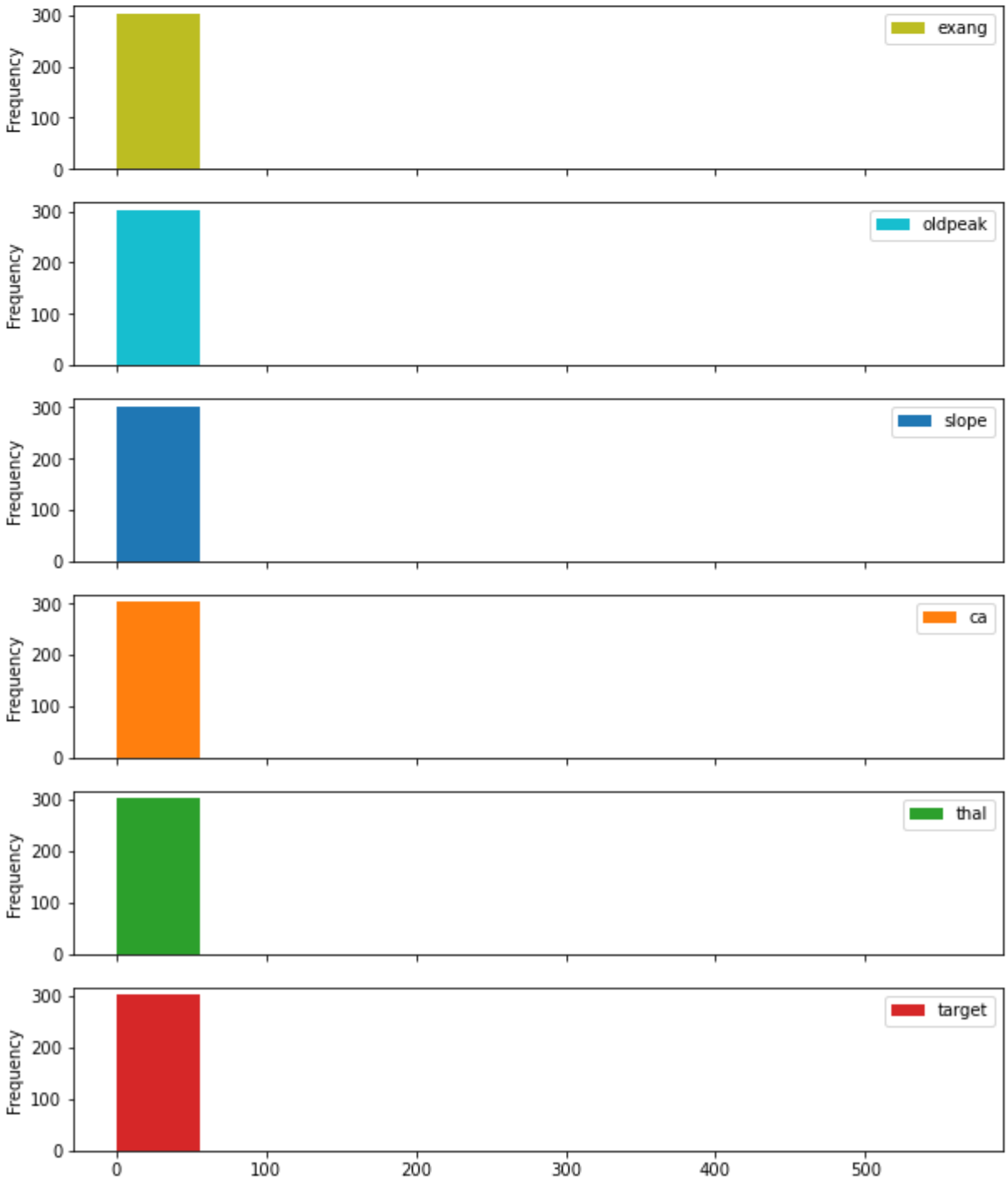
303 rows × 14 columns

In [35]:
```
# Create a histogram to see distribution of age
heart_disease["age"].plot.hist();
```



In [36]:
```
heart_disease.plot.hist(figsize=(10, 30), subplots=True);
```





Which one to use? (Pyplot Vs Matplotlib OO method)

- When potting something quickly its okay to use pyplot method
- When plotting something more advanced, use the OO method

Object Orienteted (OO) mehod

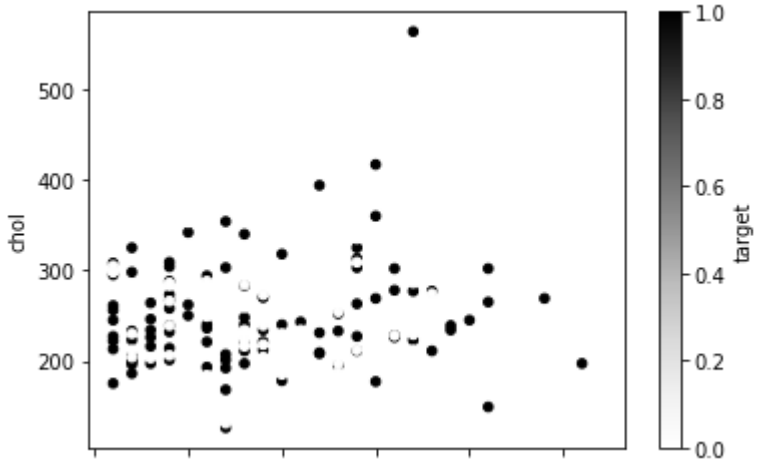
In [37]:
```
over_50 = heart_disease[heart_disease["age"] > 50]
over_50
```

Out[37]:

