Data visualization allows us to quickly interpret the data and adjust different variables to see their effect

•Technology is increasingly making it easier for us to do so

Why visualize data?

o Observe the patterns

o Identify extreme values that could be anomalies

o Easy interpretation Popular plotting libraries in Python Python offers multiple graphing libraries that offers diverse features

• 1) matplotlib --> to create 2D graphs and plots •

2) pandas visualization --> easy to use interface, built on Matplotlib •

3) seaborn --> provides a high level interface for drawing attractive and informative statistical graphics •

4) ggplot --> based on R’s ggplot2, uses Grammar of Graphics •

5) plotly --> can create interactive plots

Scatter Plot What is a scatter plot? A scatter plot is a set of points that represents the values obtained for two different variables plotted on a horizontal and vertical axes

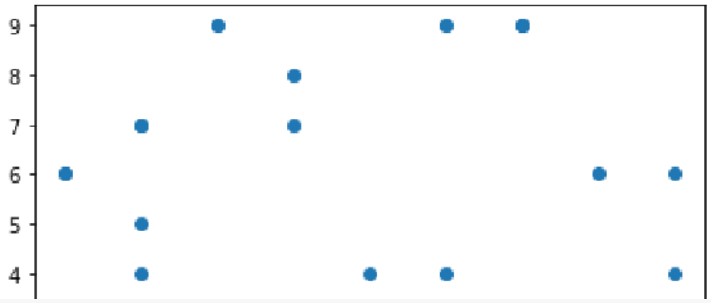
When to use scatter plots?

Scatter plots are used to convey the relationship between two numerical variables

Scatter plots are sometimes called correlation plots because they show how two variables are correlated

```
import matplotlib.pyplot as plt
# create a figure and axis
fig, ax = plt.subplots()
```

```
<matplotlib.collections.PathCollection at 0x7fb57e1d3e10>
```

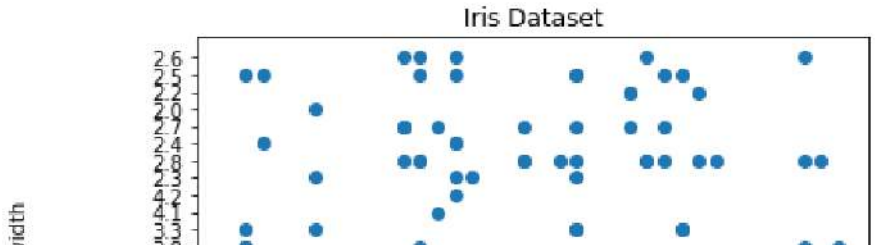



```
import pandas as pd
iris = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Iris.csv', names=['sepal_length',
print(iris.head())
```

```
     sepal_length   sepal_width   petal_length   petal_width        class
Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
1             5.1           3.5            1.4           0.2  Iris-setosa
2             4.9           3.0            1.4           0.2  Iris-setosa
3             4.7           3.2            1.3           0.2  Iris-setosa
4             4.6           3.1            1.5           0.2  Iris-setosa
```

```
import matplotlib.pyplot as plt
# create a figure and axis
fig, ax = plt.subplots()

# scatter the sepal_length against the sepal_width
ax.scatter(iris['sepal_length'], iris['sepal_width'])
# set a title and labels
ax.set_title('Iris Dataset')
ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
```

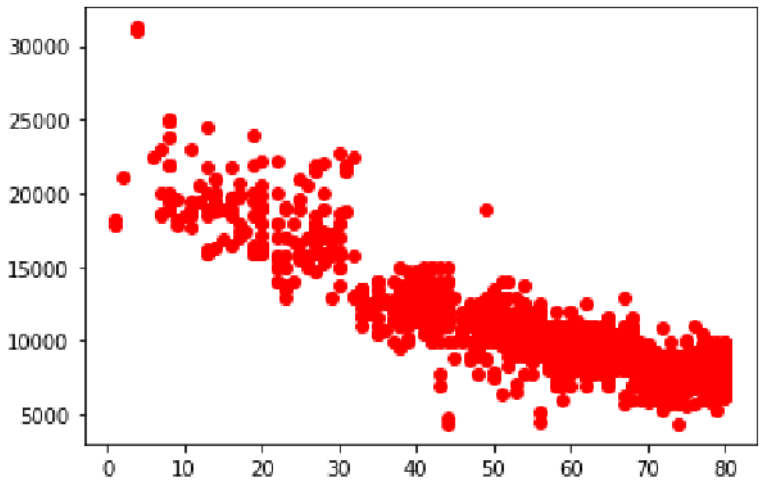
```
Text(0, 0.5, 'sepal_width')
```



```
cars_data.head()
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170

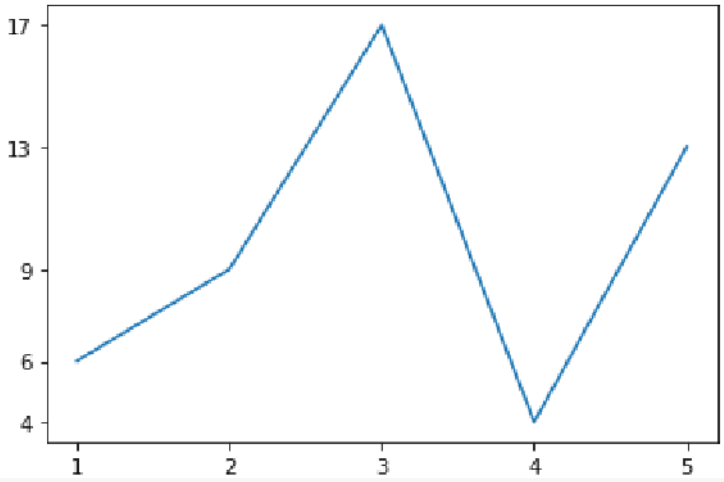
```
import matplotlib.pyplot as plt
plt.scatter(cars_data['Age'],cars_data['Price'], c='red')
plt.show()
```



Line Chart

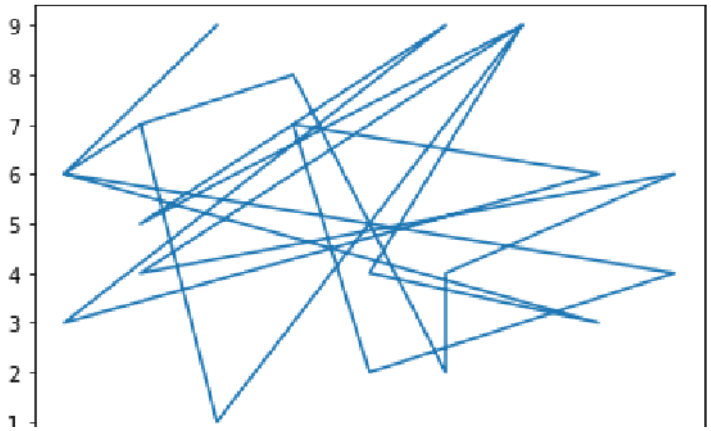
In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want and plotting each column on the same axis.

