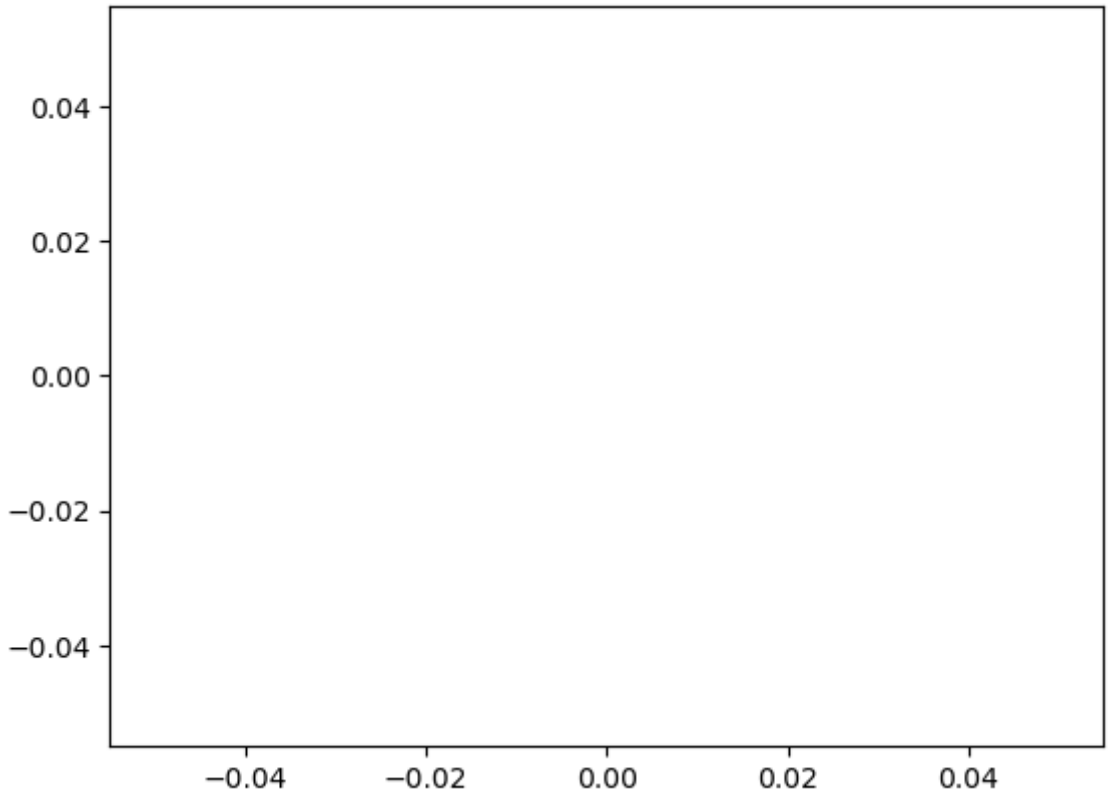
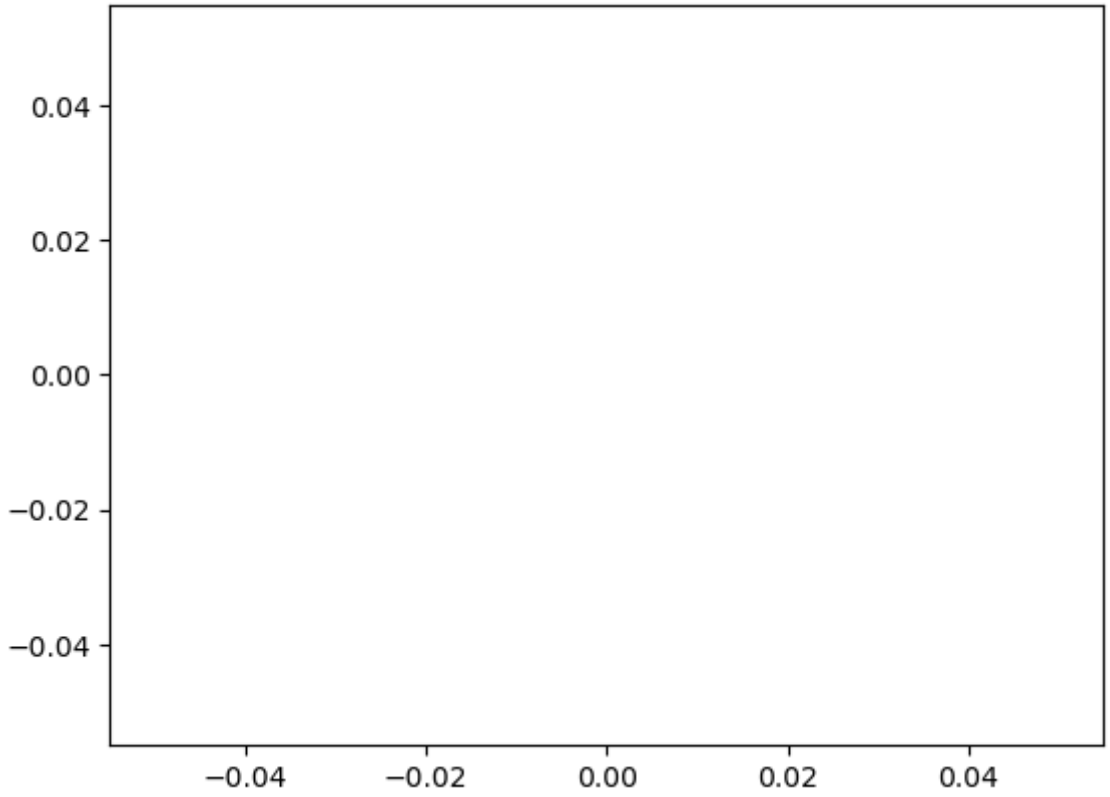
In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

In [2]:
```
plt.plot()
```

Out[2]:
```
[]
```

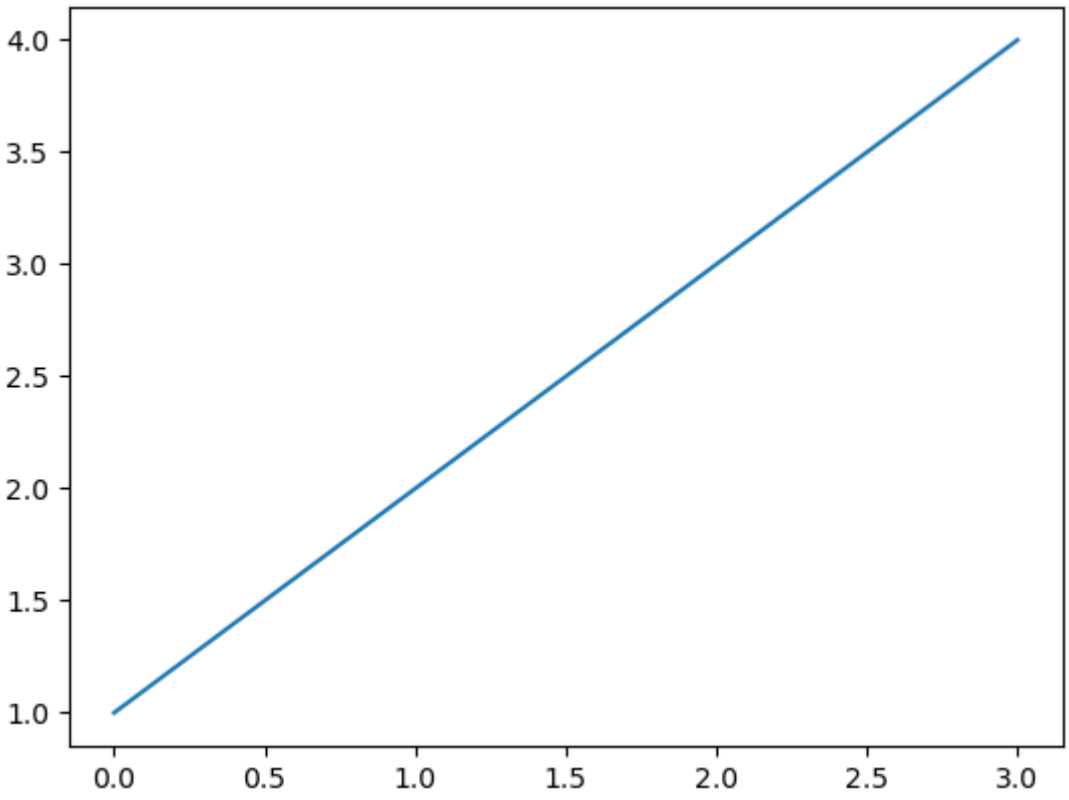



In [3]:
```
plt.plot(); # or plt.plot()
            # plt.show()
```

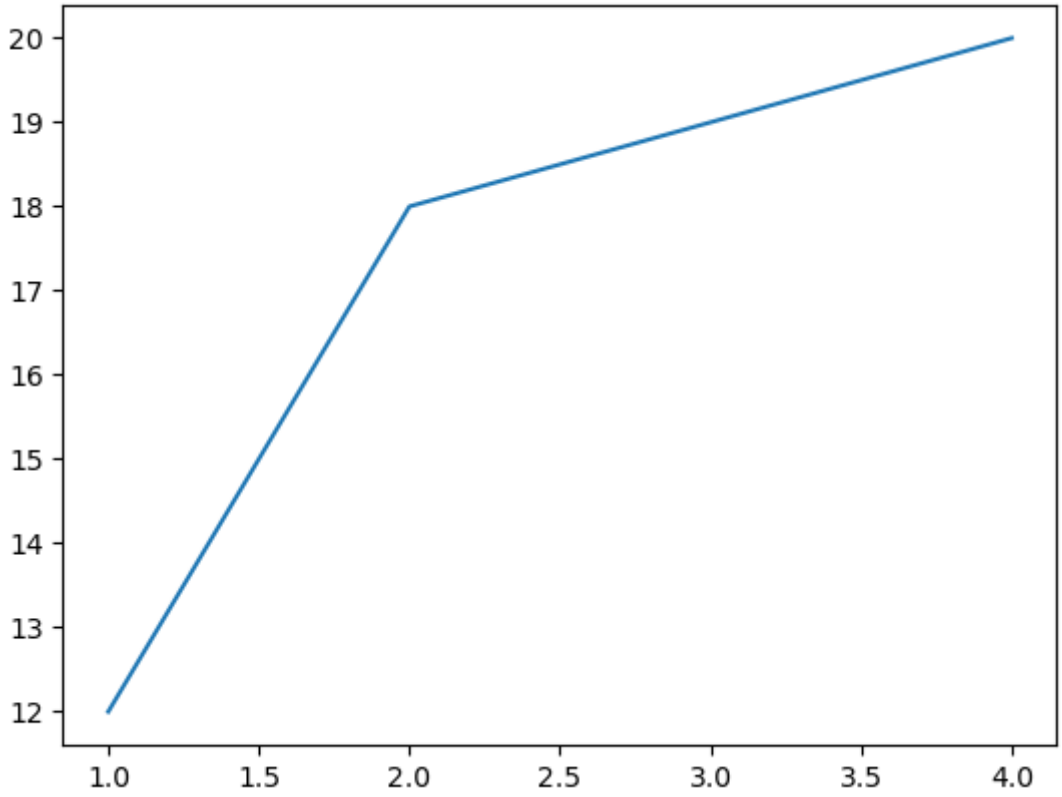



In [4]:
```
plt.plot([1,2,3,4])
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x2151a353b80>]
```



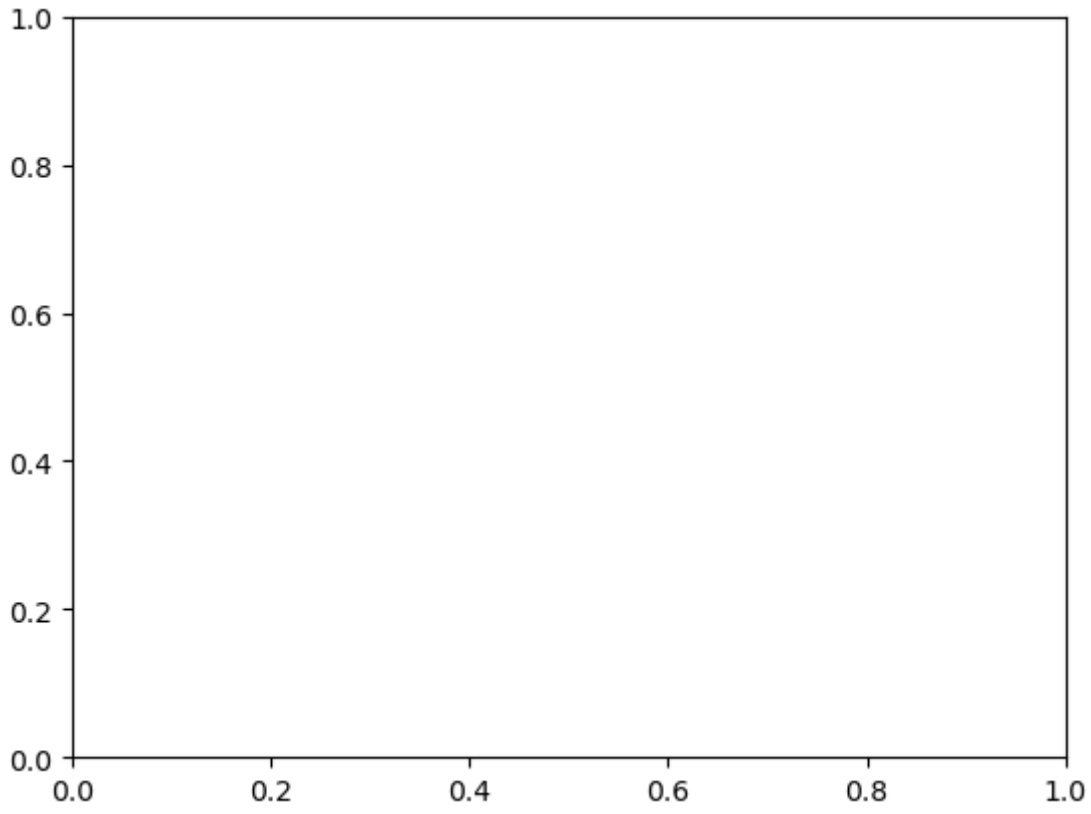

In [5]:
```
x = [1,2,3,4]
y = [12,18,19,20]
plt.plot(x,y);
```



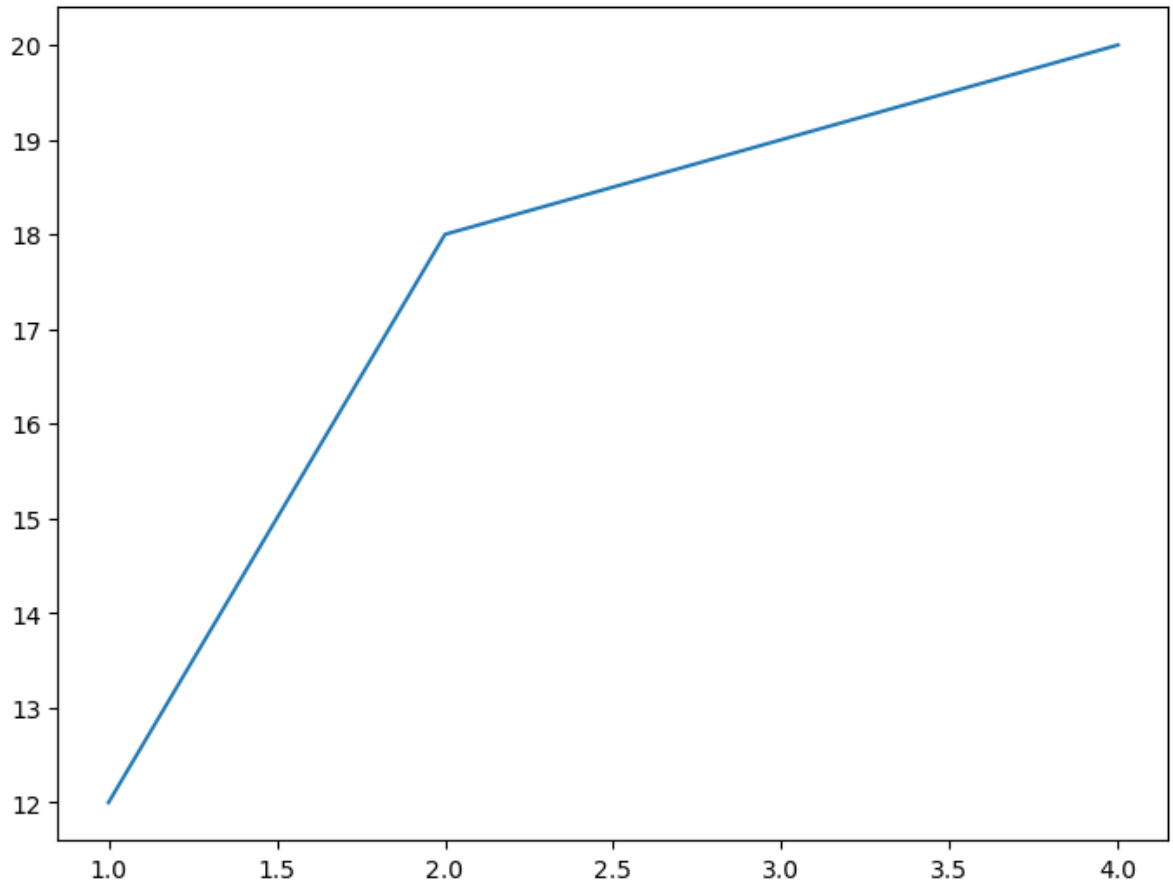

In [6]:
```
# Pyplot API is generally less flexible than object-oriented API
# we will focus on object oriented API
```