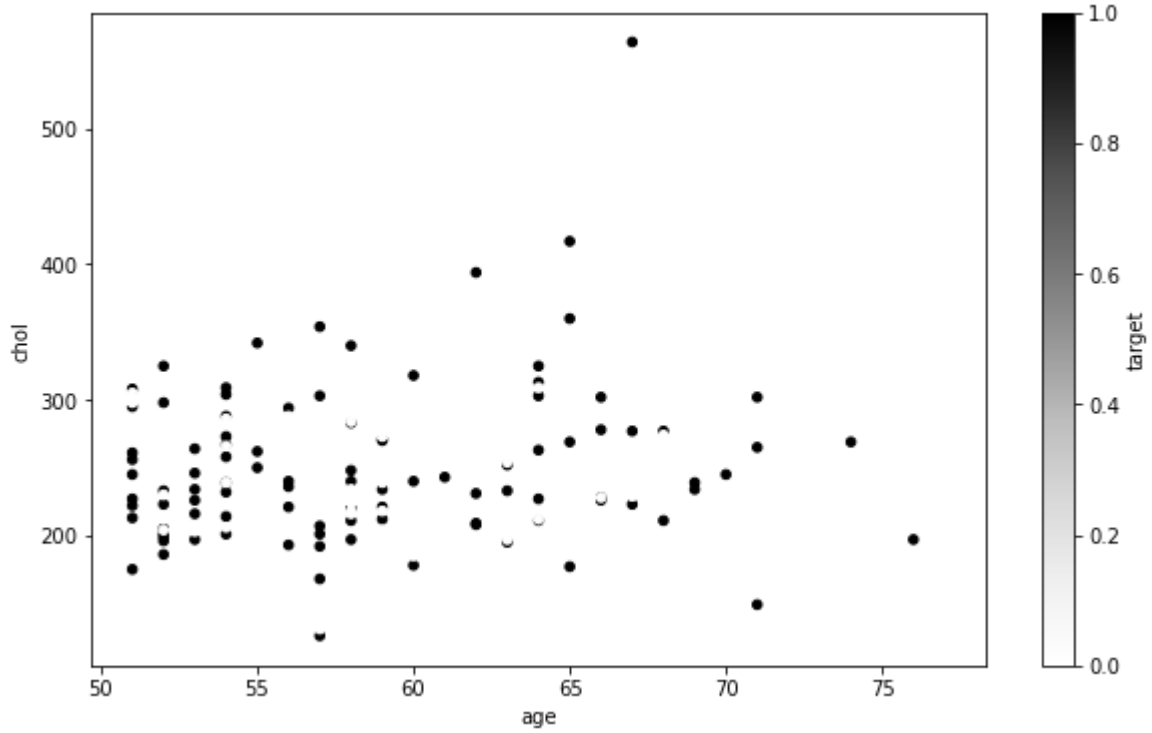
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
297	59	1	0	164	176	1	0	90	0	1.0	1	2	1	0
298	57	0	0	140	241	0	1	123	1	0.2	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

208 rows × 14 columns

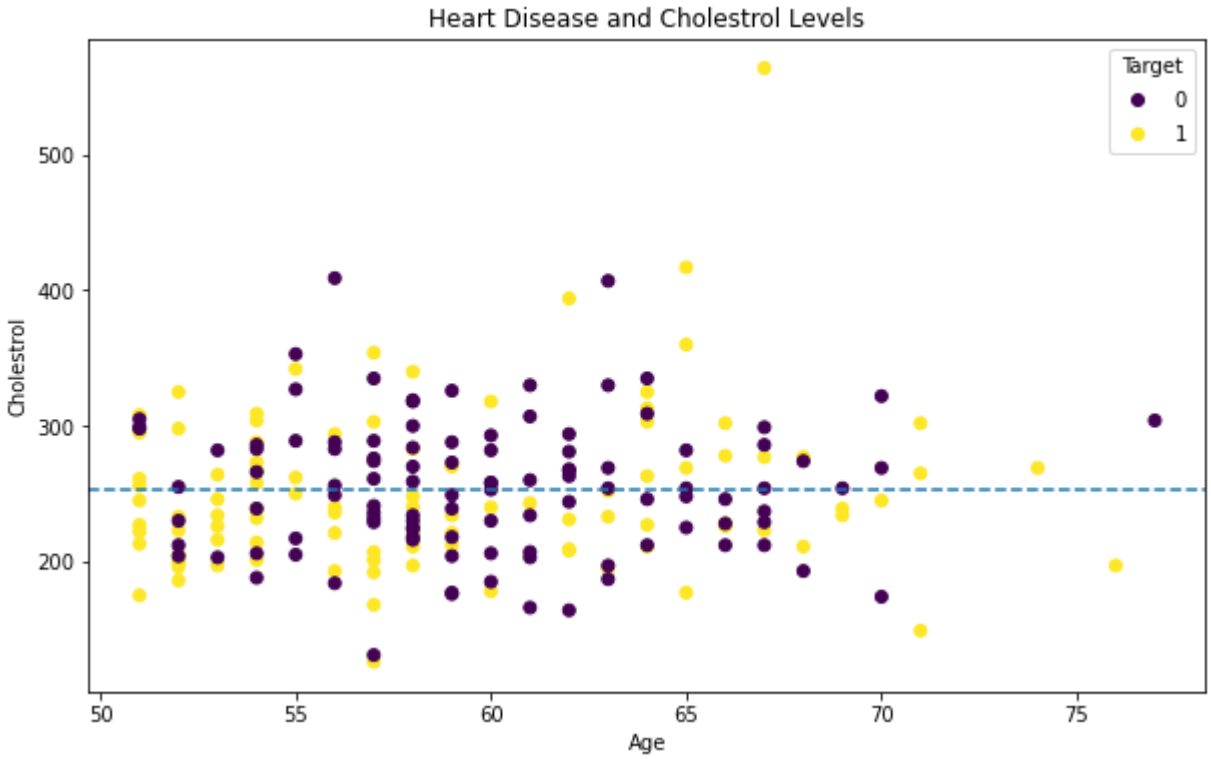
In [38]:
```
# Pyplot method
over_50.plot(kind="scatter",
             x="age",
             y="chol",
             c="target"); # c=color
```