```
([<matplotlib.axis.YTick at 0x7fb576e52f90>,
  <matplotlib.axis.YTick at 0x7fb576e52850>,
  <matplotlib.axis.YTick at 0x7fb576e99ad0>,
  <matplotlib.axis.YTick at 0x7fb576e78b10>,
  <matplotlib.axis.YTick at 0x7fb576e78d90>],
 <a list of 5 Text major ticklabel objects>)
```



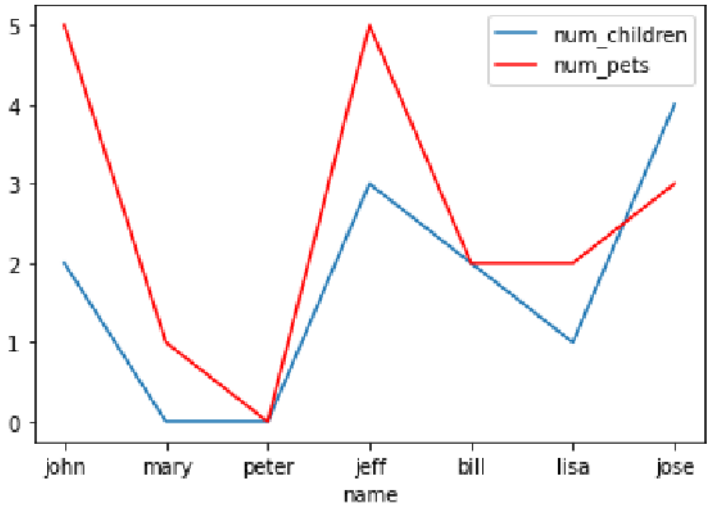

```
import matplotlib.pyplot as plt
# create a figure and axis
fig, ax = plt.subplots()

x = [2, 4, 6, 6, 9, 2, 7, 2, 6, 1, 8, 4, 5, 9, 1, 2, 3, 7, 5, 8, 1, 3]
y = [7, 8, 2, 4, 6, 4, 9, 5, 9, 3, 6, 7, 2, 4, 6, 7, 1, 9, 4, 3, 6, 9]
ax.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7fb5755ca110>]
```



```
})
# From pandas to plot multiple plots on same figure
# gca stands for 'get current axis'
ax = plt.gca()

df.plot(kind='line',x='name',y='num_children',ax=ax)
df.plot(kind='line',x='name',y='num_pets', color='red',ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5755f8f90>
```

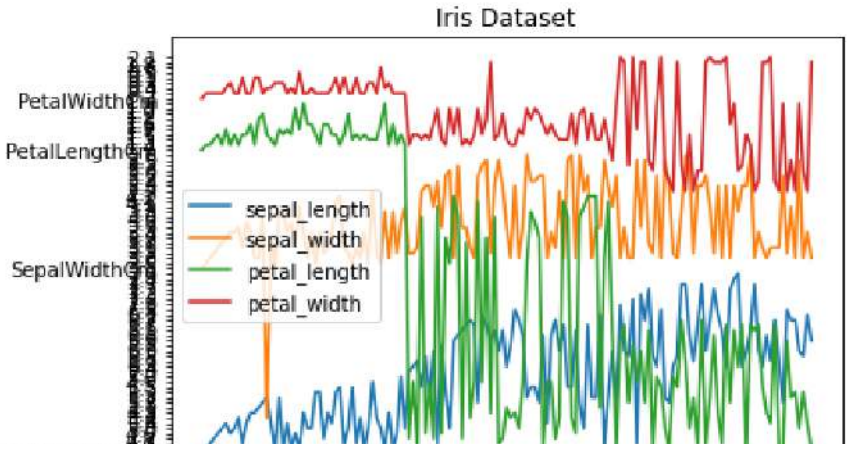



```
import pandas as pd
iris = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Iris.csv', names=['sepal_length',
print(iris.head())
```