In [7]:
```
# 1st Method
fig = plt.figure() # it will create a figure with 0 axes
ax = fig.add_subplot() # to add axes in figure
plt.show
```

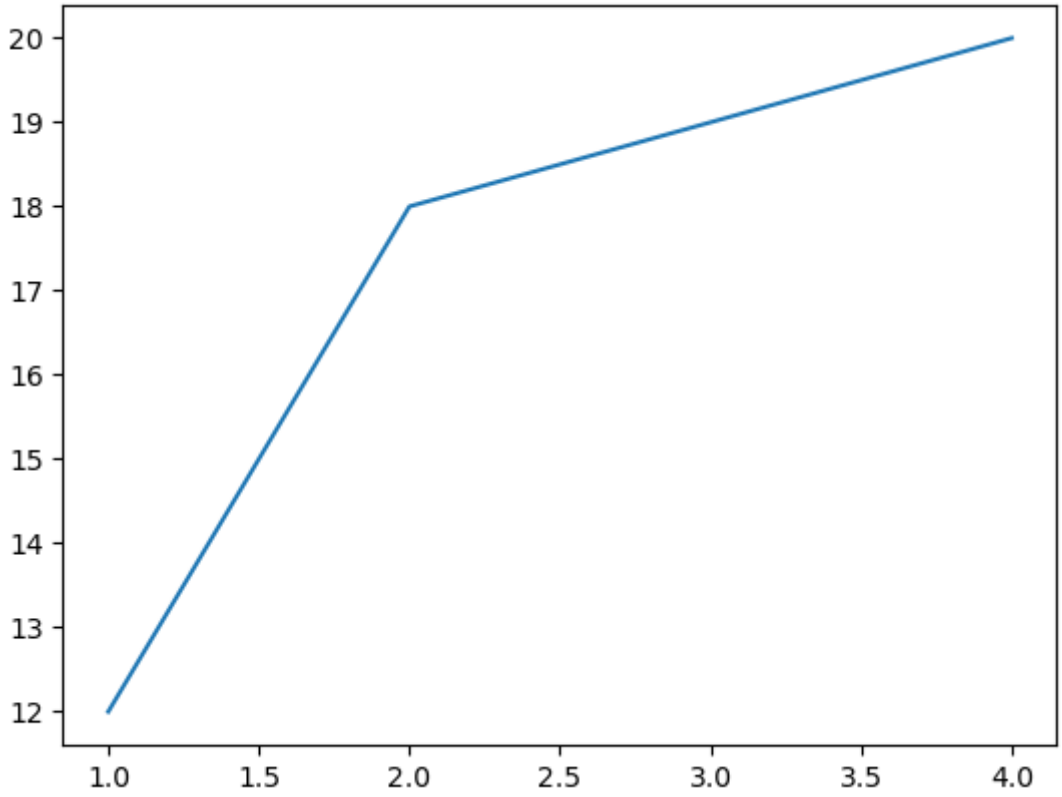
Out[7]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [8]:
```
# 2nd Method
fig = plt.figure() # create figure
ax = fig.add_axes([1,1,1,1]) # add axes
ax.plot(x,y)
plt.show()
```



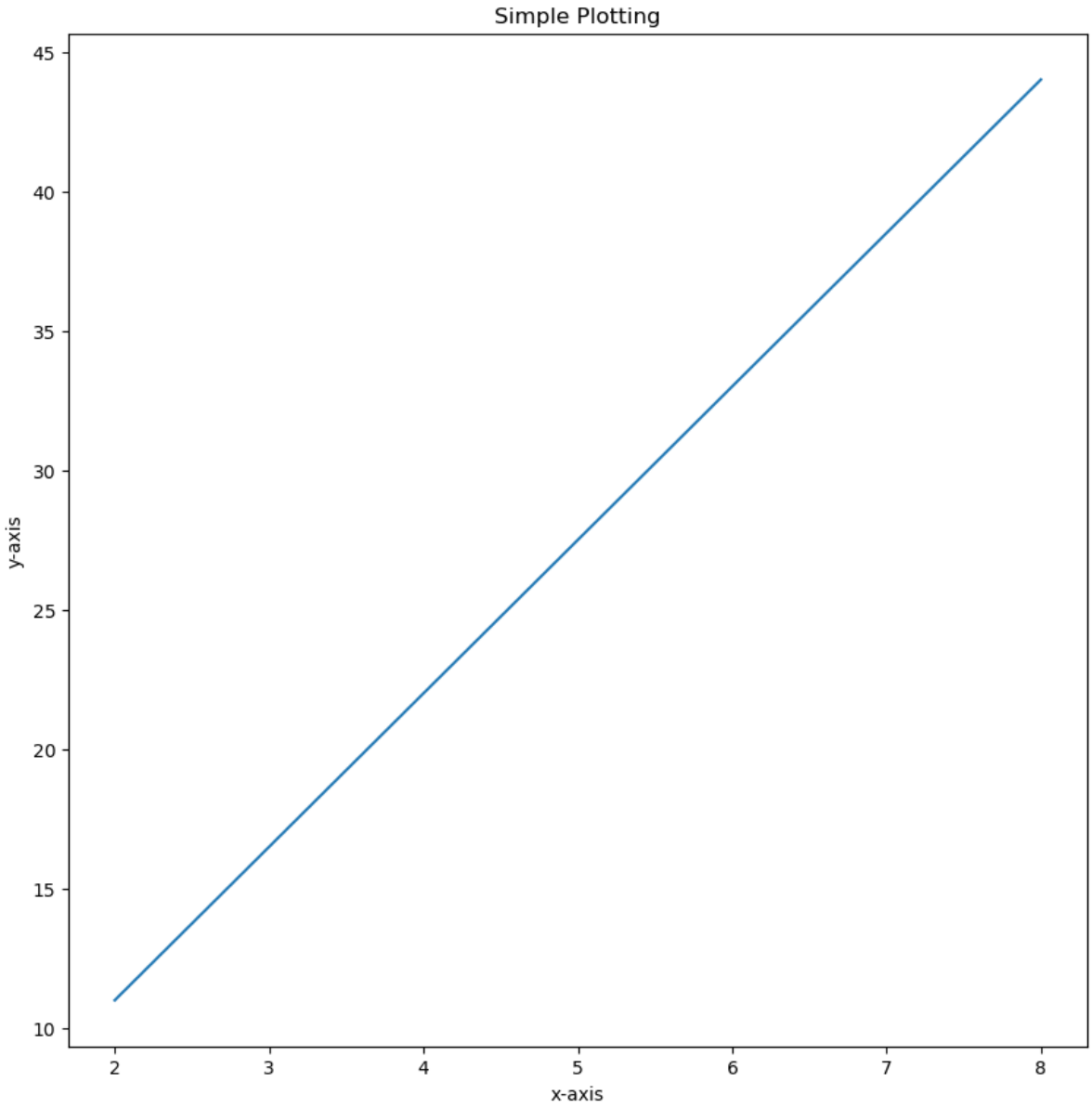

In [9]:
```
# 3rd Method(Recommended)
fig, ax = plt.subplots()
ax.plot(x,y);
```




In [ ]:

# Matplotlib workflow example

In [10]:
```
# start with importing matplotlib and get it ready for work in jupyter
%matplotlib inline
import matplotlib.pyplot as plt

# prepare date
x =[2,4,6,8]
y=[11,22,33,44]

# setup pot
fig, ax = plt.subplots(figsize=(10,10)) # width , height

#plot the data
ax.plot(x,y);

#cutomize  plot
ax.set(title="Simple Plotting", xlabel="x-axis", ylabel="y-axis")

#save & show
fig.savefig("matplotlib figures/simple_plotting.png")
```



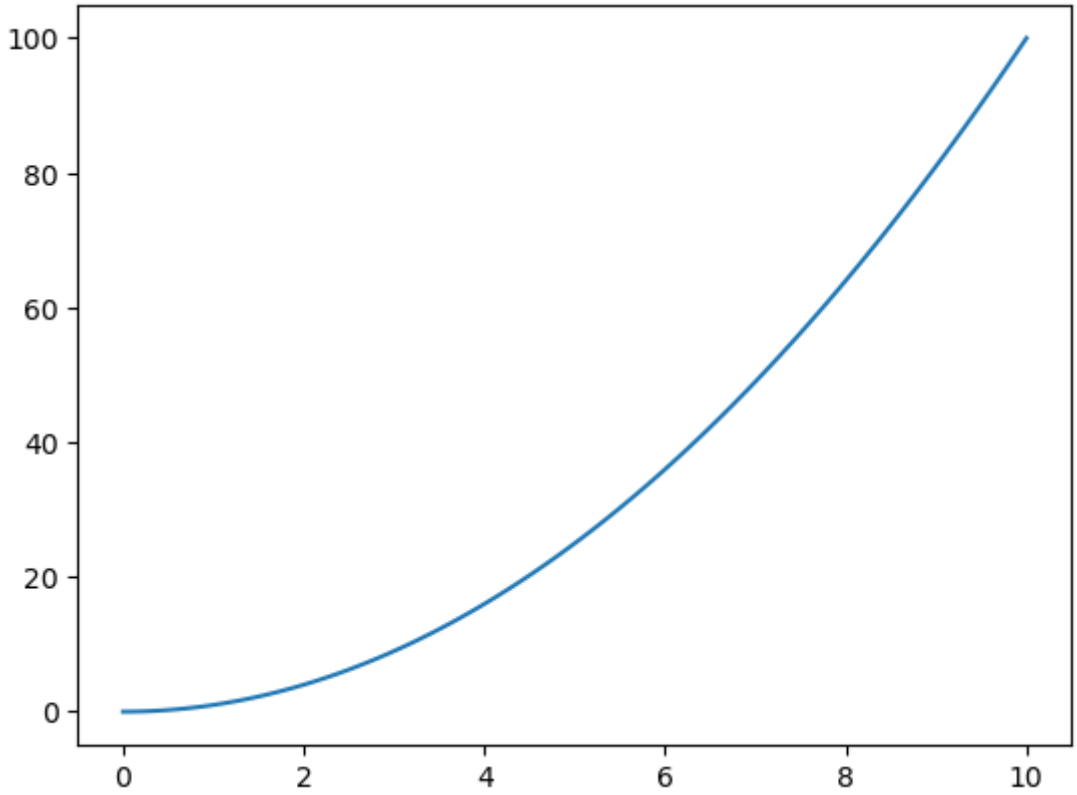

# making figure with numpy arrays

- Line Plot
- Scatter Plot
- Bar Plot
- Histogram
- Sub Plot

In [11]:
```
# create data
x = np.linspace(0,10,100)
x[:10]
```

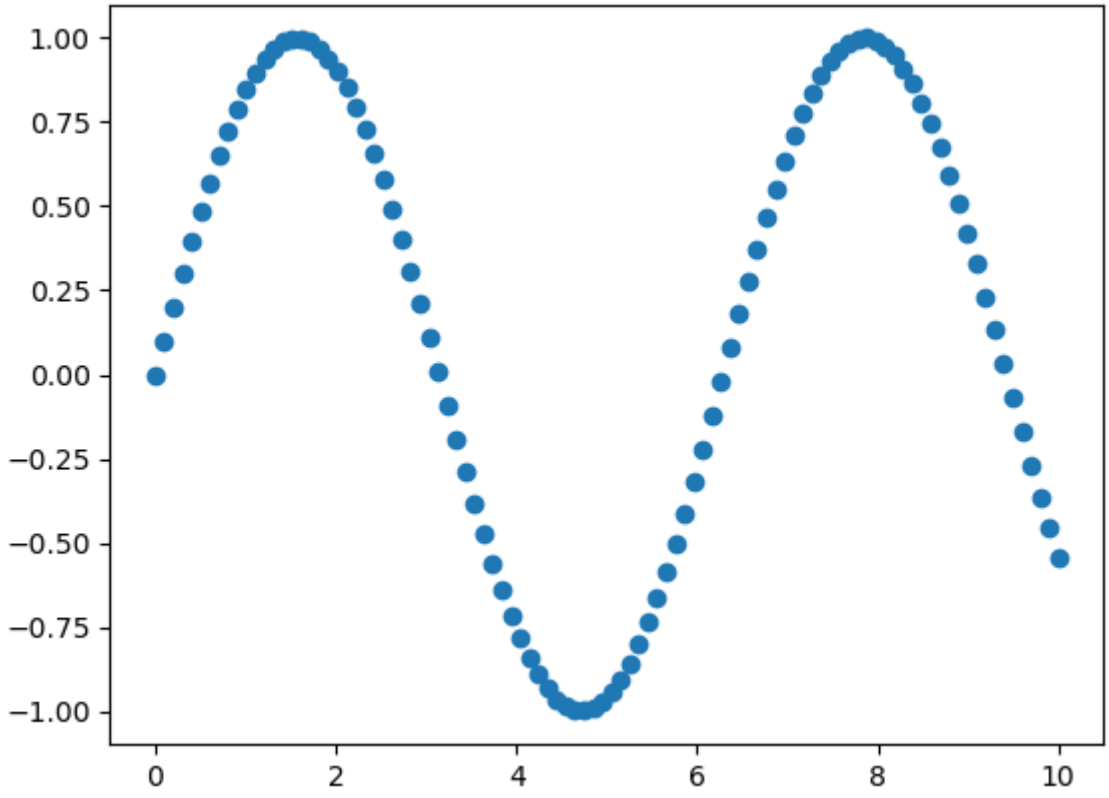
Out[11]:
```
array([0.        , 0.1010101 , 0.2020202 , 0.3030303 , 0.4040404 ,
       0.50505051, 0.60606061, 0.70707071, 0.80808081, 0.90909091])
```

In [12]:
```
# plot the data and create a line plot
fig, ax=plt.subplots()
ax.plot(x,x**2);
```