In [39]:
```
## OO method mixed with pyplot
fig, ax = plt.subplots(figsize=(10, 6))
over_50.plot(kind="scatter",
             x="age",
             y="chol",
             c="target",
             ax=ax); # Plot func gets 'ax' parameter in case of OO method
```



In [40]:
```
## 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 Cholestrol Levels",
       xlabel="Age",
       ylabel="Cholestrol");
# Add a legend
ax.legend(*scatter.legend_elements(), title="Target");
# Add a horizontal line showing mean cholestrol
ax.axhline(over_50["chol"].mean(), linestyle="--");
```



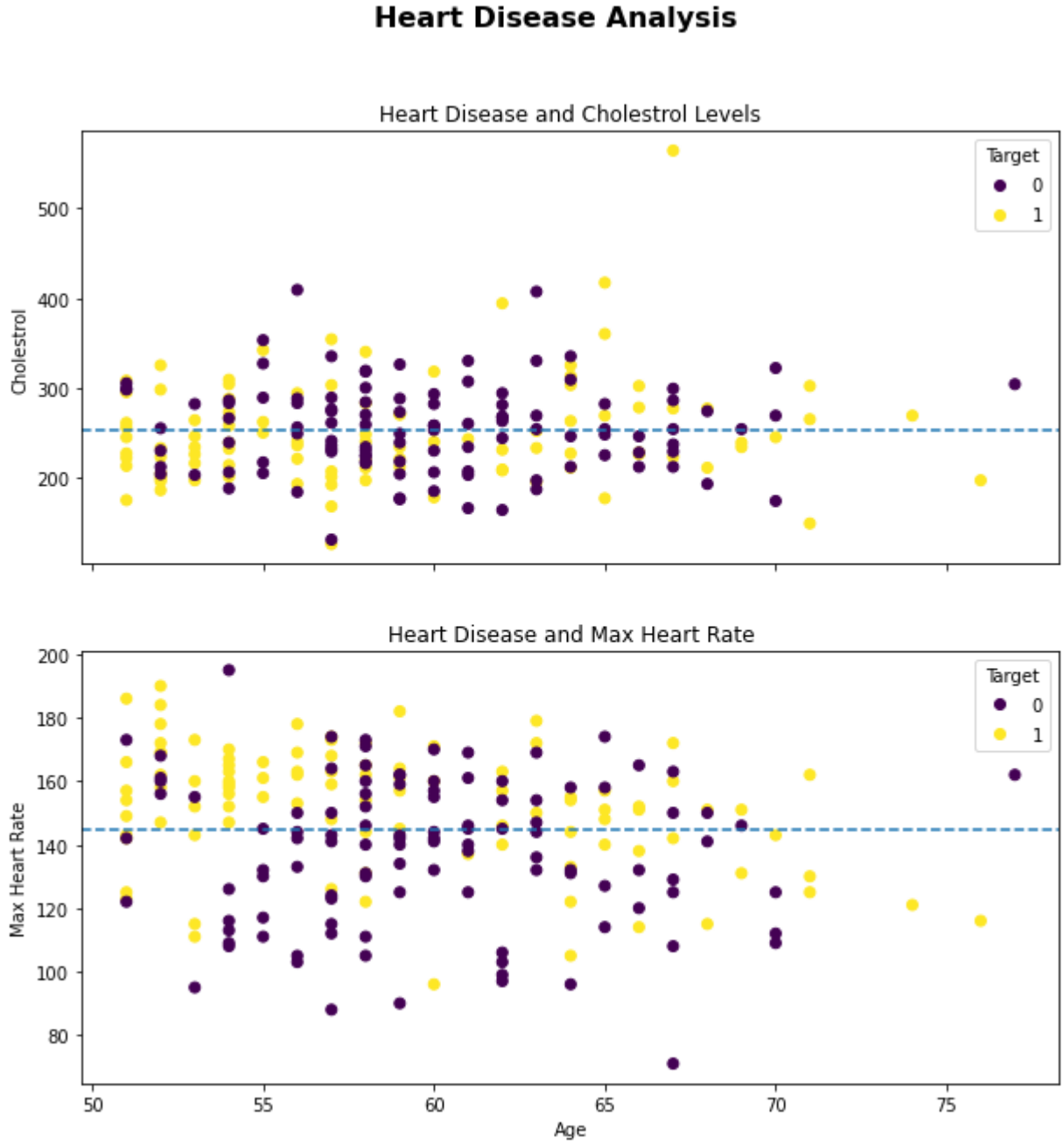
In [41]:
```
over_50
```

Out[41]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
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 [42]:
```
## Subplot of chol, age, thalach
fig, (ax0, ax1) = plt.subplots(nrows=2,
                               ncols=1,
                               figsize=(10,10),
                               sharex=True) # Share X-axis values