```
     sepal_length   sepal_width   petal_length   petal_width        class
Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
1             5.1           3.5            1.4           0.2  Iris-setosa
2             4.9           3.0            1.4           0.2  Iris-setosa
3             4.7           3.2            1.3           0.2  Iris-setosa
4             4.6           3.1            1.5           0.2  Iris-setosa
```

```
# get columns to plot
columns = iris.columns.drop(['class'])
# create x data
x_data = range(0, iris.shape[0])
```

```
<matplotlib.legend.Legend at 0x7fb575538590>
```

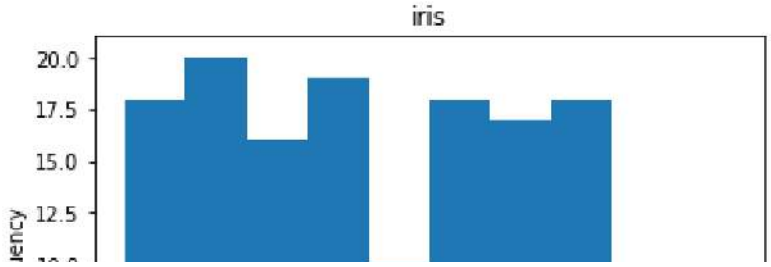


Histogram

In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.

```
# create figure and axis
fig, ax = plt.subplots()
# plot histogram
ax.hist(iris['sepal_length'])
# set title and labels
ax.set_title('iris')
ax.set_xlabel('sepal_length')
ax.set_ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```



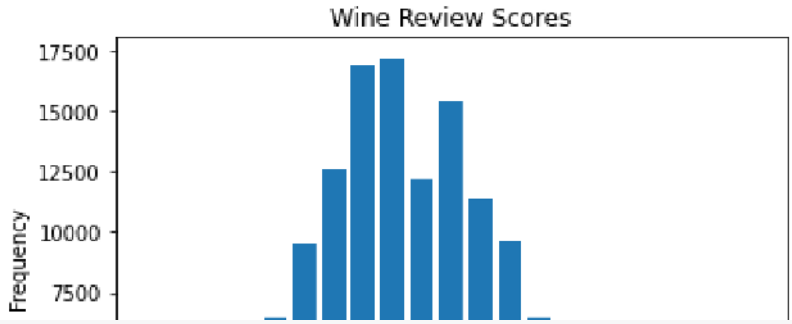
A bar chart can be created using the bar method. The bar-chart isn't automatically calculating the frequency of a category so we are going to use pandas value_counts function to do this. The bar-chart is useful for categorical data that doesn't have a lot of different categories (less than 30) because else it can get quite messy.

```
wine_reviews = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/winemag-data-130k-v2.csv',
wine_reviews.head()
```

	country	description	designation	points	price	province	region_1	region_2	tas
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Keri
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	R
2	US	Tart and snappy, the flavors of lime flesh	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Pa

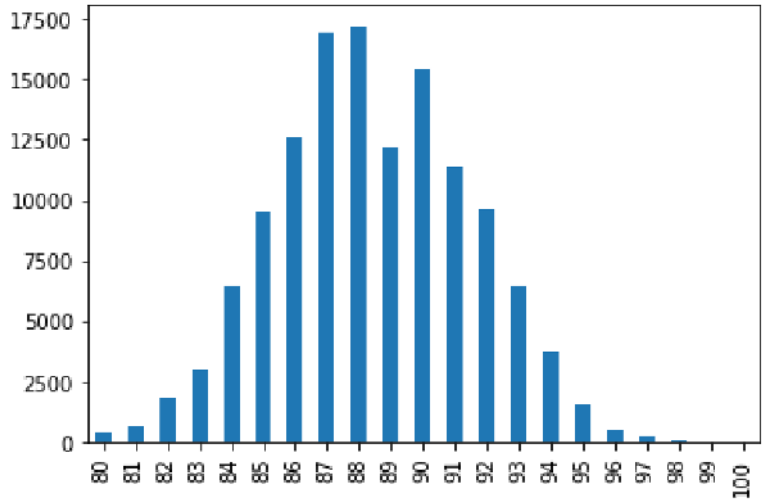
```
#Bar Chart
# create a figure and axis
fig, ax = plt.subplots()
# count the occurrence of each class
data = wine_reviews['points'].value_counts()
# get x and y data
points = data.index
frequency = data.values
# create bar chart
ax.bar(points, frequency)
```

```
Text(0, 0.5, 'Frequency')
```



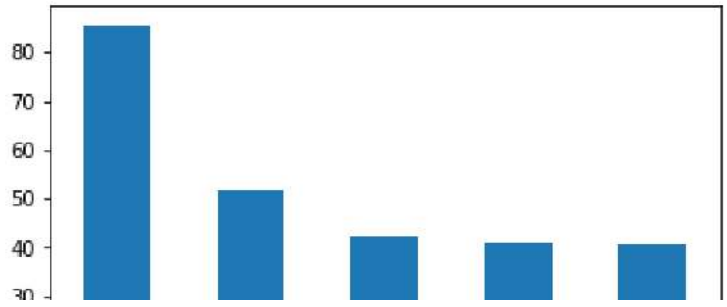

```
wine_reviews['points'].value_counts().sort_index().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f629781eb10>
```