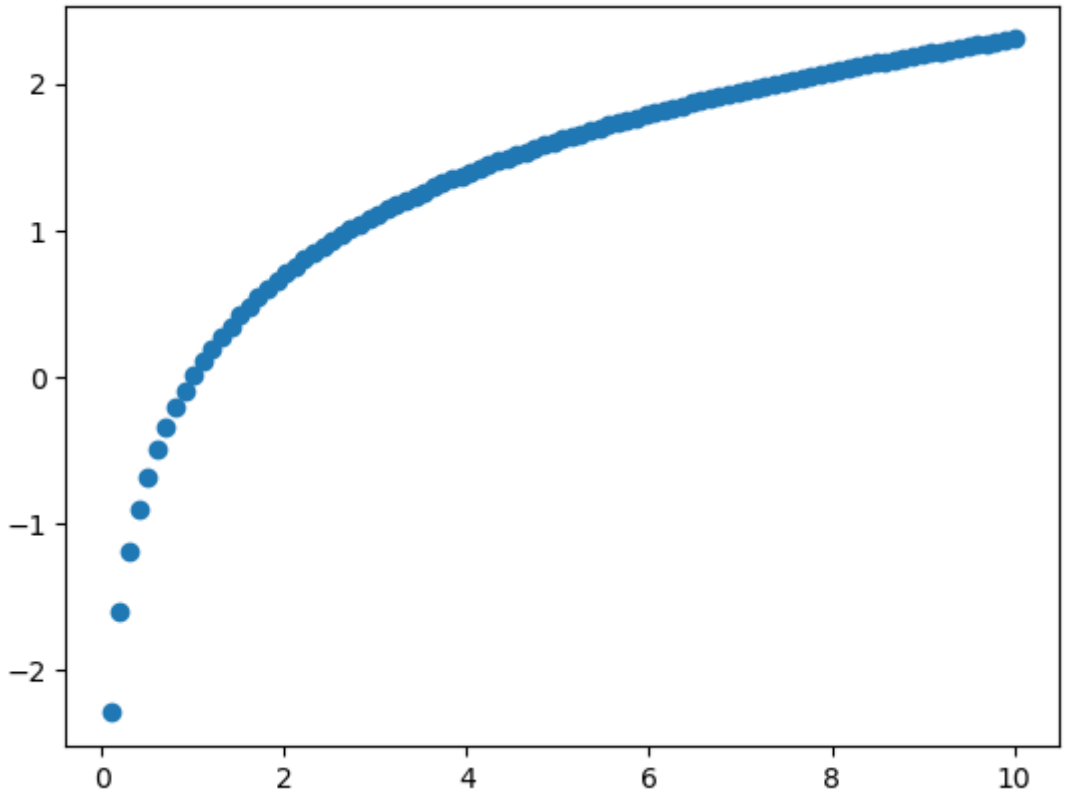

In [13]:
```
# use same data to create scatter plot
```

In [14]:
```
fig, ax = plt.subplots()
ax.scatter(x,np.sin(x));
```



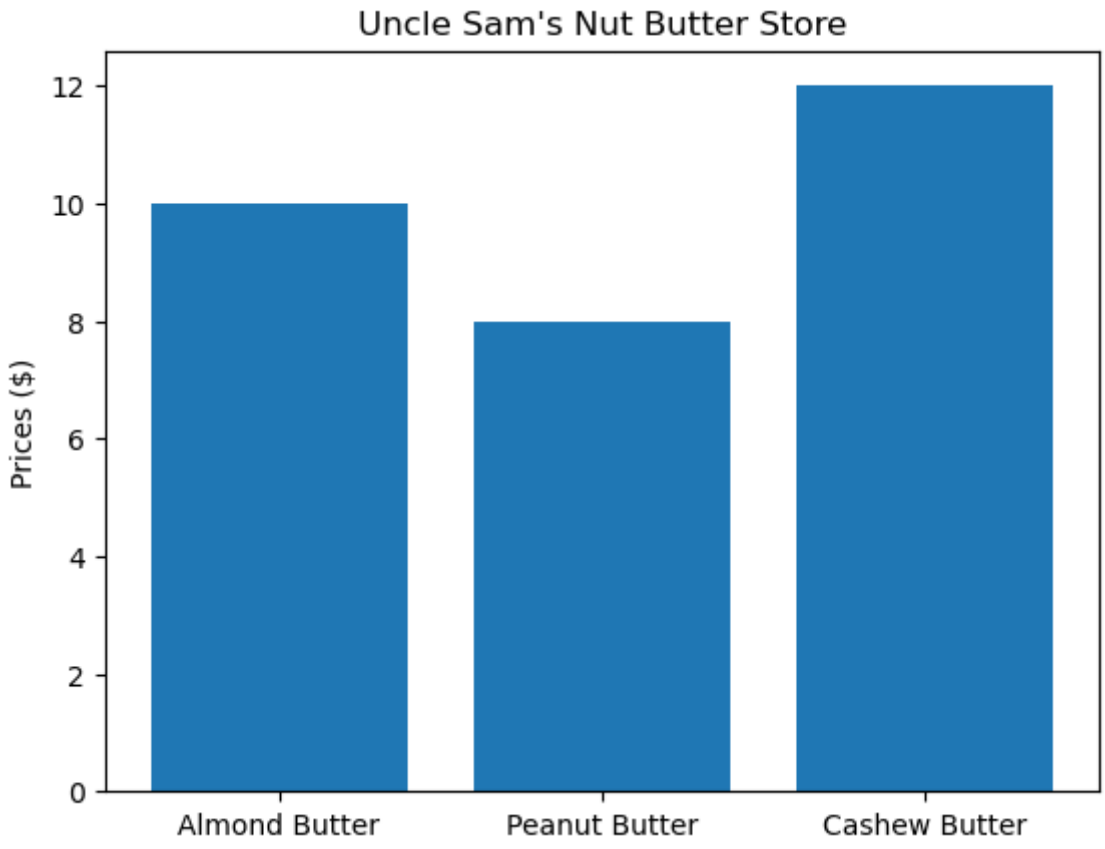

In [15]:
```
fig, ax = plt.subplots()
ax.scatter(x,np.log(x));
```

```
C:\Users\Hanu\AppData\Local\Temp\ipykernel_6560\1810901915.py:2: RuntimeWarning: d
ivide by zero encountered in log
  ax.scatter(x,np.log(x));
```




In [16]:
```
# create bar plot from dictionary
```

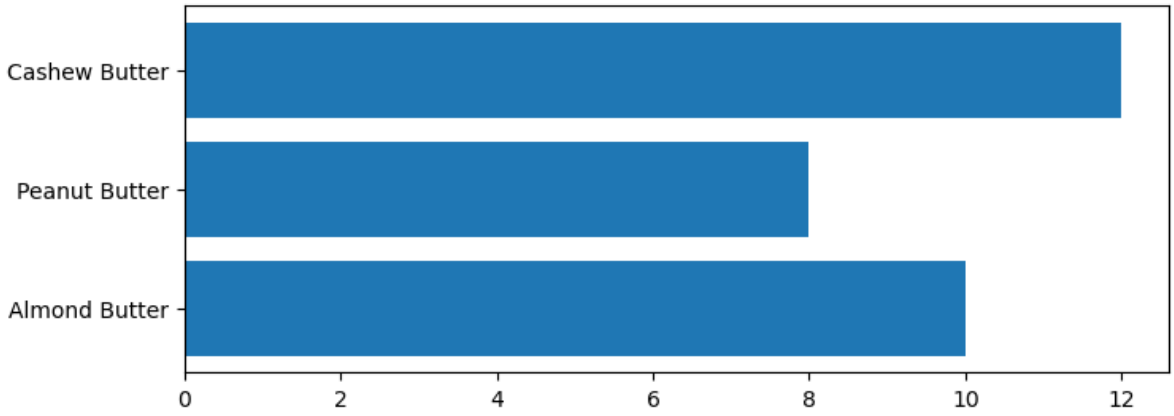
In [17]:
```
nut_butter_price = {"Almond Butter":10,"Peanut Butter":8,"Cashew Butter":12}
fig, ax=plt.subplots()
ax.bar(nut_butter_price.keys(),nut_butter_price.values())
ax.set(title="Uncle Sam's Nut Butter Store", ylabel="Prices ($)");
```



In [18]:
```
# Horizontal Bar Plot
```

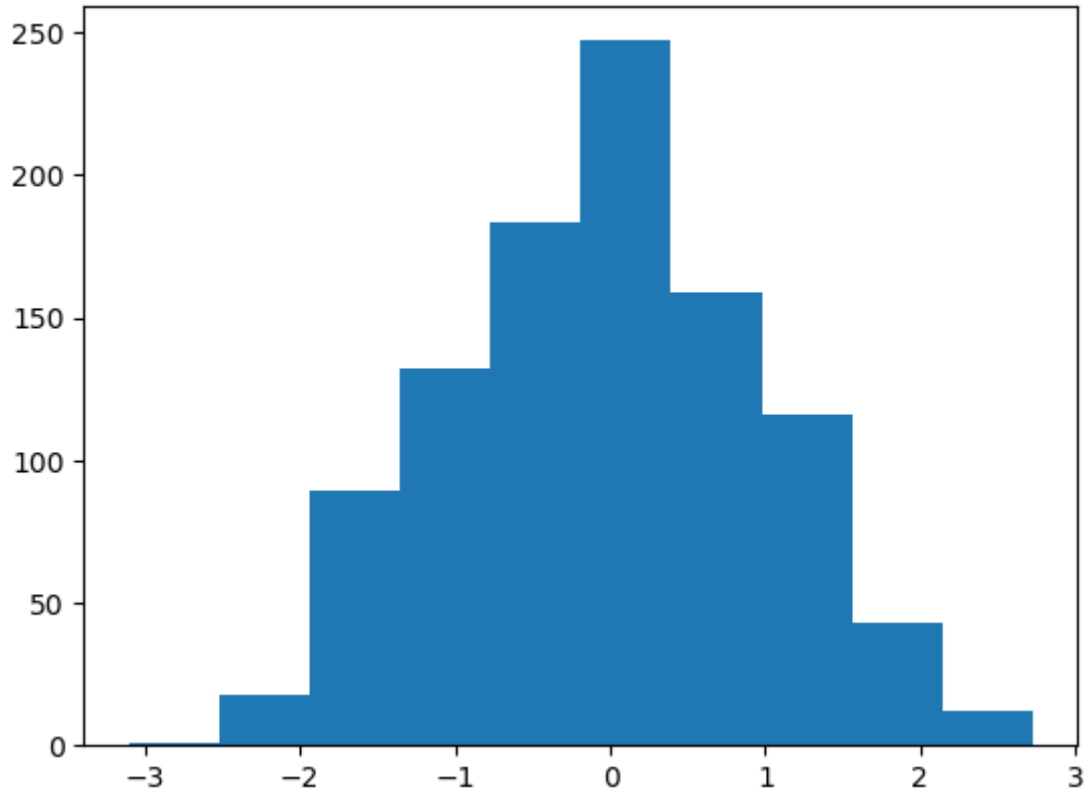
In [19]:
```
fig, ax= plt.subplots(figsize=(8,3))
ax.barh(list(nut_butter_price.keys()), list(nut_butter_price.values()))
```

Out[19]:
```
<BarContainer object of 3 artists>
```




In [20]:
```
# Histogram
```

In [21]:
```
hist_x = np.random.randn(1000)
fig, ax = plt.subplots()
ax.hist(hist_x);
```

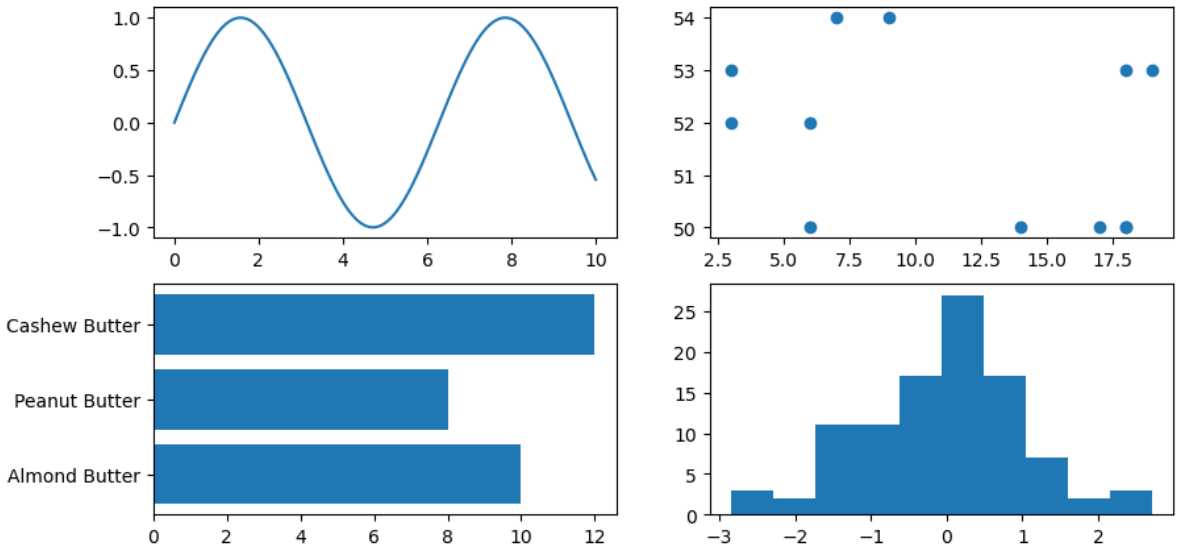



# Two options for subplots

In [22]:
```
# we are creating 4 axes in 1 figure
```

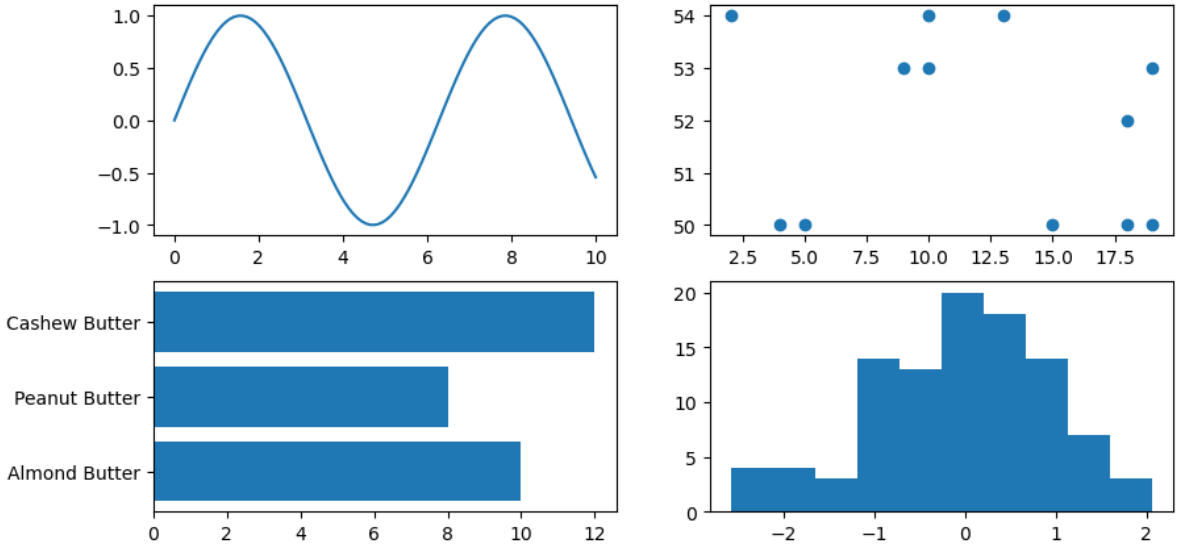
In [23]:
```
# 1st Option
#creating plots for every axes
```

```
fig, ((ax1,ax2),(ax3,ax4)) = plt.subplots(nrows=2, ncols=2, figsize=(10,5))
# plot the data in each axes (data can be different for every axes)
ax1.plot(x,np.sin(x));
ax2.scatter(np.random.randint(2,20,size=(3,4)),np.random.randint(50,55,size=(3,4))
ax3.barh(list(nut_butter_price.keys()),list(nut_butter_price.values()));
ax4.hist(np.random.randn(100));
```




In [24]:
```
# 2nd Option
```

In [25]:
```
fig, ax = plt.subplots(nrows=2,ncols=2,figsize=(10,5))

# plot data to each indexes
ax[0,0].plot(x,np.sin(x));
ax[0,1].scatter(np.random.randint(2,20,size=(3,4)),np.random.randint(50,55,size=(3
ax[1,0].barh(list(nut_butter_price.keys()),list(nut_butter_price.values()));
ax[1,1].hist(np.random.randn(100));
```