# 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 Cholestrol Levels",
      ylabel="Cholestrol");
# Add a legend to ax0
ax0.legend(*scatter.legend_elements(), title="Target");
# Add a horizontal line showing mean cholestrol to ax0
ax0.axhline(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 horizontal line showing mean cholestrol to ax0
ax1.axhline(over_50["thalach"].mean(), linestyle="--");

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



Customizing Matplotlib plots

In [43]:
```
# See the different styles available
plt.style.available
```

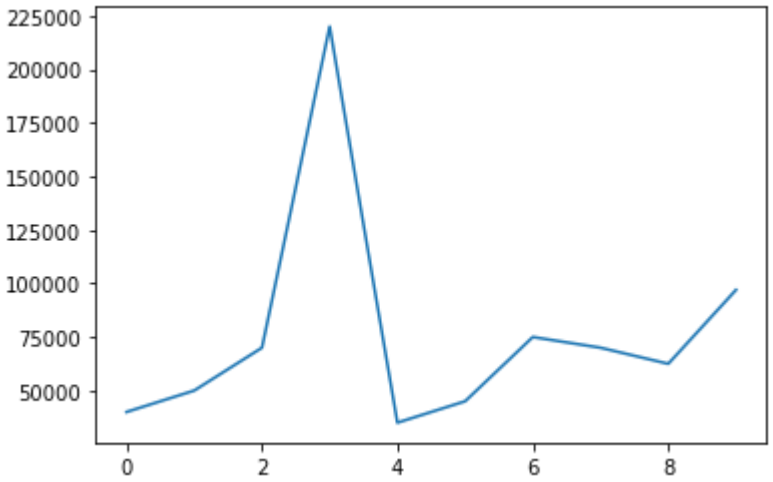
Out[43]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

In [44]:
```
car_sales.head()
```

Out[44]:

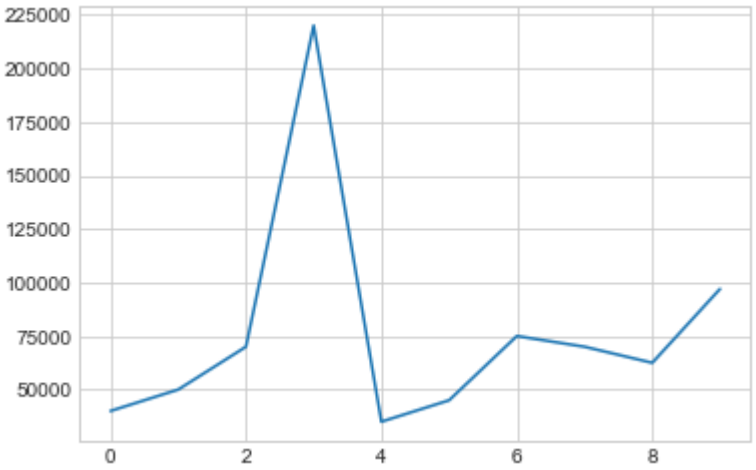
	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	40000	2021-12-12	40000
1	Honda	Red	87899	4	50000	2021-12-13	90000
2	Toyota	Blue	32549	3	70000	2021-12-14	160000
3	BMW	Black	11179	5	220000	2021-12-15	380000