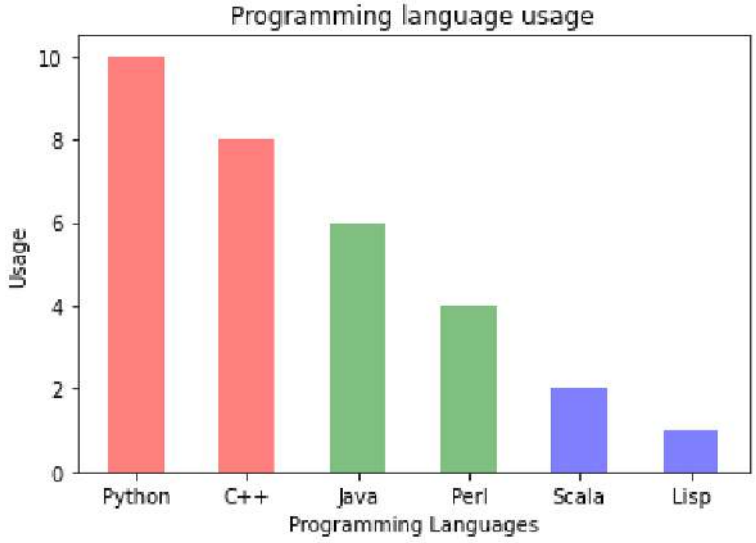

```
wine_reviews.groupby("country").price.mean().sort_values(ascending=False)[:5].plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f629781e290>
```



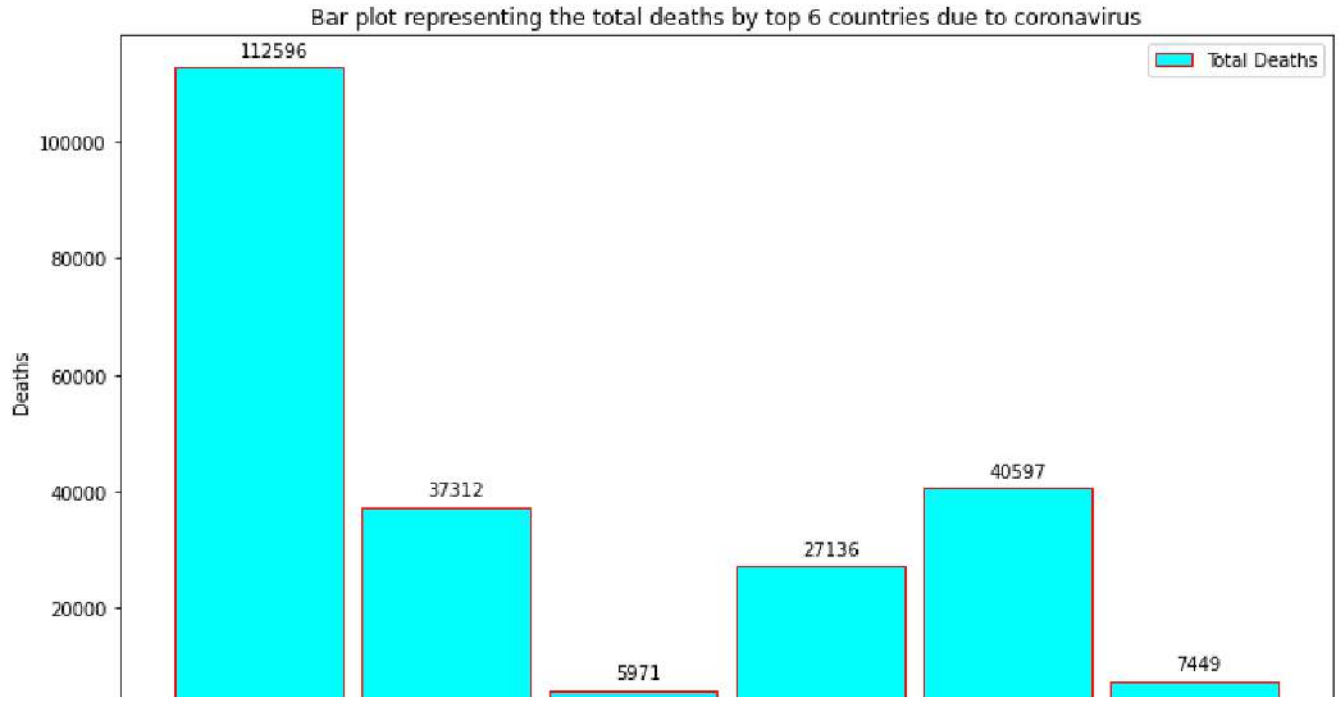
```
objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]
# Bar Chart
# X Axis positions as first parameter list, it can be floating point numbers also
# Y Values as 2nd parameter list
# Alpha is transparency,
# Align can be center or edge
# Color can be single value or a list of color codes, one for each bar.
plt.bar(y_pos, performance, width=0.5, align='center', alpha=0.5, color=['r', 'r', 'g', 'g',
# To define labels for x axis values.
plt.xticks(y_pos, objects)
plt.ylabel('Usage')
plt.xlabel('Programming Languages')
plt.title('Programming language usage')
```

```
Text(0.5, 1.0, 'Programming language usage')
```




```
# Importing the matplotlib library
import matplotlib.pyplot as plt

# Declaring the figure or the plot (y, x) or (width, height)
plt.figure(figsize = (12,7))
```