# Plotting from pandas dataframe

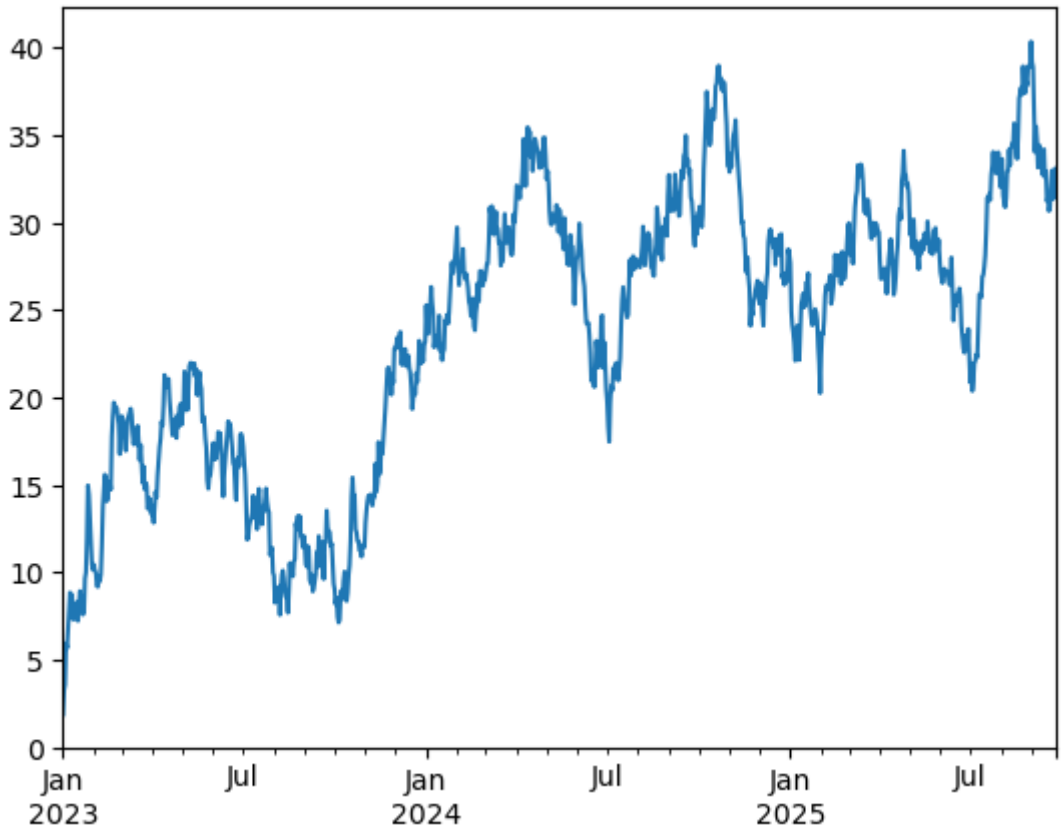
In [26]:
```
car_sales = pd.read_csv('car-sales.csv')
car_sales
```

Out[26]:

| | 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 [27]:
```
ts = pd.Series(np.random.randn(1000),index=pd.date_range("1/1/2023", periods=1000)
ts = ts.cumsum()
ts.plot()
```

Out[27]:
```
<AxesSubplot: >
```




In [28]:
```
# Replace the '$',',','.' sumbol from Price column
car_sales['Price'] = car_sales['Price'].str.replace('[\$\,\.]','')
car_sales
```

```
C:\Users\Hanu\AppData\Local\Temp\ipykernel_6560\3279740493.py:2: FutureWarning: Th
e default value of regex will change from True to False in a future version.
  car_sales['Price'] = car_sales['Price'].str.replace('[\$\,\.]','')
```

Out[28]:

| | 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 [29]:
```
type(car_sales['Price'])
```

Out[29]:
```
pandas.core.series.Series
```

In [30]:
```
# converting Price column into int type and removing values(zeros) after the decim
car_sales['Price'] = car_sales['Price'].astype(int)//100
car_sales
```

Out[30]:

| | 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 [31]:
```
# adding the car sales date in dataset
car_sales['Sale Date'] = pd.date_range('5/2/2023',periods=len(car_sales))
car_sales
```

Out[31]:

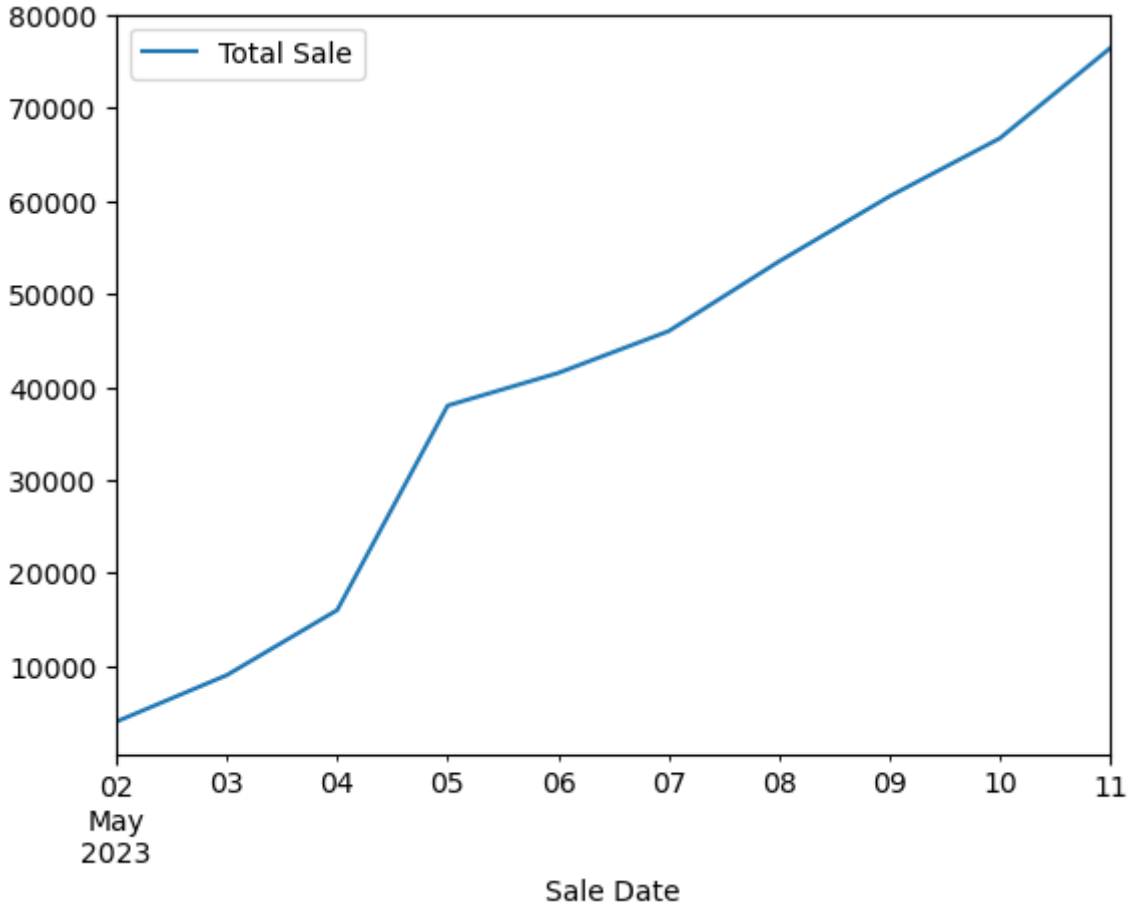
| | Make | Colour | Odometer (KM) | Doors | Price | Sale Date |
|---|---|---|---|---|---|---|
| **0** | Toyota | White | 150043 | 4 | 4000 | 2023-05-02 |
| **1** | Honda | Red | 87899 | 4 | 5000 | 2023-05-03 |
| **2** | Toyota | Blue | 32549 | 3 | 7000 | 2023-05-04 |
| **3** | BMW | Black | 11179 | 5 | 22000 | 2023-05-05 |
| **4** | Nissan | White | 213095 | 4 | 3500 | 2023-05-06 |
| **5** | Toyota | Green | 99213 | 4 | 4500 | 2023-05-07 |
| **6** | Honda | Blue | 45698 | 4 | 7500 | 2023-05-08 |
| **7** | Honda | Blue | 54738 | 4 | 7000 | 2023-05-09 |
| **8** | Toyota | White | 60000 | 4 | 6250 | 2023-05-10 |
| **9** | Nissan | White | 31600 | 4 | 9700 | 2023-05-11 |

In [32]:
```
# creating a column name  Total sale and putting total sale till each day
car_sales['Total Sale'] = car_sales['Price'].cumsum()
car_sales
```

Out[32]:

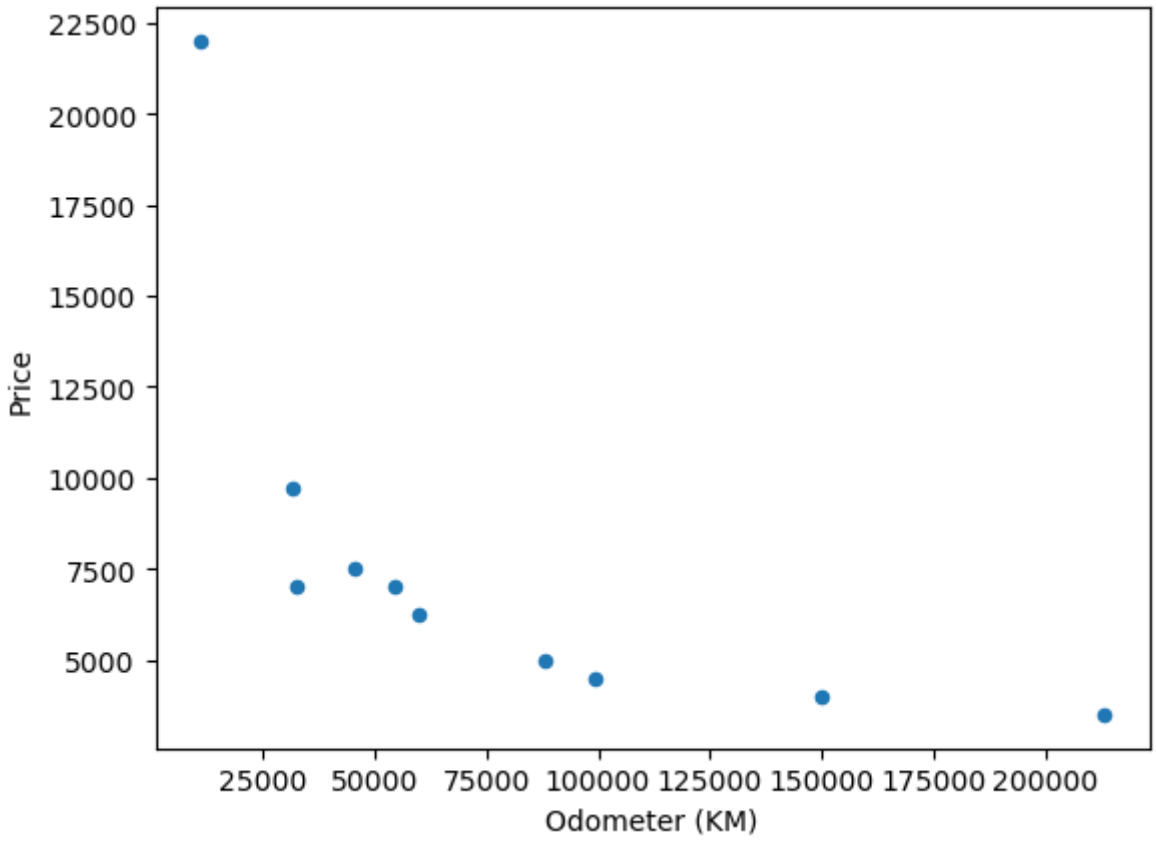
| | Make | Colour | Odometer (KM) | Doors | Price | Sale Date | Total Sale |
|---|---|---|---|---|---|---|---|
| **0** | Toyota | White | 150043 | 4 | 4000 | 2023-05-02 | 4000 |
| **1** | Honda | Red | 87899 | 4 | 5000 | 2023-05-03 | 9000 |
| **2** | Toyota | Blue | 32549 | 3 | 7000 | 2023-05-04 | 16000 |
| **3** | BMW | Black | 11179 | 5 | 22000 | 2023-05-05 | 38000 |
| **4** | Nissan | White | 213095 | 4 | 3500 | 2023-05-06 | 41500 |
| **5** | Toyota | Green | 99213 | 4 | 4500 | 2023-05-07 | 46000 |
| **6** | Honda | Blue | 45698 | 4 | 7500 | 2023-05-08 | 53500 |
| **7** | Honda | Blue | 54738 | 4 | 7000 | 2023-05-09 | 60500 |
| **8** | Toyota | White | 60000 | 4 | 6250 | 2023-05-10 | 66750 |
| **9** | Nissan | White | 31600 | 4 | 9700 | 2023-05-11 | 76450 |

In [33]:
```
#line plot to see relation between 'sale date' and 'total sale' & analyze the patte
car_sales.plot(x='Sale Date', y='Total Sale');
```




In [34]:
```
# scatter plot to see realtion between odometer and price
car_sales.plot(x='Odometer (KM)',y='Price',kind='scatter')
```

Out[34]:
```
<AxesSubplot: xlabel='Odometer (KM)', ylabel='Price'>
```

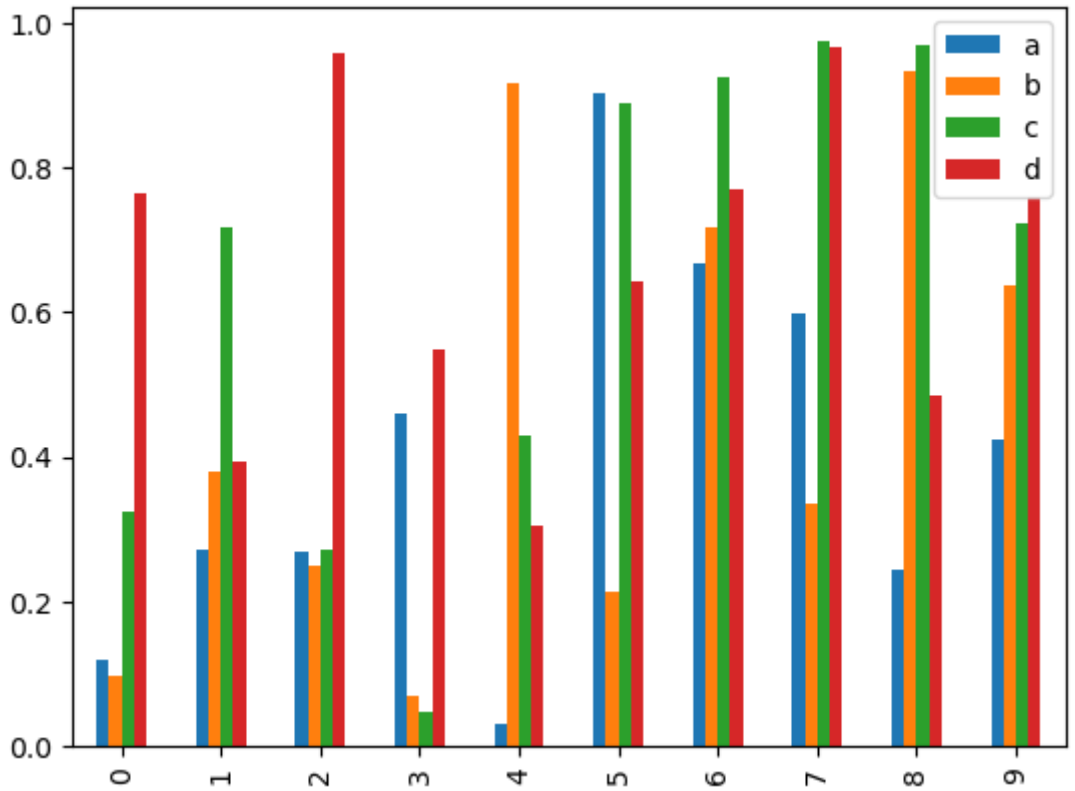


In [35]:
```
x2 = np.random.rand(10,4)
df = pd.DataFrame(x2,columns=['a','b','c','d'])
df
```