4	Nissan	White	213095	4	35000	2021-12-16	415000

In [45]:
```
car_sales["Price"].plot();
```

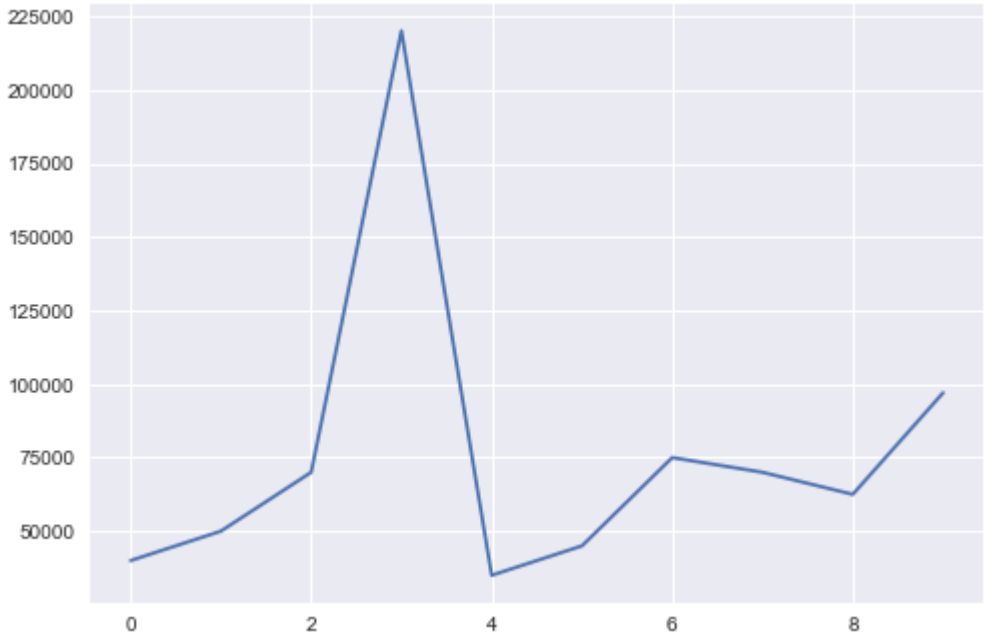


In [46]:
```
plt.style.use('seaborn-whitegrid') # Changes style internally
```

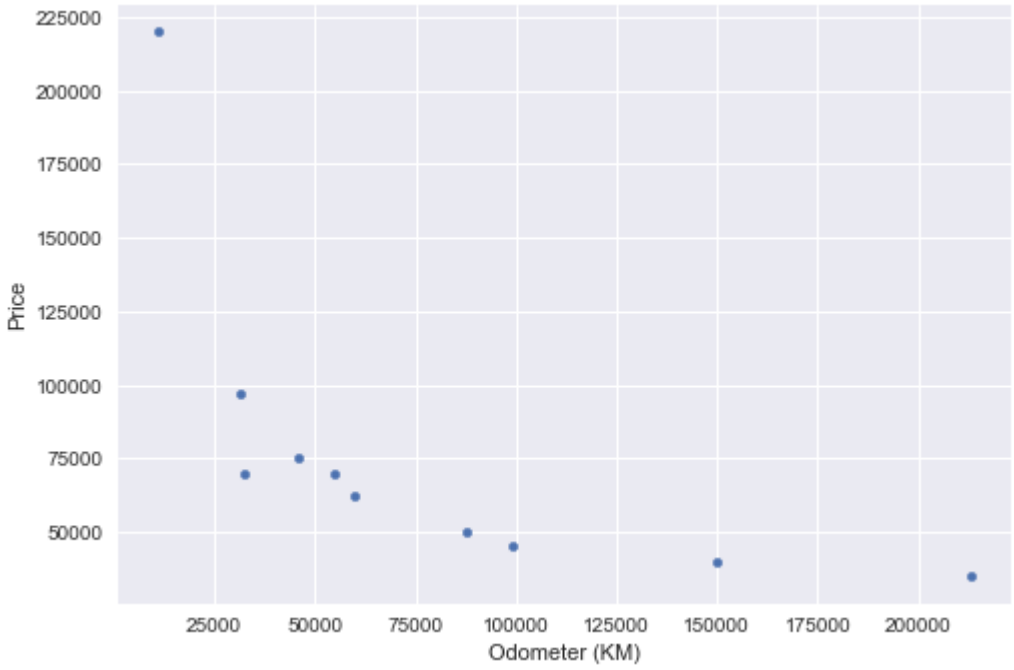
In [47]:
```
car_sales["Price"].plot();
```



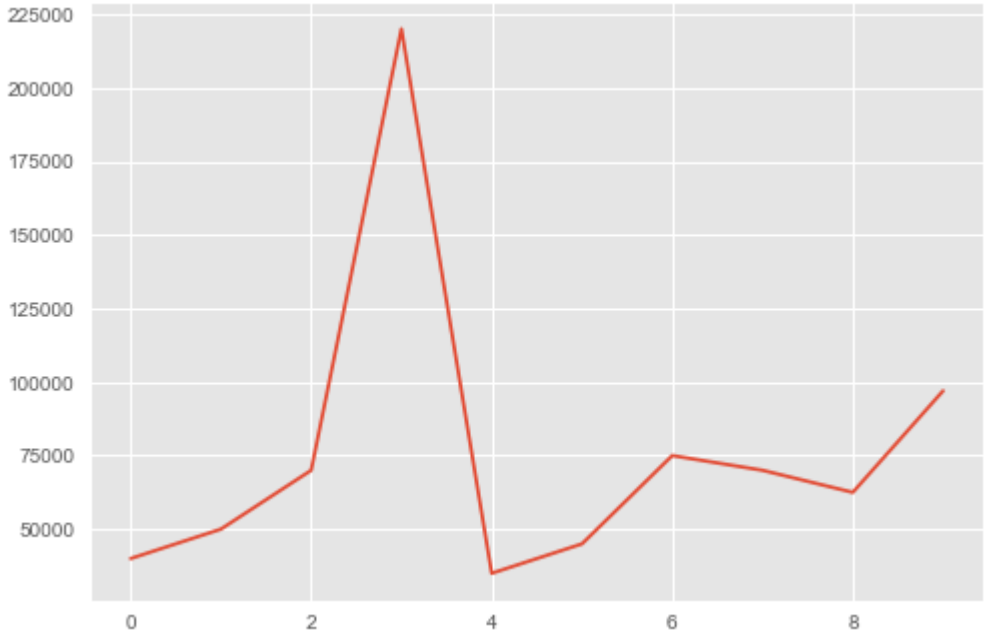
In [48]:
```
plt.style.use('seaborn')
car_sales["Price"].plot();
```



In [49]:
```
car_sales.plot(x="Odometer (KM)",
               y="Price",
               kind="scatter");
```



In [50]:
```
plt.style.use('ggplot')
car_sales["Price"].plot();
```



In [51]:
```
# Crate some data
x = np.random.randn(10, 4)
x
```

Out[51]:
```
array([[ 1.21468261, -0.9642426 , -1.45135276, -1.00459787],
       [-1.56953451, -0.37543385, -0.7370683 ,  1.40109098],
       [-0.36740529,  1.19766837, -0.91377723, -1.84964072],
       [ 0.80636313,  0.4140809 , -0.18659467, -1.2389172 ],
       [ 0.3949744 ,  1.51253709,  0.40103522,  0.63032606],