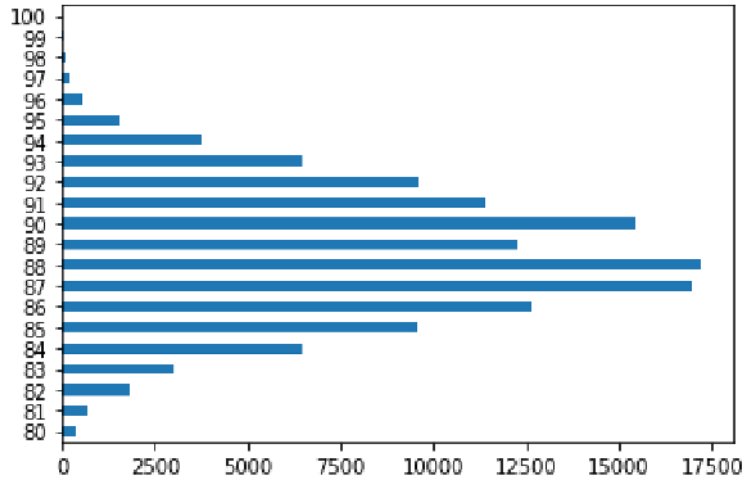
```
# Annotating the bar plot with the values (total death count)
for i in range(len(countries)):
    plt.annotate(totalDeaths[i], (-0.1 + i, totalDeaths[i] + j))

# Creating the legend of the bars in the plot
plt.legend(labels = ['Total Deaths'])

# Giving the tilte for the plot
plt.title("Bar plot representing the total deaths by top 6 countries due to coronavirus")

# Namimg the x and y axis
plt.xlabel('Countries')
plt.ylabel('Deaths')

# Saving the plot as a 'png'
plt.savefig('1BarPlot.png')

# Displaying the bar plot
plt.show()
```



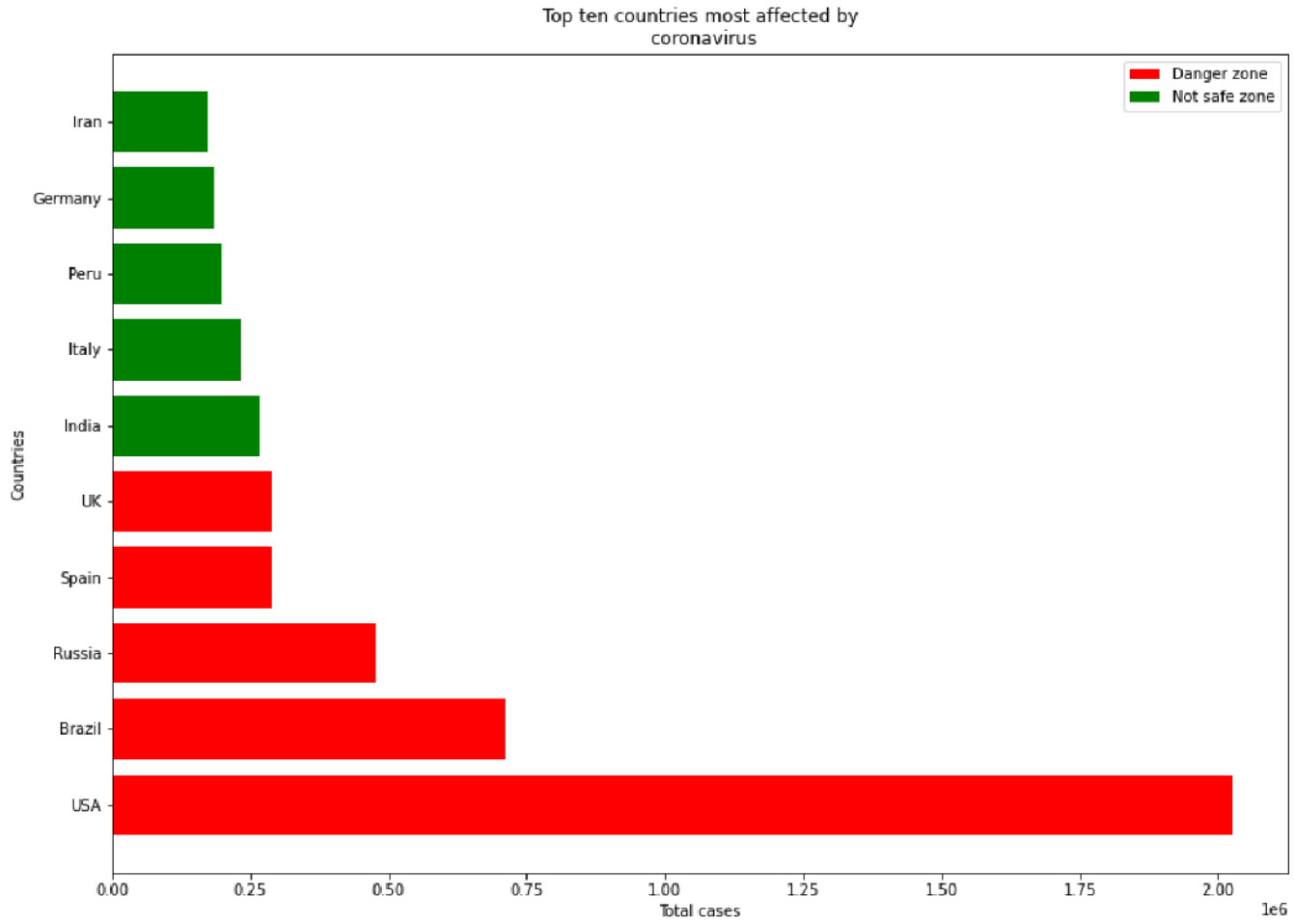
It's also really simple to make a horizontal bar-chart using the plot.barh() method. By adding one extra character 'h', we can align the bars horizontally. Also, we can represent the bars in two or more different colors, this will increase the readability of the plots

```
wine_reviews['points'].value_counts().sort_index().plot.barh()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f62975f6d50>
```




```
# Importing the matplotlib library
import matplotlib.pyplot as plt

# Declaring the figure or the plot (y, x) or (width, height)
plt.figure(figsize=[14, 10])

# Passing the parameters to the bar function, this is the main function which creates the bar
# For creating the horizontal make sure that you append 'h' to the bar function name
plt.barh(['USA', 'Brazil', 'Russia', 'Spain', 'UK'], [2026493, 710887, 476658, 288797, 287399
plt.barh(['India', 'Italy', 'Peru', 'Germany', 'Iran'], [265928, 235278, 199696, 186205, 1738

# Creating the legend of the bars in the plot
plt.legend()

# Namimg the x and y axis
plt.xlabel('Total cases')
plt.ylabel('Countries')
```