Out[35]:

| | a | b | c | d |
|---|---|---|---|---|
| **0** | 0.118778 | 0.098725 | 0.325350 | 0.765079 |
| **1** | 0.271229 | 0.380128 | 0.716626 | 0.393334 |
| **2** | 0.270605 | 0.250901 | 0.271057 | 0.957736 |
| **3** | 0.460654 | 0.069335 | 0.047502 | 0.547518 |
| **4** | 0.031802 | 0.917140 | 0.430924 | 0.306114 |
| **5** | 0.904047 | 0.213794 | 0.888466 | 0.643919 |
| **6** | 0.668035 | 0.717123 | 0.923977 | 0.771109 |
| **7** | 0.597353 | 0.335996 | 0.973540 | 0.967134 |
| **8** | 0.244268 | 0.932493 | 0.969402 | 0.483921 |
| **9** | 0.425513 | 0.638445 | 0.724061 | 0.757718 |

In [36]:
```
df.plot.bar();
```



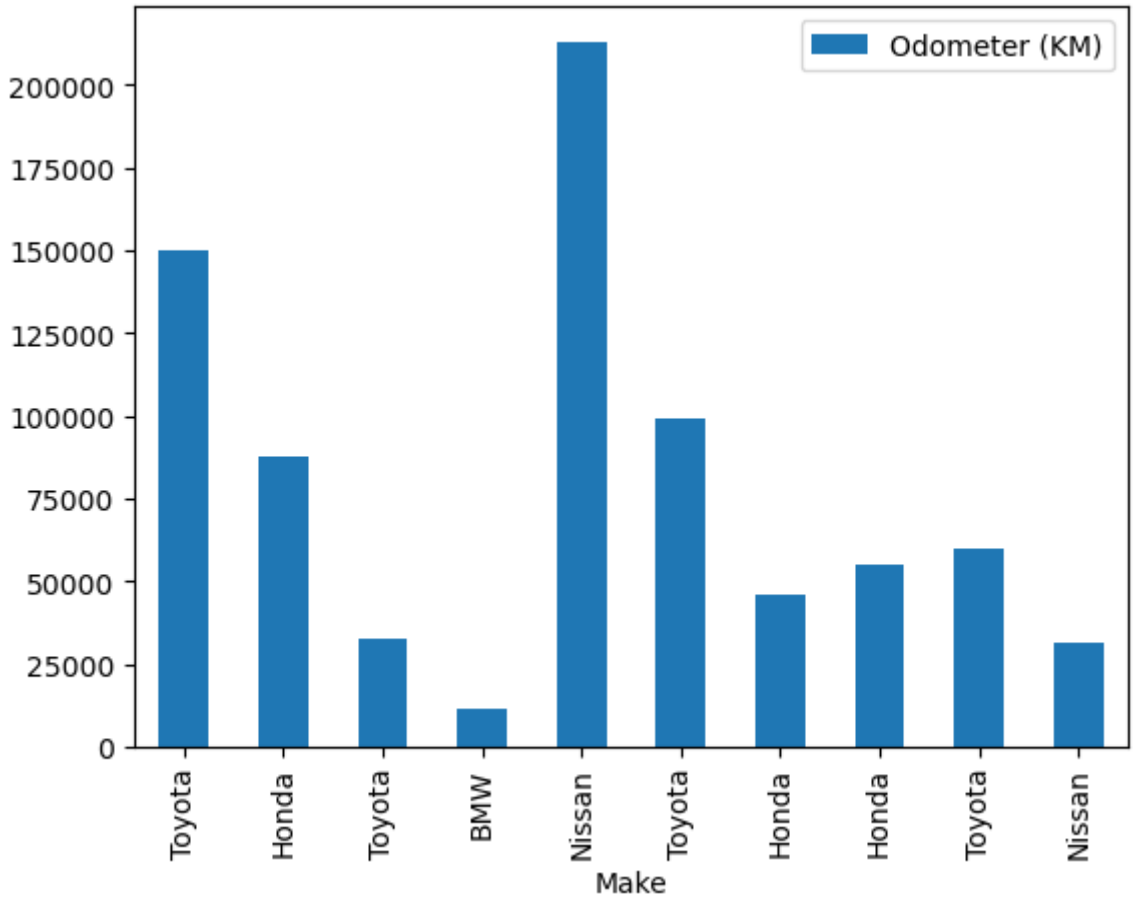

In [37]:
```
car_sales
```

Out[37]:

| | Make | Colour | Odometer (KM) | Doors | Price | Sale Date | Total Sale |
|---|---|---|---|---|---|---|---|
| **0** | Toyota | White | 150043 | 4 | 4000 | 2023-05-02 | 4000 |
| **1** | Honda | Red | 87899 | 4 | 5000 | 2023-05-03 | 9000 |
| **2** | Toyota | Blue | 32549 | 3 | 7000 | 2023-05-04 | 16000 |
| **3** | BMW | Black | 11179 | 5 | 22000 | 2023-05-05 | 38000 |
| **4** | Nissan | White | 213095 | 4 | 3500 | 2023-05-06 | 41500 |
| **5** | Toyota | Green | 99213 | 4 | 4500 | 2023-05-07 | 46000 |
| **6** | Honda | Blue | 45698 | 4 | 7500 | 2023-05-08 | 53500 |
| **7** | Honda | Blue | 54738 | 4 | 7000 | 2023-05-09 | 60500 |
| **8** | Toyota | White | 60000 | 4 | 6250 | 2023-05-10 | 66750 |
| **9** | Nissan | White | 31600 | 4 | 9700 | 2023-05-11 | 76450 |

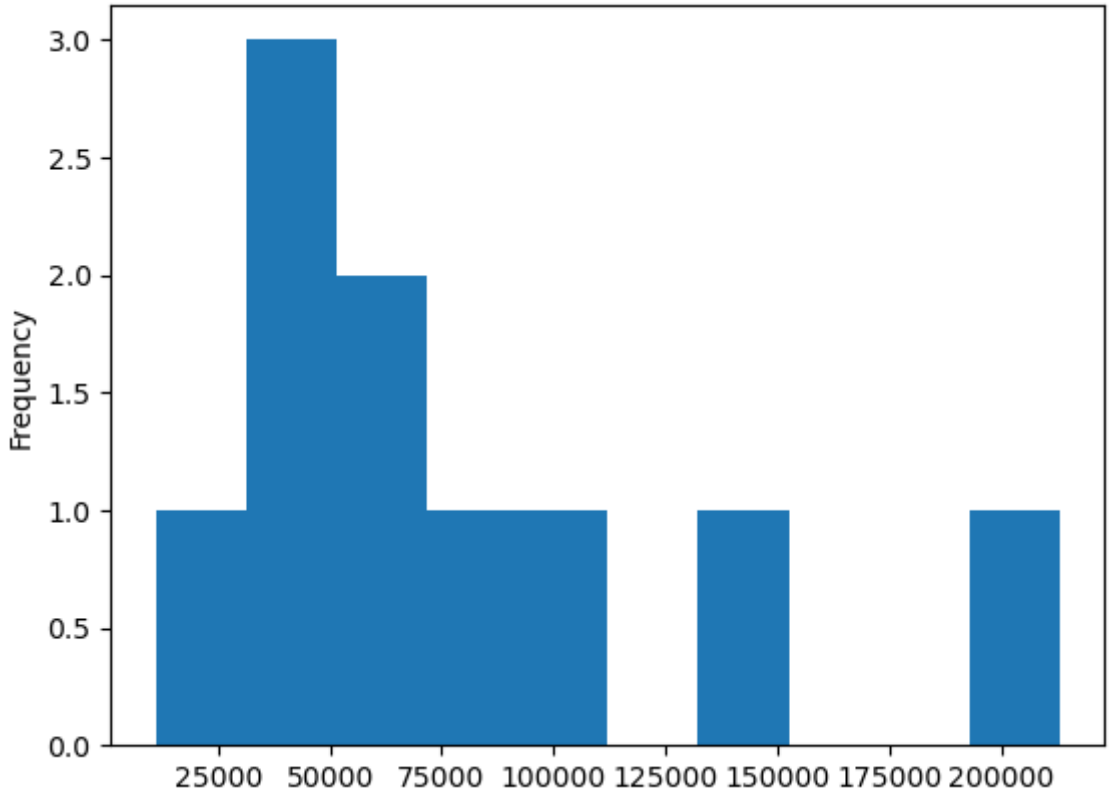
In [38]:
```
car_sales.plot(x='Make',y='Odometer (KM)',kind='bar')
```

Out[38]:
```
<AxesSubplot: xlabel='Make'>
```



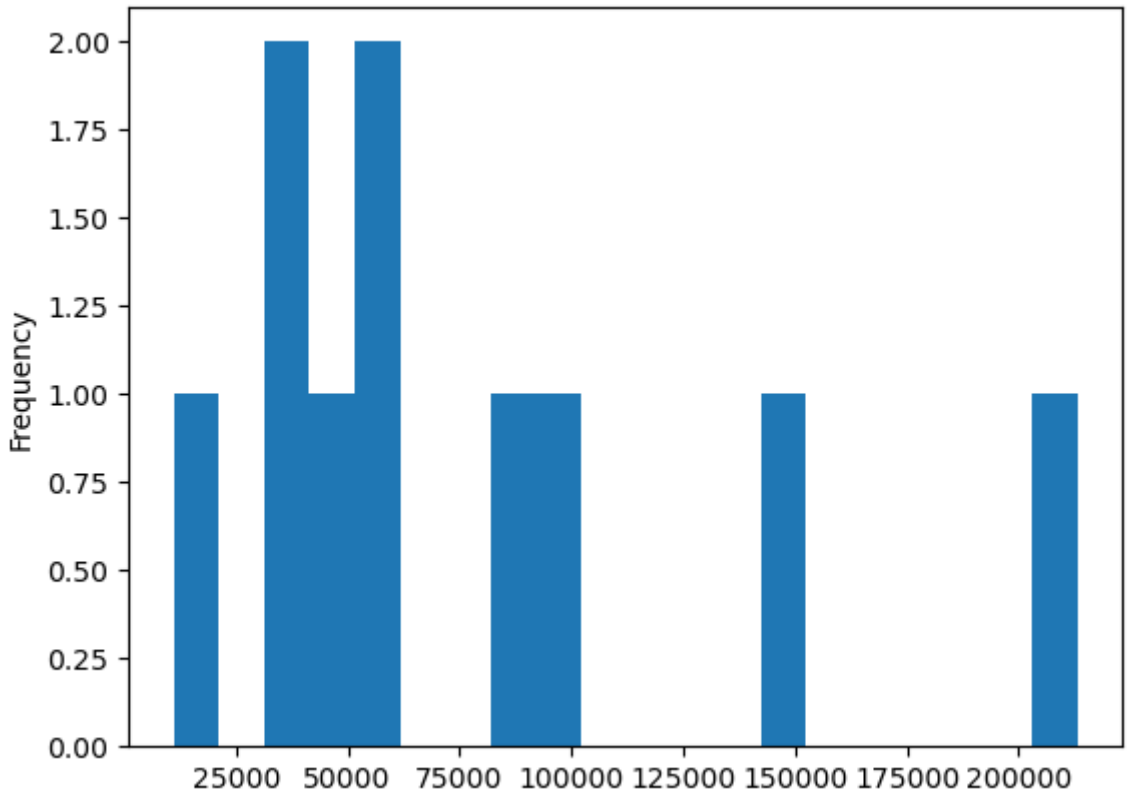

In [39]:
```
car_sales['Odometer (KM)'].plot.hist()
```

Out[39]:
```
<AxesSubplot: ylabel='Frequency'>
```




In [40]:
```
car_sales['Odometer (KM)'].plot.hist(bins=20)
```

Out[40]:
```
<AxesSubplot: ylabel='Frequency'>
```




In [41]:
```
#Try another dataset
```

In [42]:
```
heart_disease = pd.read_csv('heart-disease.csv')
heart_disease
```

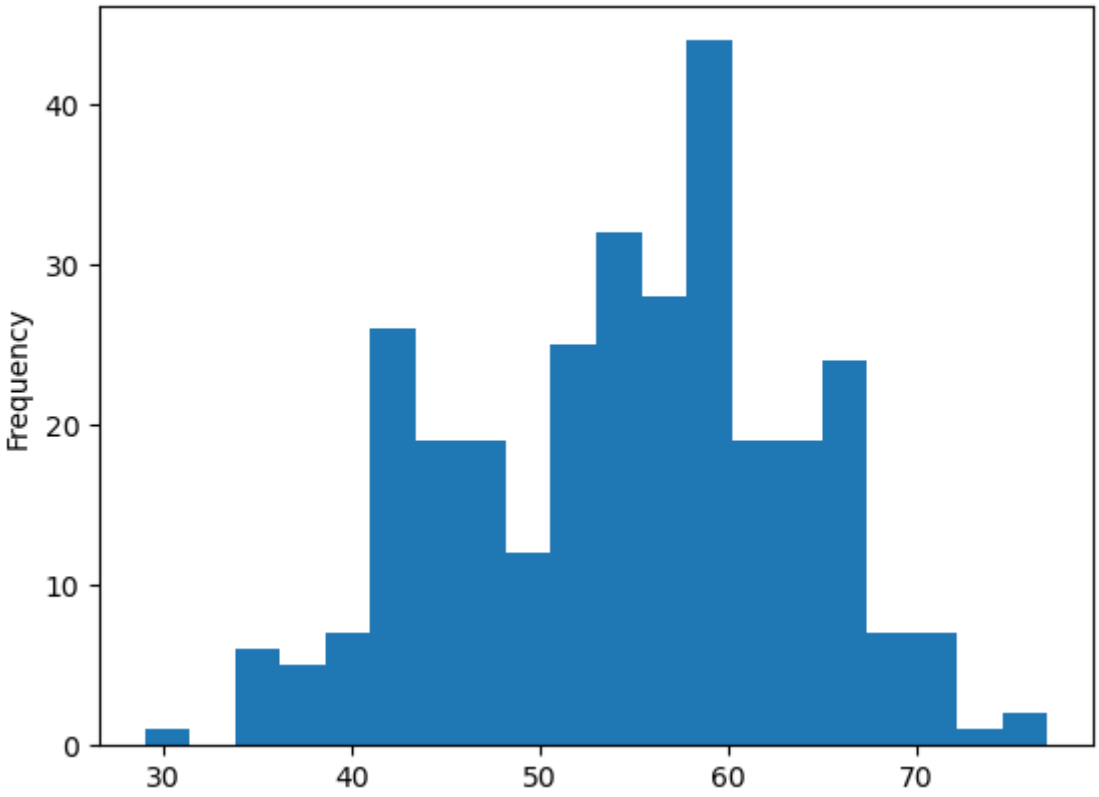
Out[42]:

| | 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 |

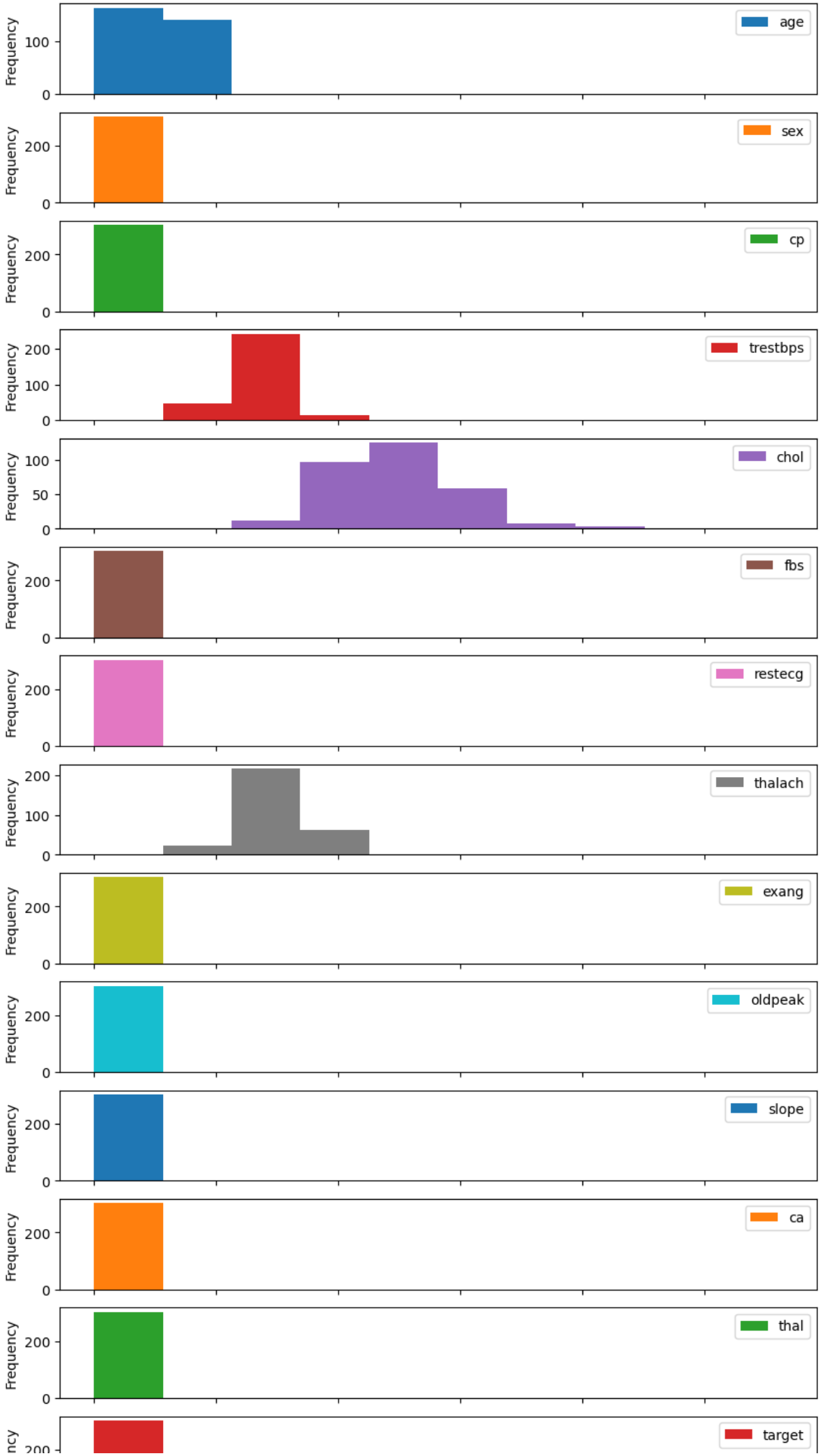
303 rows × 14 columns

In [43]:
```
heart_disease['age'].plot.hist(bins=20)
```

Out[43]:
```
<AxesSubplot: ylabel='Frequency'>
```




In [44]:
```
heart_disease.plot.hist(figsize=(10,20),subplots=True);
```


Frequency
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target

# Two type of methods Pyplot and OO Method

## which one should you use ?

### -> when you have to plot something quickly, use pyplot Method

### -> when you have to plot something advanced, use OO Method

In [45]:
```
heart_disease.head()
```

Out[45]:

| | 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 |

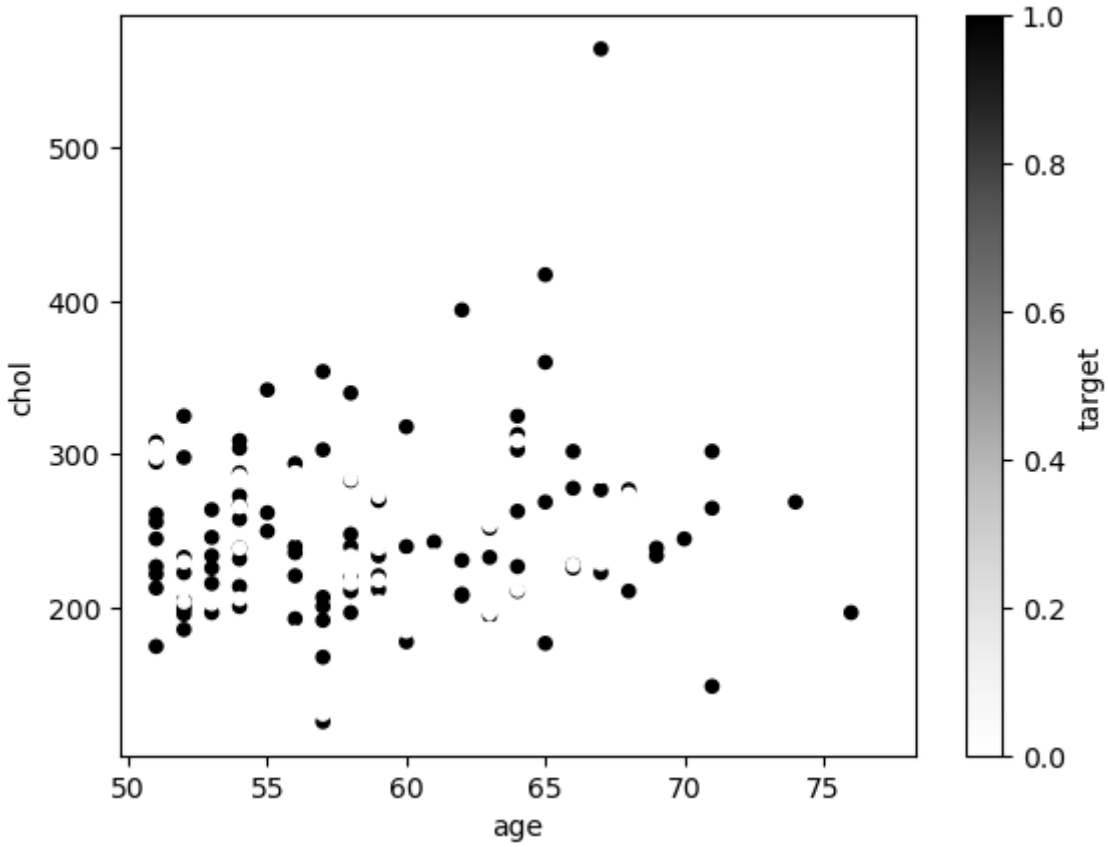
In [46]:
```
over_50 = heart_disease[heart_disease['age']>50]
over_50
```

Out[46]:

| | 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 |

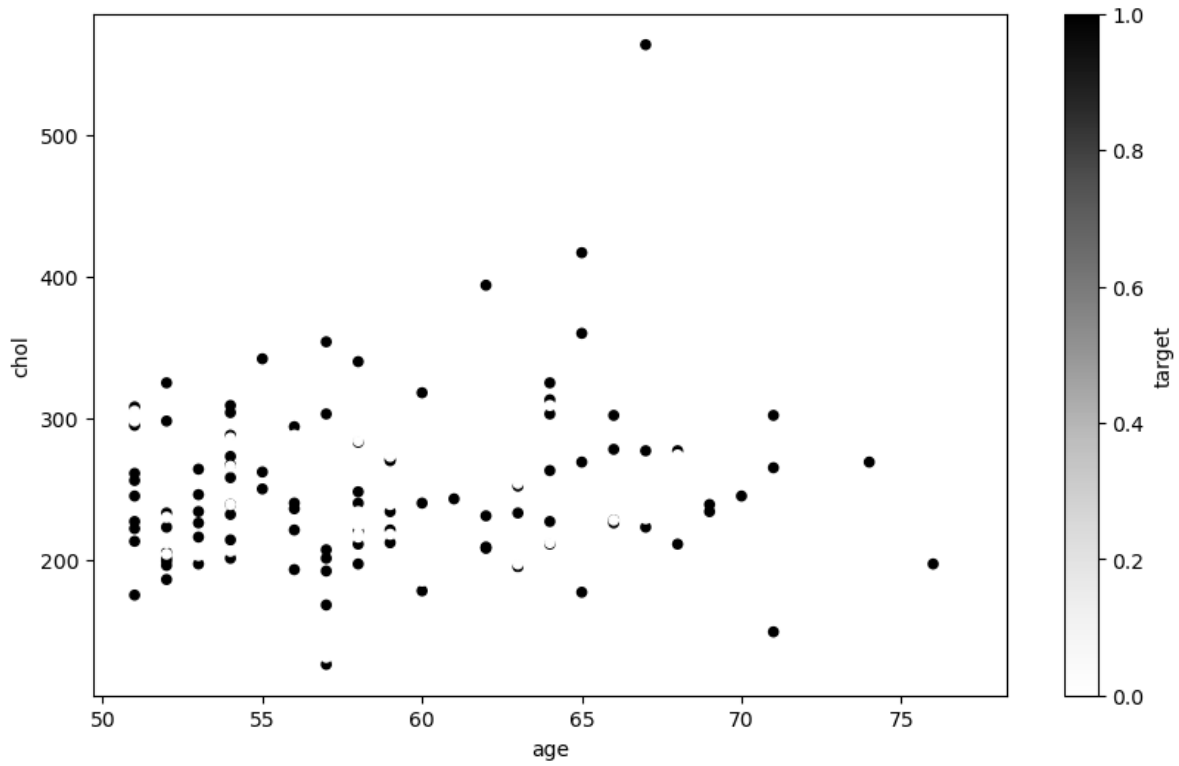
208 rows × 14 columns

In [47]:
```
# scatter plot of age and cholestrol
# pyplot method
over_50.plot(kind='scatter',x='age',y='chol',c='target');
```

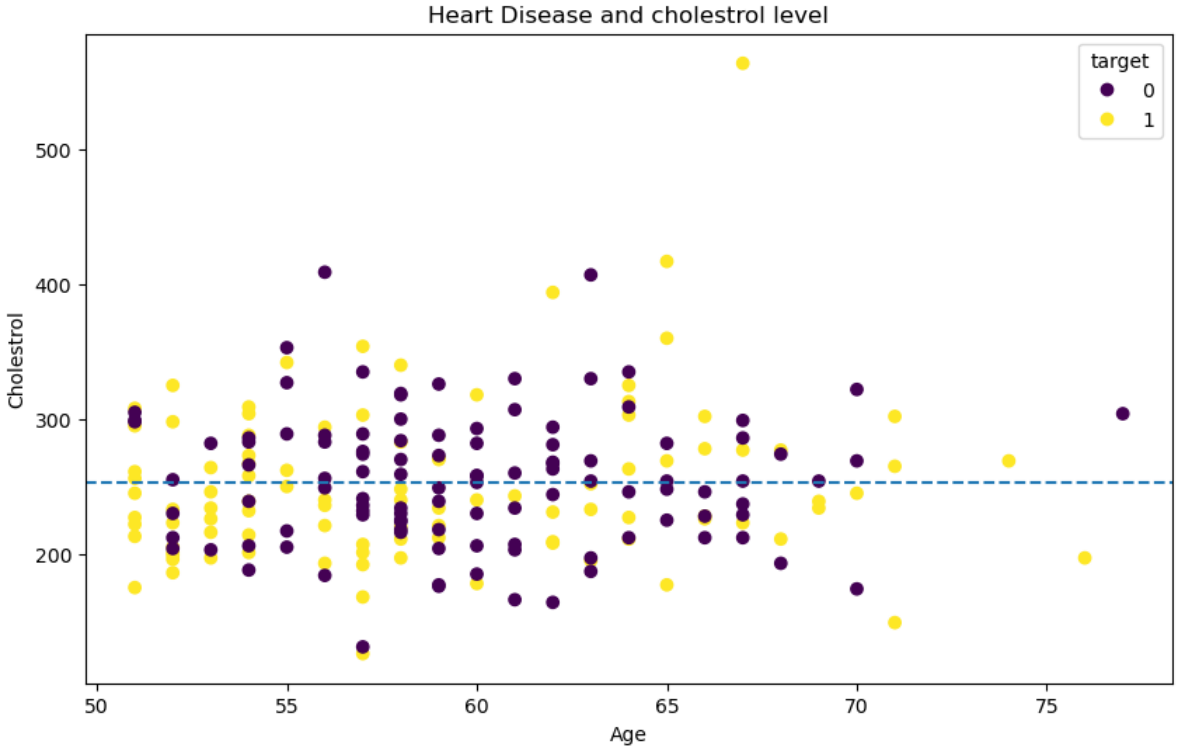



In [48]:
```
# By OO Method mix with pyplot
fig, ax_1 = plt.subplots(figsize=(10,6))
over_50.plot(kind='scatter', x='age',y='chol',c='target',ax=ax_1)
#ax_1.set_xlim([45,100])  #it extend the limit of axes
```

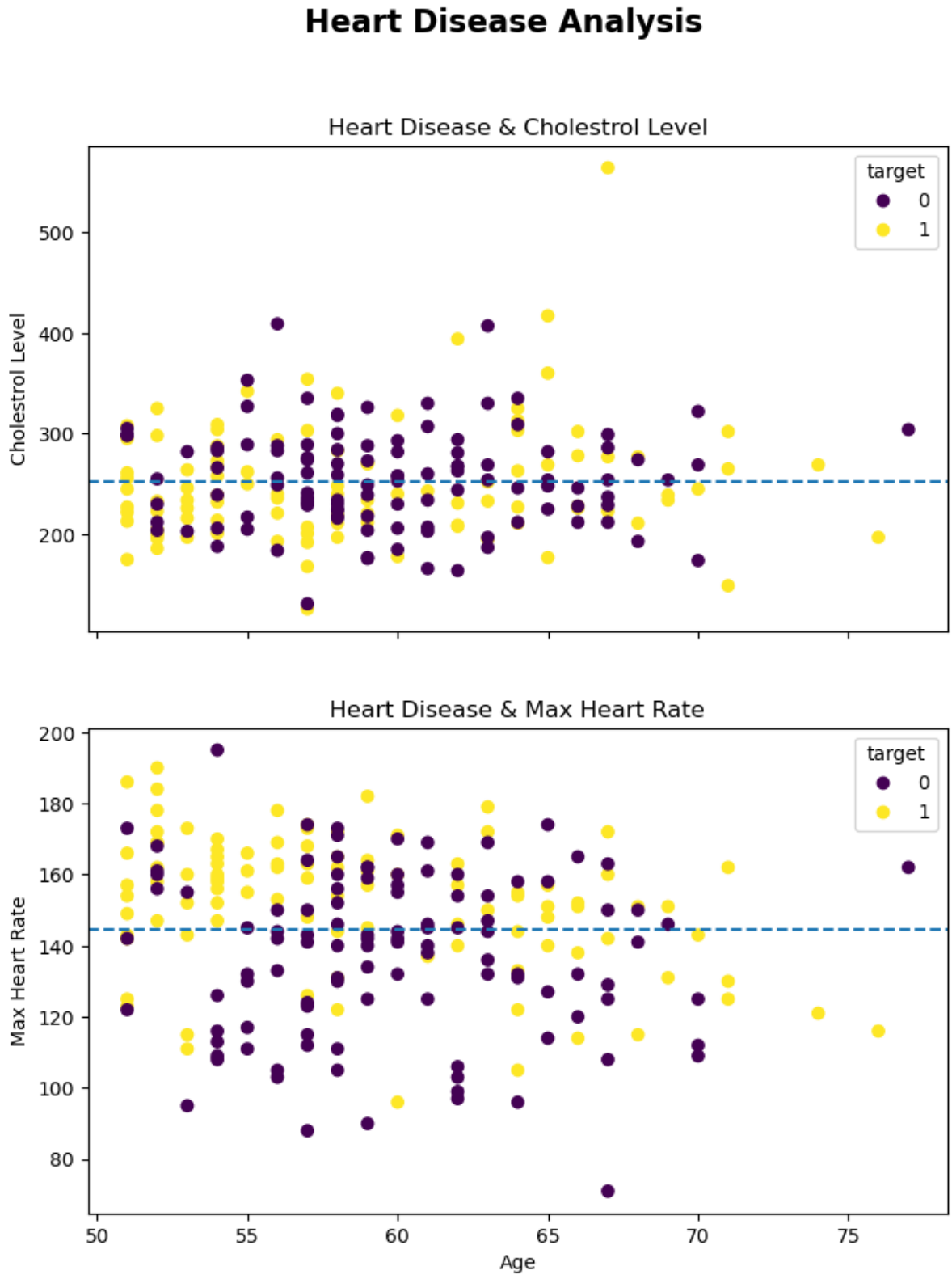
Out[48]:
```
<AxesSubplot: xlabel='age', ylabel='chol'>
```