       [-0.28342546, -0.33793629,  1.04154529, -0.11023133],
       [ 1.437136  , -1.41151375,  0.41495131,  2.00376679],
       [ 0.81300292,  0.07346029,  0.90593225,  0.58666385],
       [ 0.74793954,  0.82096028,  0.72125269,  2.72245624],
       [-0.32304236, -0.79385376, -0.58447852, -1.11778578]])
```

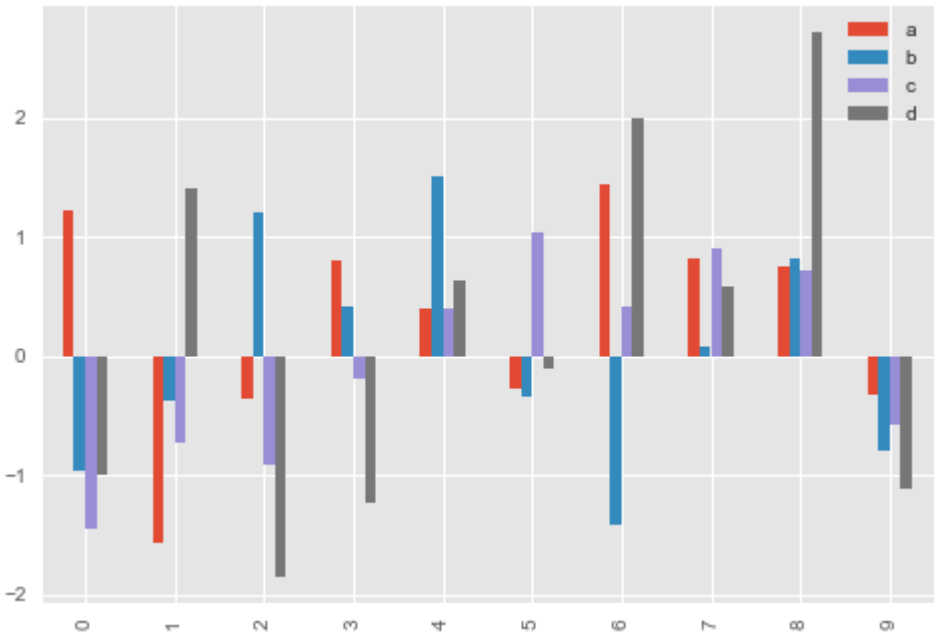
In [52]:
```
df = pd.DataFrame(x, columns=['a', 'b', 'c', 'd'])
df
```

Out[52]:

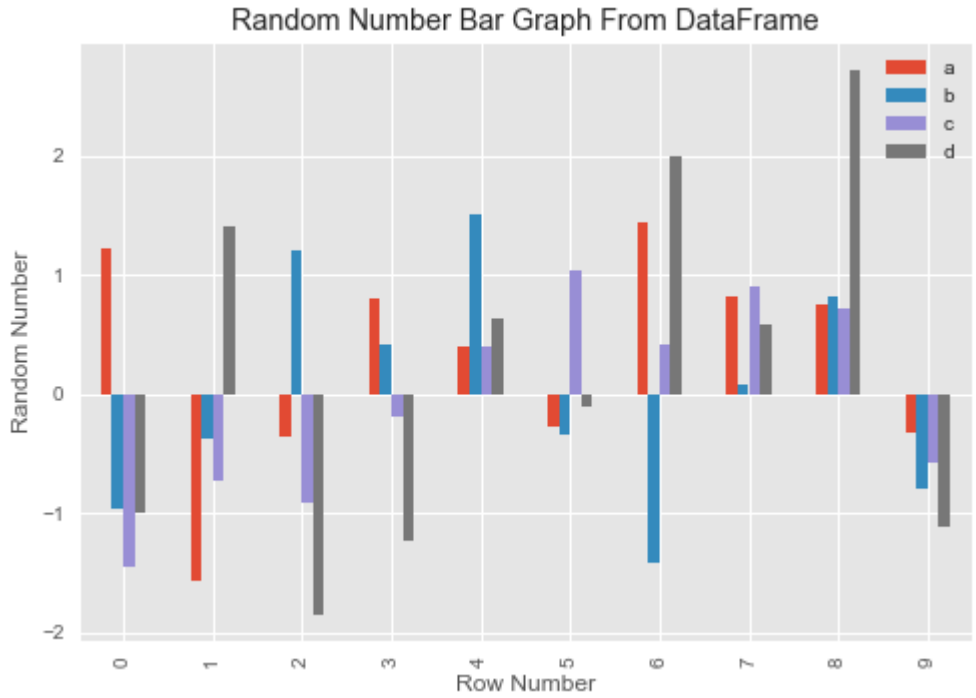
	a	b	c	d
0	1.214683	-0.964243	-1.451353	-1.004598
1	-1.569535	-0.375434	-0.737068	1.401091
2	-0.367405	1.197668	-0.913777	-1.849641
3	0.806363	0.414081	-0.186595	-1.238917
4	0.394974	1.512537	0.401035	0.630326
5	-0.283425	-0.337936	1.041545	-0.110231
6	1.437136	-1.411514	0.414951	2.003767
7	0.813003	0.073460	0.905932	0.586664
8	0.747940	0.820960	0.721253	2.722456
9	-0.323042	-0.793854	-0.584479	-1.117786

In [53]:
```
ax = df.plot(kind="bar")
type(ax)
```

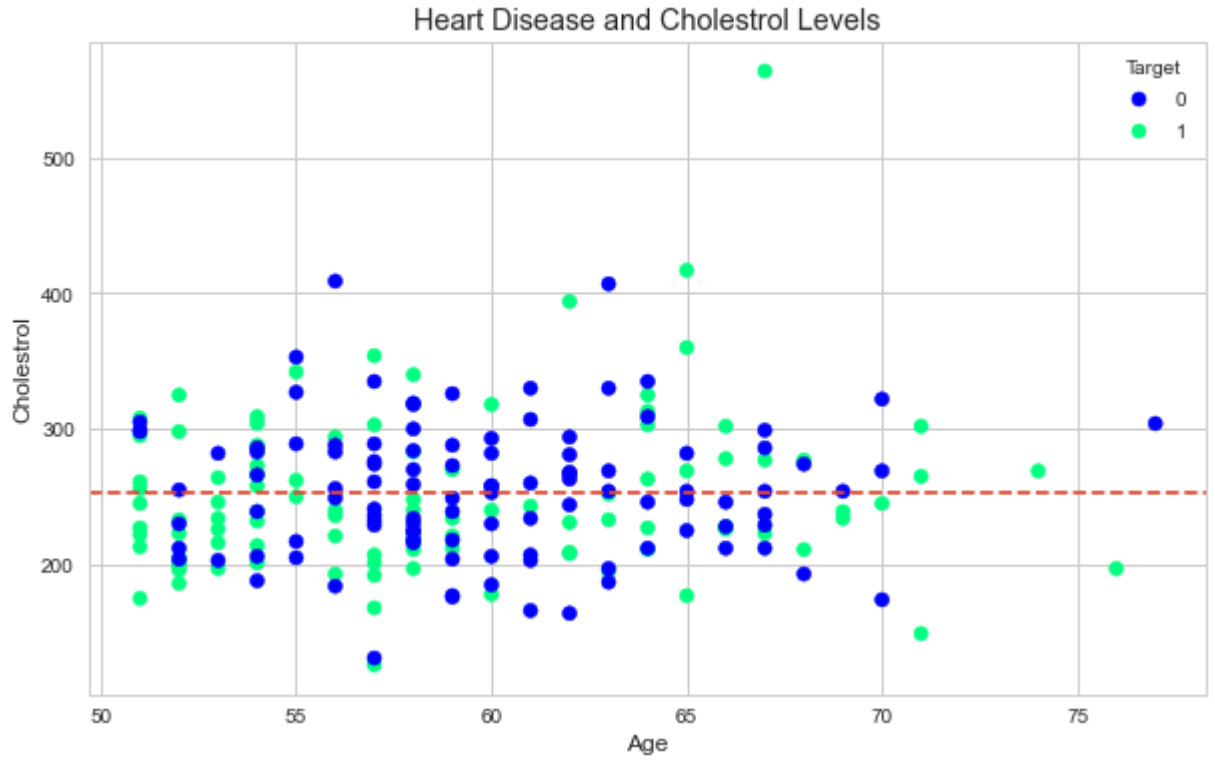