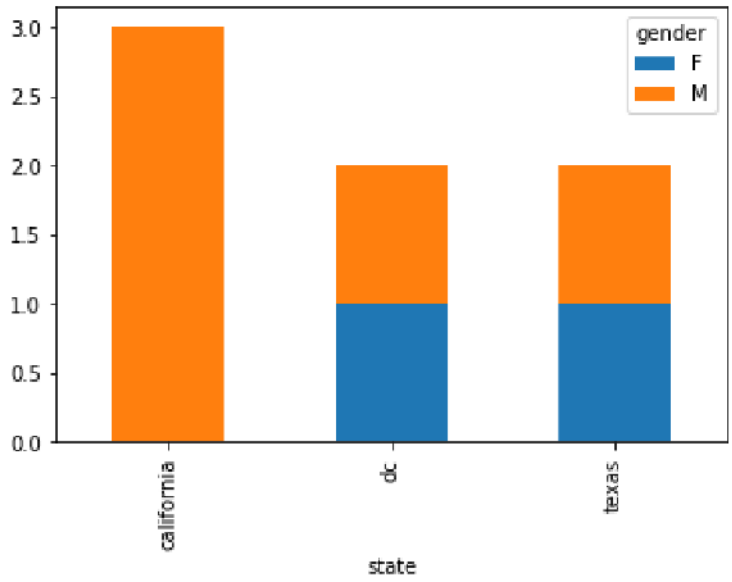


Stacking two bar plots on top of each other

At times you might want to stack two or more bar plots on top of each other. With the help of this, you can differentiate two separate quantities visually. To do this just follow.

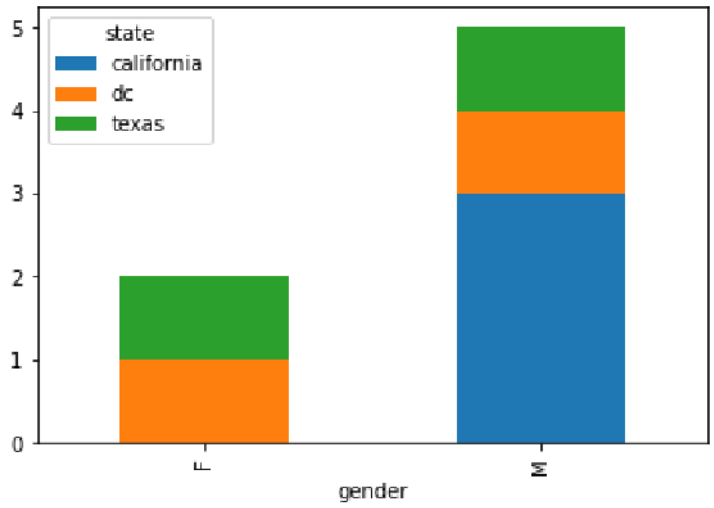
```
import pandas as pd
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f62974a7a50>
```



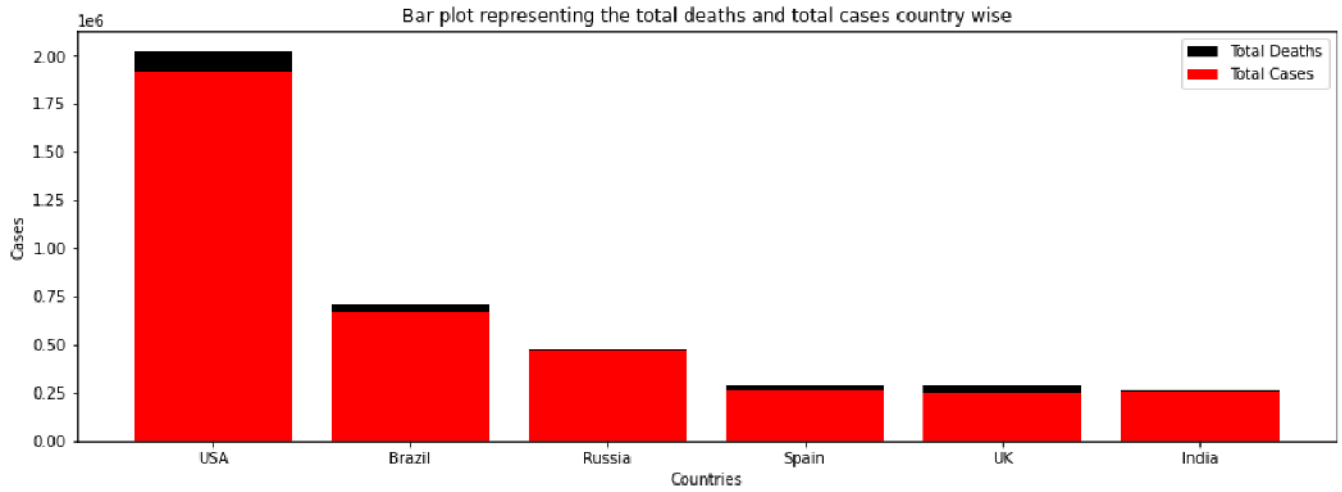

```
df.groupby(['gender','state']).size().unstack().plot(kind='bar',stacked=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f629748ed10>
```