In [49]:
```
#OO Method from scratch

# create a axes
fig, ax=plt.subplots(figsize=(10,6))

#plot the data on axes
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 level',xlabel='Age',ylabel='Cholestrol'

# add legends to make it more understandable
ax.legend(*scatter.legend_elements(),title='target')

# add horizontal line in axes which shows average value ,, we can also use 'dashed
ax.axhline(over_50['chol'].mean(), linestyle='--')
plt.show()
```




In [50]:
```
# subplot of chol, age, thalach

#creating axes of subplots
fig, (ax1,ax2) = plt.subplots(nrows=2,ncols=1,figsize=(8,10),sharex=True)

# plot data on both axes
scatter_1 =ax1.scatter(x=over_50['age'],y=over_50['chol'],c=over_50['target'])
scatter_2 = ax2.scatter(x=over_50['age'],y=over_50['thalach'],c=over_50['target'])

# Customize the plot
ax1.set(title='Heart Disease & Cholestrol Level',ylabel='Cholestrol Level')
ax2.set(title='Heart Disease & Max Heart Rate',xlabel='Age',ylabel='Max Heart Rate

#show legend on plot so that it will be easily understandable
ax1.legend(*scatter_1.legend_elements(),title='target')
ax2.legend(*scatter_2.legend_elements(),title='target')

#create a horizontal line which represent the mean of cholestrol and max heart rate
ax1.axhline(over_50['chol'].mean(),linestyle='--')
ax2.axhline(over_50['thalach'].mean(),linestyle='--')

#create a title of figure
fig.suptitle('Heart Disease Analysis', fontsize='16',fontweight='bold')
plt.show()
```

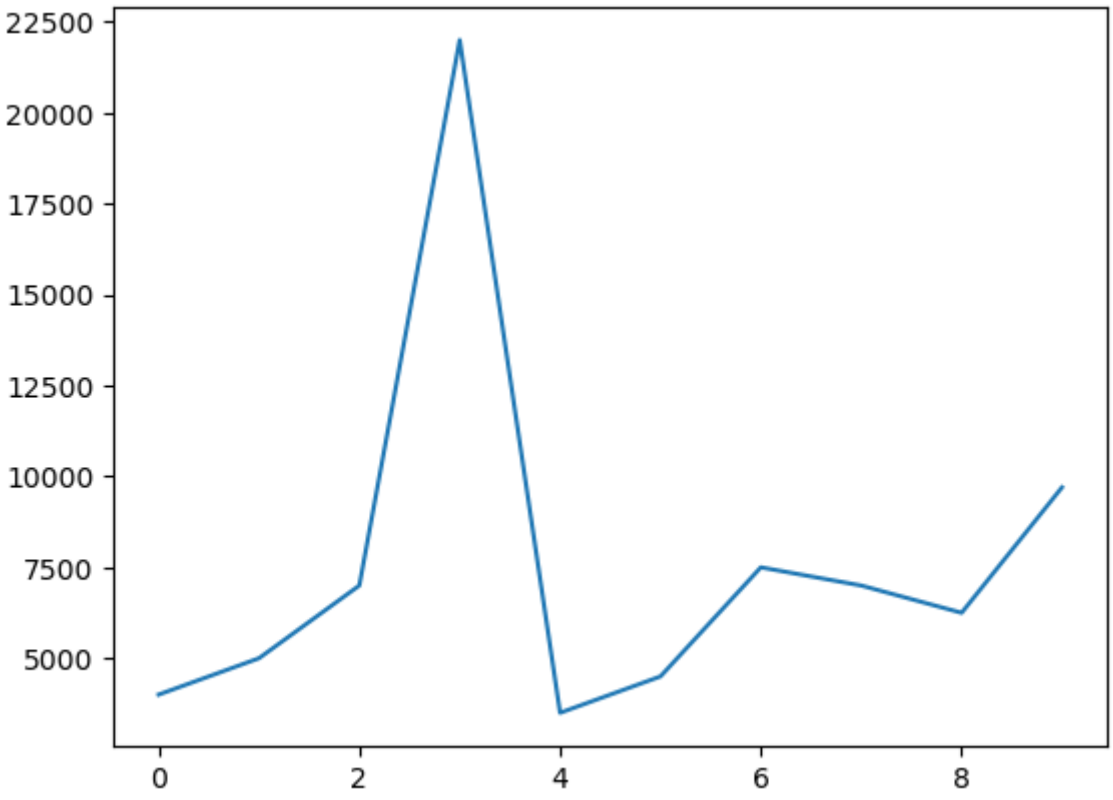



# Customizing Matplotlib plot and getting stylish

In [51]:
```
# see different style availabel
plt.style.available
```

Out[51]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
 'seaborn-v0_8-muted',
 'seaborn-v0_8-notebook',
 'seaborn-v0_8-paper',
 'seaborn-v0_8-pastel',
 'seaborn-v0_8-poster',
 'seaborn-v0_8-talk',
 'seaborn-v0_8-ticks',
 'seaborn-v0_8-white',
 'seaborn-v0_8-whitegrid',
 'tableau-colorblind10']
```

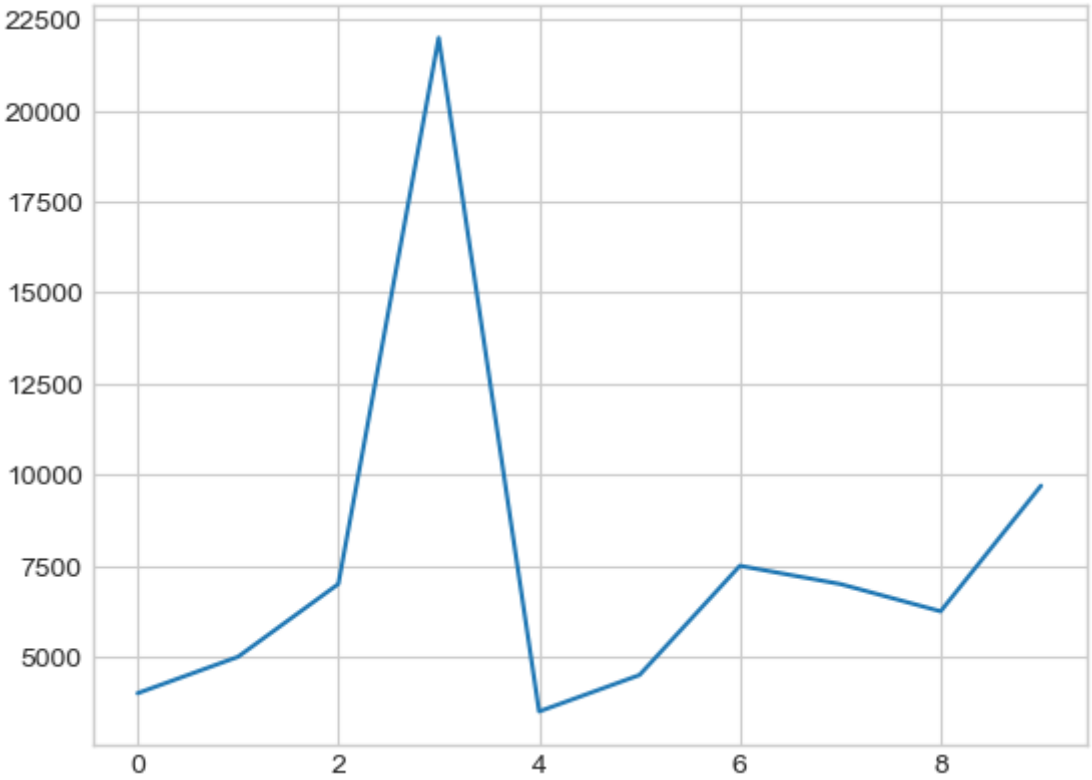
In [54]:
```
car_sales['Price'].plot();
```




In [61]:
```
# after changing style
plt.style.use('seaborn-whitegrid')
```