Out[53]: matplotlib.axes._subplots.AxesSubplot



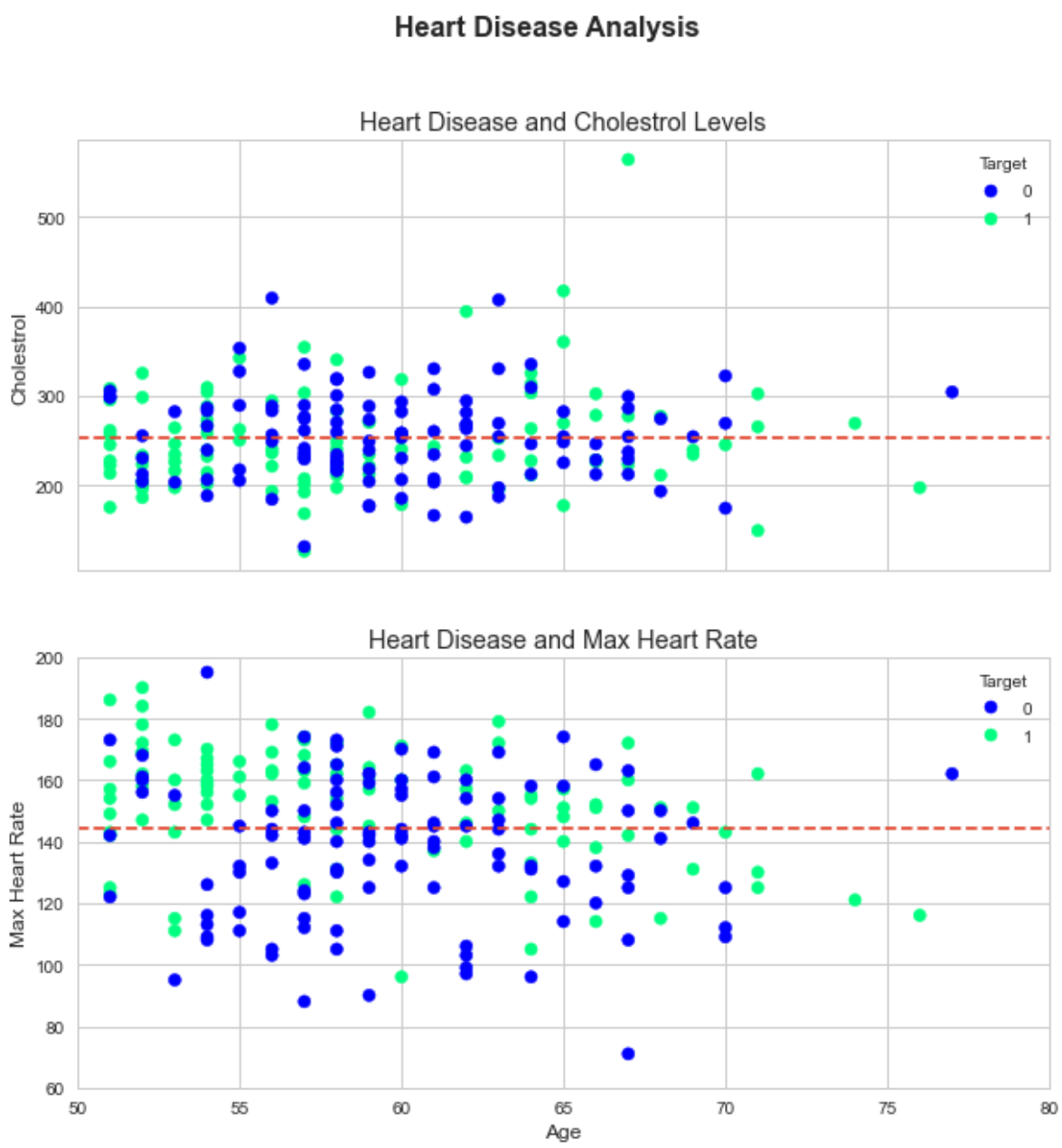
In [54]:
```
# Customize plot with set() method
ax = df.plot(kind="bar")
# Add labels and title
ax.set(title="Random Number Bar Graph From DataFrame",
       xlabel="Row Number",
       ylabel="Random Number");
# Make legend visible
ax.legend().set_visible(True) # In case its not visible
```



In [55]:
```
# Set style
plt.style.use('seaborn-whitegrid')
## 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"],
                     cmap="winter"); # Changes color scheme
# Customize the plot
ax.set(title="Heart Disease and Cholestrol Levels",
      xlabel="Age",
      ylabel="Cholestrol");
# Add a legend
ax.legend(*scatter.legend_elements(), title="Target");
# Add a horizontal line showing mean cholestrol
ax.axhline(over_50["chol"].mean(), linestyle="--");
```

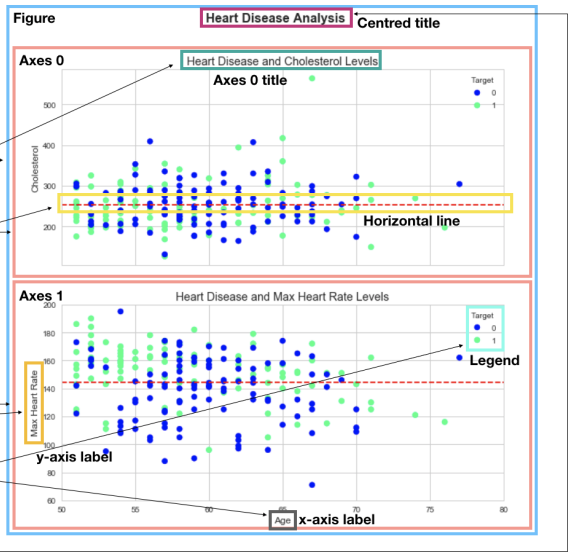


In [56]:
```
## Customizing the y and x axis limitations