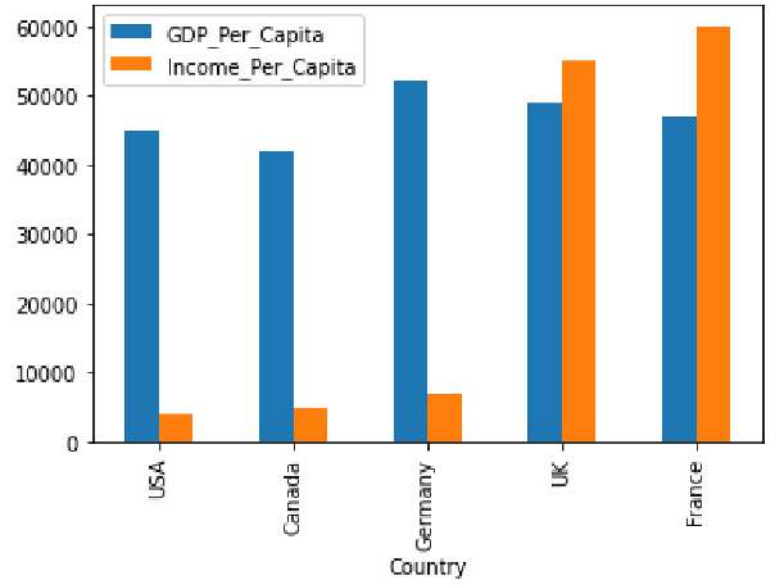

```
# Importing the matplotlib library
import matplotlib.pyplot as plt
```

```
# Plotting both the total death and the total cases in a single plot. Formula total cases - t
for i in range(len(countries)):
    plt.bar(countries[i], totalDeaths[i], bottom = totalCases[i] -  totalDeaths[i], color='bl
    plt.bar(countries[i], totalCases[i] - totalDeaths[i], color='red')

# Creating the legend of the bars in the plot
plt.legend(labels = ['Total Deaths','Total Cases'])

# Giving the tilte for the plot
plt.title("Bar plot representing the total deaths and total cases country wise")

# Namimg the x and y axis
plt.xlabel('Countries')
plt.ylabel('Cases')

# Saving the plot as a 'png'
plt.savefig('3BarPlot.png')

# Displaying the bar plot
plt.show()
```



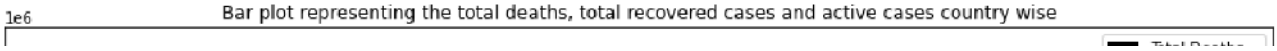
```
Data = {'Country': ['USA','Canada','Germany','UK','France'],
        'GDP_Per_Capita': [45000,42000,52000,49000,47000],
        'Income_Per_Capita': [4000,5000,7000,55000,60000]
       }


df = pd.DataFrame(Data)
# Multiple metrics in same chart
df.plot(x ='Country', y=['GDP_Per_Capita', 'Income_Per_Capita'], kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f62973d7550>
```



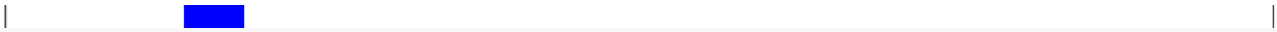

```
# Importing the matplotlib library
import numpy as np
import matplotlib.pyplot as plt

# Declaring the figure or the plot (y, x) or (width, height)
plt.figure(figsize=[15, 10])

# Data to be plotted
totalDeath = [113055, 37312, 5971, 7473, 33964]
```

```
plt.bar(X + 0.5, activeCases, color = 'b', width = 0.25)

# Creating the legend of the bars in the plot
plt.legend(['Total Deaths', 'Total Recovery', 'Active Cases'])

# Overiding the x axis with the country names
plt.xticks([i + 0.25 for i in range(5)], country)

# Giving the tilte for the plot
plt.title("Bar plot representing the total deaths, total recovered cases and active cases cou

# Namimg the x and y axis
plt.xlabel('Countries')
plt.ylabel('Cases')

# Saving the plot as a 'png'
plt.savefig('4BarPlot.png')

# Displaying the bar plot
plt.show()
```