```
C:\Users\Hanu\AppData\Local\Temp\ipykernel_6560\196564673.py:2: MatplotlibDeprecat
ionWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as
they no longer correspond to the styles shipped by seaborn. However, they will rem
ain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn A
PI instead.
  plt.style.use('seaborn-whitegrid')
```

In [62]:
```
car_sales['Price'].plot();
```

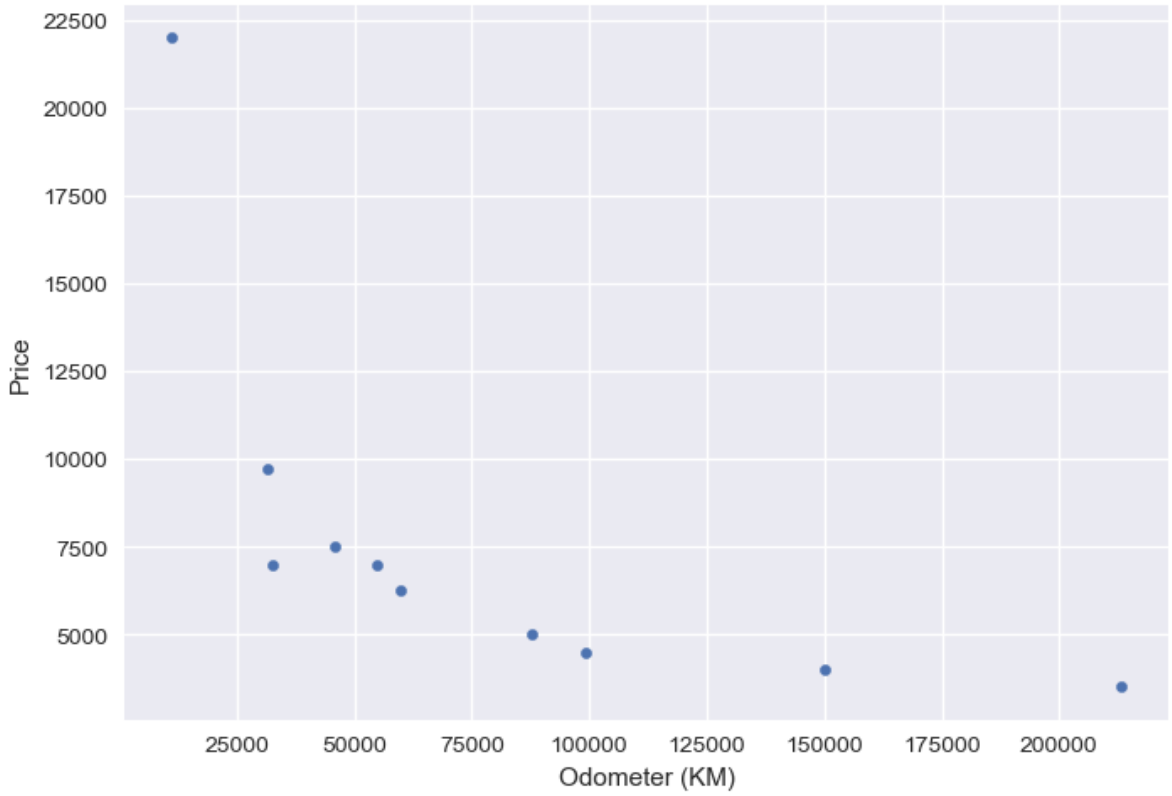



In [63]:
```
plt.style.use('seaborn')
```

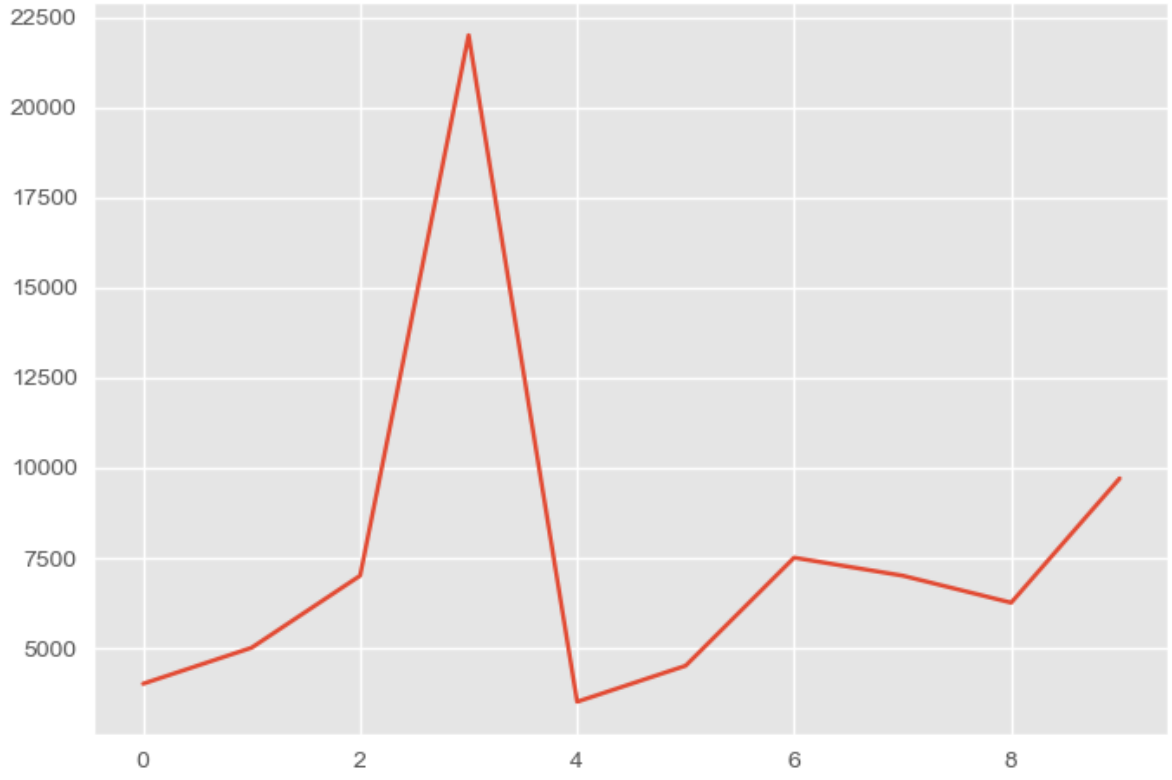
```
C:\Users\Hanu\AppData\Local\Temp\ipykernel_6560\240305066.py:1: MatplotlibDeprecat
ionWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as
they no longer correspond to the styles shipped by seaborn. However, they will rem
ain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn A
PI instead.
  plt.style.use('seaborn')
```

In [64]:
```
car_sales.plot(x='Odometer (KM)',y='Price',kind='scatter')
```

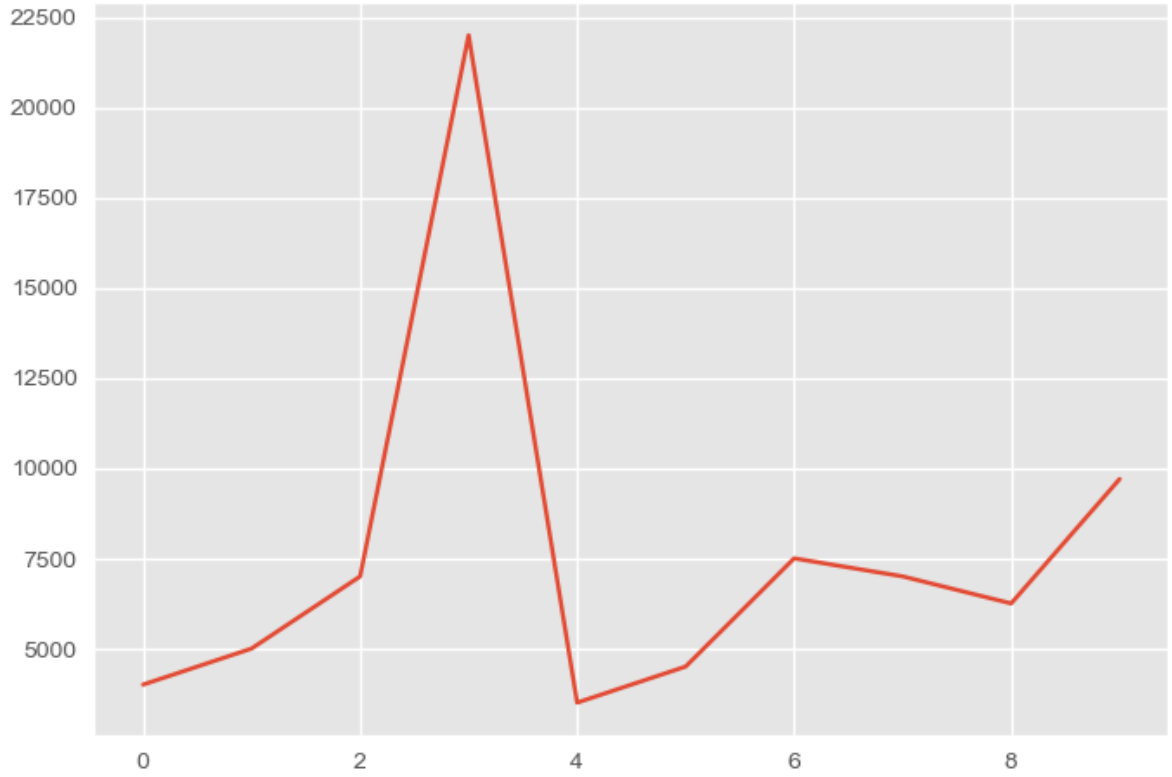
Out[64]:
```
<AxesSubplot: xlabel='Odometer (KM)', ylabel='Price'>
```




In [66]:
```
plt.style.use('ggplot')
car_sales['Price'].plot();
```




In [67]:
```
plt.style.use('fast')
car_sales['Price'].plot();
```




In [69]:
```
# create some data
Sample_data = np.random.randn(10,4)
Sample_data
```

Out[69]:
```
array([[ 0.13250066,  0.47342855,  0.60347372, -0.59506293],
       [ 1.12096814, -0.32141256,  1.32250754,  1.2509432 ],
       [ 0.44165505,  0.97607372, -0.96017013, -0.04773204],
       [ 0.37004709, -0.09580748, -1.33586752, -0.13778895],
       [ 0.4019787 ,  0.35529778, -0.65266999,  0.31771859],
       [-0.89484067,  0.1970945 , -1.83210869,  1.97093601],
       [ 0.48060277, -1.45938723, -0.31646659,  0.3919833 ],
       [ 0.62462323,  0.9111005 , -0.98715017, -0.03913643],
       [-0.92892099, -2.79972635,  0.67131166, -0.04888169],
       [-2.05078488,  1.09007995, -0.38350891, -0.89234433]])
```

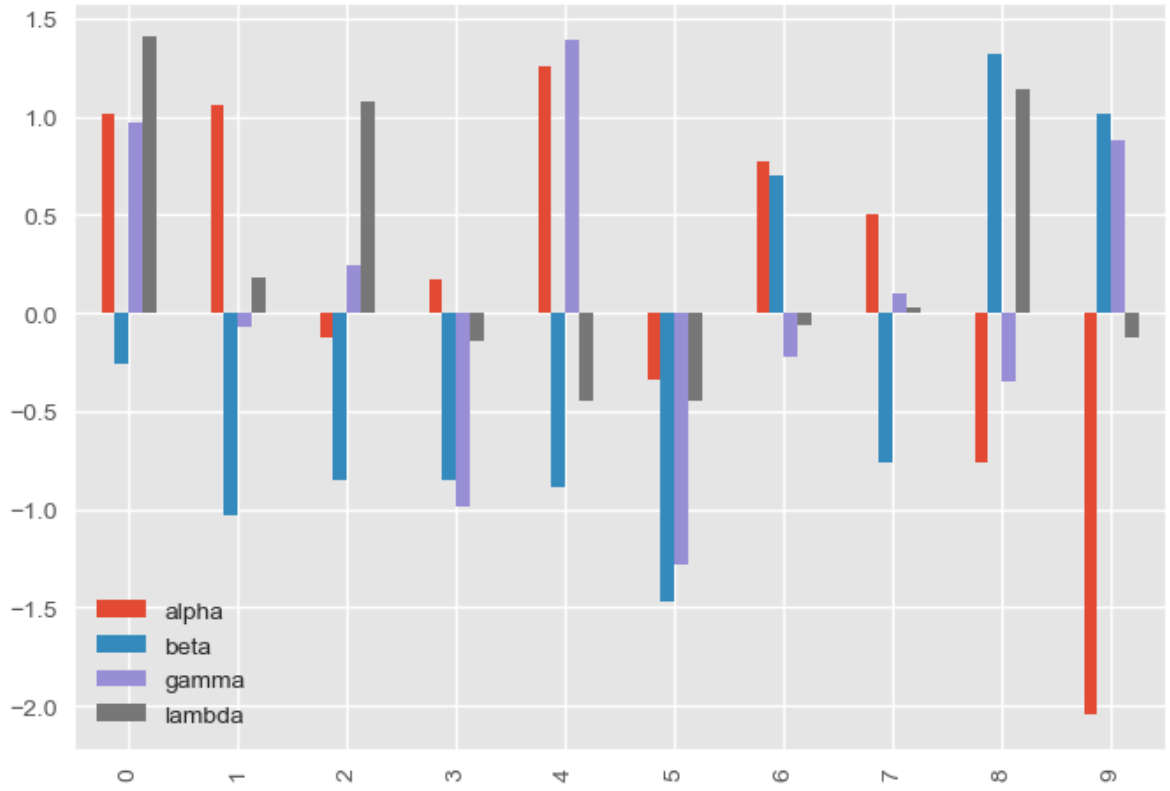
In [72]:
```
Sample_df = pd.DataFrame(x,columns=['alpha','beta','gamma','lambda'])
Sample_df
```

Out[72]:

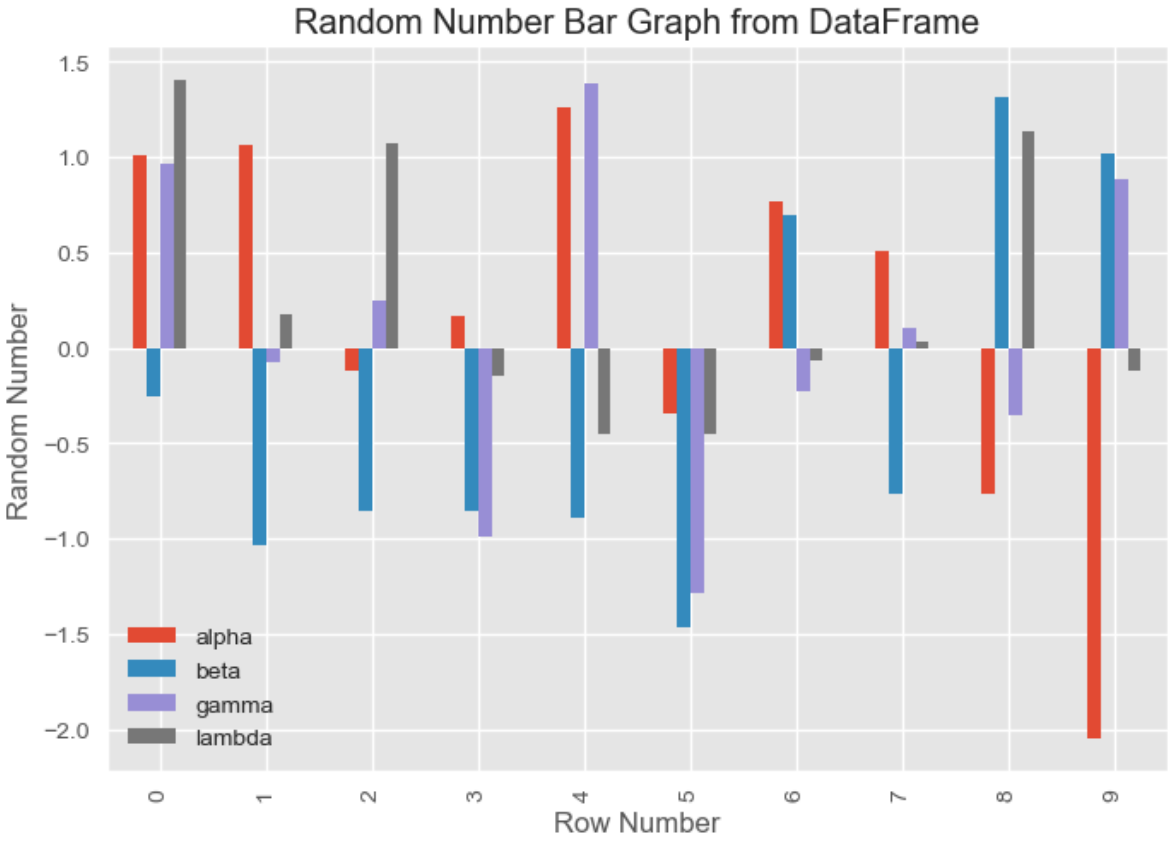
| | alpha | beta | gamma | lambda |
|---|---|---|---|---|
| **0** | 1.013714 | -0.259431 | 0.968316 | 1.407097 |
| **1** | 1.060656 | -1.031388 | -0.071920 | 0.178539 |
| **2** | -0.121181 | -0.854002 | 0.246935 | 1.076682 |
| **3** | 0.168935 | -0.851748 | -0.986361 | -0.143816 |
| **4** | 1.261360 | -0.891466 | 1.387766 | -0.449085 |
| **5** | -0.341835 | -1.470084 | -1.285446 | -0.448073 |
| **6** | 0.771401 | 0.700177 | -0.227552 | -0.065134 |
| **7** | 0.503949 | -0.762883 | 0.100736 | 0.027910 |
| **8** | -0.762372 | 1.316568 | -0.352562 | 1.137105 |
| **9** | -2.046849 | 1.015623 | 0.882883 | -0.121415 |

In [78]:
```
Sample_df.plot(kind='bar')
```

Out[78]:
```
<AxesSubplot: >
```



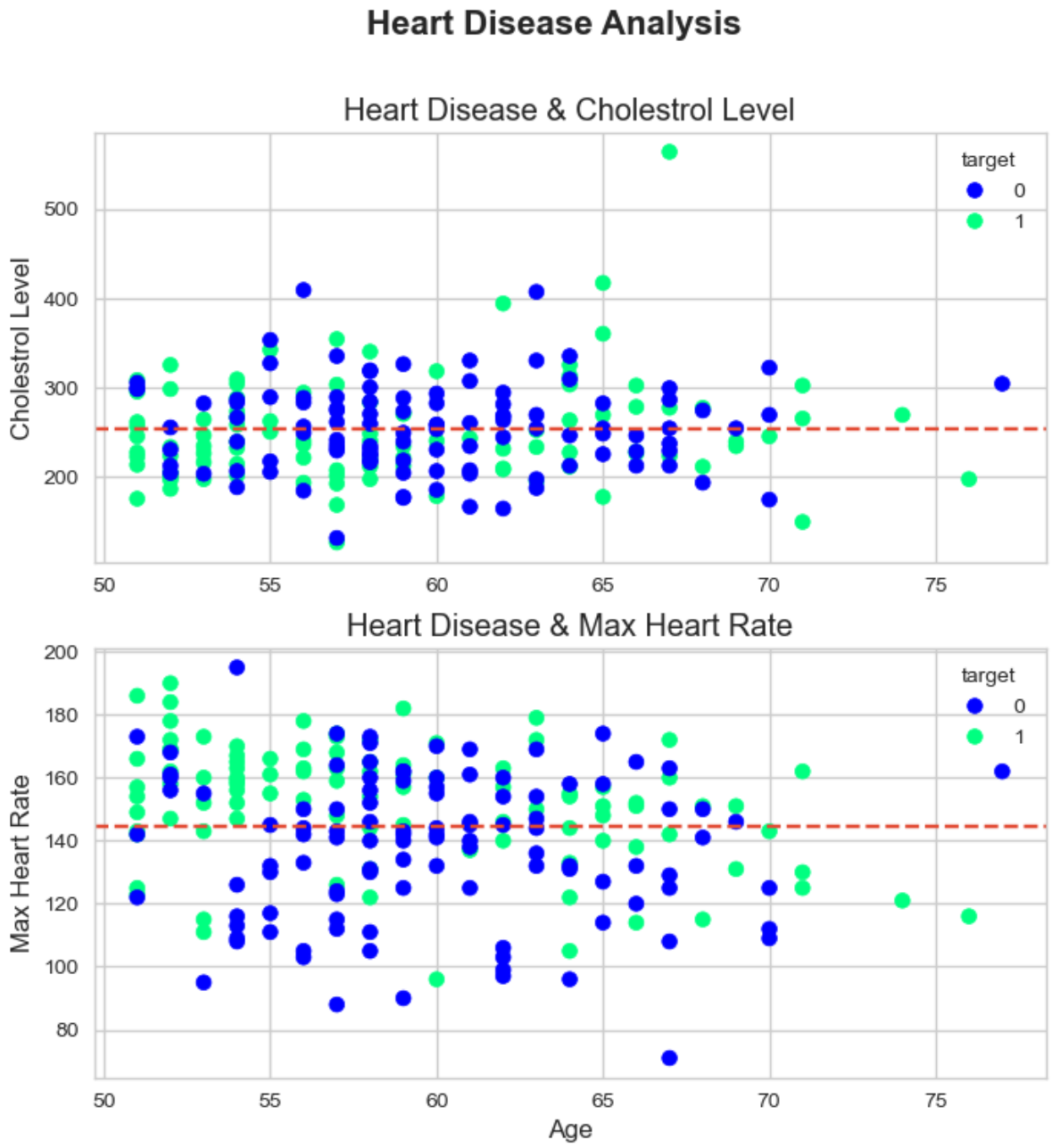

In [81]:
```
# to customize the plot with set()
ax = Sample_df.plot(kind='bar')

#customize plot and add label
ax.set(title='Random Number Bar Graph from DataFrame',xlabel='Row Number',ylabel='
```



In [82]:
```
# change style within style
```

In [95]:
```
plt.style.use('seaborn-whitegrid')


fig, (ax1,ax2) = plt.subplots(nrows=2,ncols=1,figsize=(8,8))
scatter_1 = ax1.scatter(x=over_50['age'],y=over_50['chol'],c=over_50['target'],cma
scatter_2 = ax2.scatter(x=over_50['age'],y=over_50['thalach'],c=over_50['target'],
ax1.set(title='Heart Disease & Cholestrol Level',ylabel='Cholestrol Level')
ax1.set_xlim([50,80])
ax2.set(title='Heart Disease & Max Heart Rate',xlabel='Age',ylabel='Max Heart Rate
ax2.set_xlim([50,80])
ax2.set_ylim([60,200])

#show legend on plot so that it will be easily understandable
ax1.legend(*scatter_1.legend_elements(),title='target')
ax2.legend(*scatter_2.legend_elements(),title='target')

#create a horizontal line which represent the mean of cholestrol and max heart rate
ax1.axhline(over_50['chol'].mean(),linestyle='--')
ax2.axhline(over_50['thalach'].mean(),linestyle='--')

#create a title of figure
fig.suptitle('Heart Disease Analysis', fontsize='16',fontweight='bold')
plt.show()
```

```
C:\Users\Hanu\AppData\Local\Temp\ipykernel_6560\294226472.py:1: MatplotlibDeprecat
ionWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as
they no longer correspond to the styles shipped by seaborn. However, they will rem
ain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn A
PI instead.
  plt.style.use('seaborn-whitegrid')
```




In [ ]:
```
# we can download any plot by using this
#fig.savefig('Heart-Disease-with-code.png')
```