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

# 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 Cholestrol Levels",
      ylabel="Cholestrol");
## Change the X axis limits
ax0.set_xlim([50, 80])
# Add a legend to ax0
ax0.legend(*scatter.legend_elements(), title="Target");
# Add a horizontal line showing mean cholestrol to ax0
ax0.axhline(over_50["chol"].mean(), linestyle="--");

# 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");
## Change the X & Y axis limits
ax1.set_xlim([50, 80])
ax1.set_ylim([60, 200])
# Add a legend to ax1
ax1.legend(*scatter.legend_elements(), title="Target");
# Add a horizontal line showing mean cholestrol to ax0
ax1.axhline(over_50["thalach"].mean(), linestyle="--");

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


Heart Disease Analysis
Heart Disease and Cholestrol Levels
Cholestrol
Target
0
1
500
400
300
200
Heart Disease and Max Heart Rate
Max Heart Rate
Target
0
1
200
180
160
140
120
100
80
60
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');
```

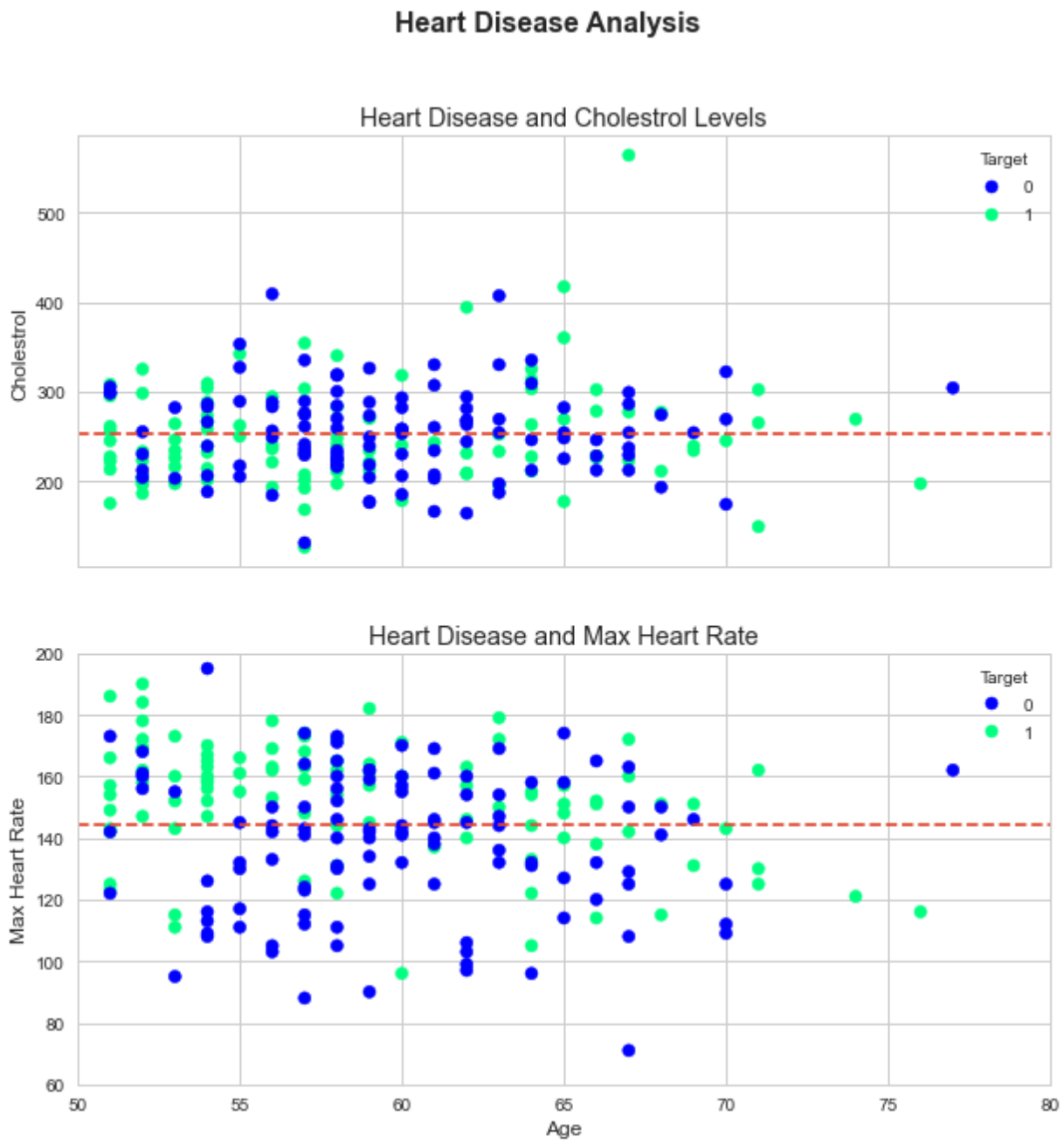


Saving and Sharing plots

- One way is to right click and select save image as
- Another way is using code

In [57]:
```
fig
```

Out[57]:



In [58]:
```
fig.savefig("heart-disease-analysis-plot-saved-with-code.png")
```