Pie chart

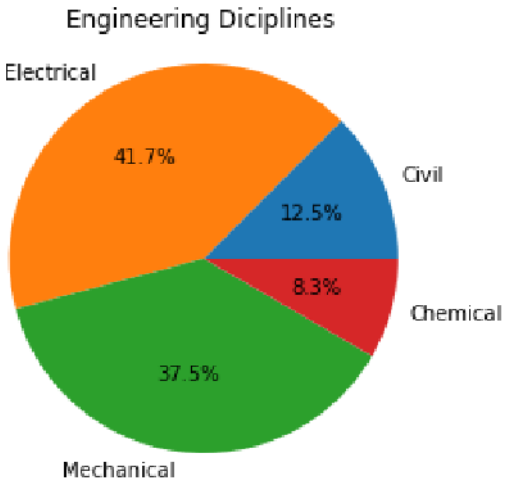
A pie chart is a type of data visualization that is used to illustrate numerical proportions in data.



```
# Data Frame plotting
from pandas import DataFrame
import matplotlib.pyplot as plt

Data = {'Tasks': [300,500,700],
       'Task Type' : ['Tasks Pending','Tasks Ongoing','Tasks Completed']
       }

df = DataFrame(Data)
df.set_index('Task Type', inplace=True)

# autopct has extra % at the end as escape, as % is interpreted as formatting string begin by
# Only pie chart needs labels to be data frame index
df.plot.pie(y='Tasks', figsize=(10,10),autopct='%1.1f%%', startangle=90)
```

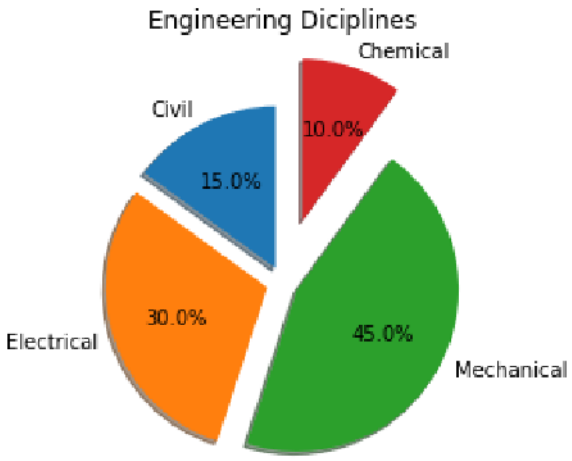
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6297541150>
```


```
import numpy as np
import matplotlib.pyplot as plt
# if using a Jupyter notebook, include:
%matplotlib inline


# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = ['Civil', 'Electrical', 'Mechanical', 'Chemical']
sizes = [15, 50, 45, 10]

fig, ax = plt.subplots()
ax.pie(sizes, labels=labels, autopct='%1.1f%%')
ax.axis('equal')  # Equal aspect ratio ensures the pie chart is circular.
ax.set_title('Engineering Diciplines')

plt.show()
```



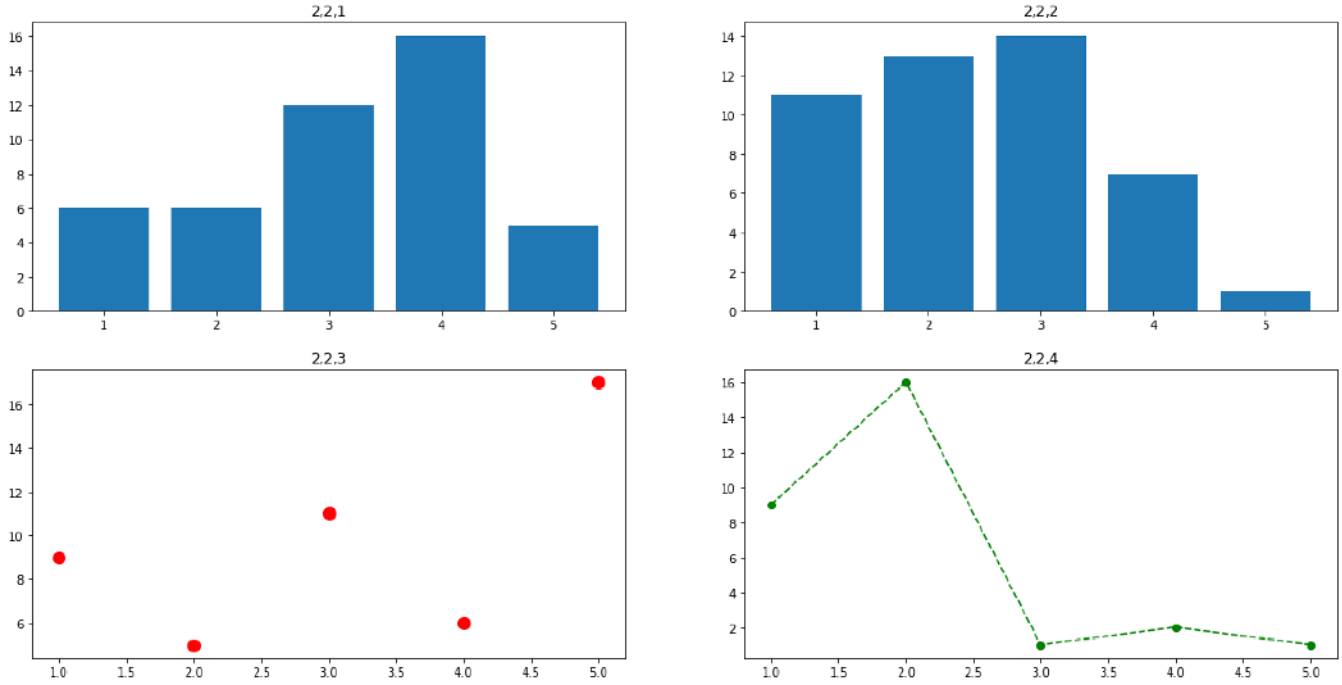

```
import numpy as np
import matplotlib.pyplot as plt
# if using a Jupyter notebook, include:
%matplotlib inline
```

```
        labels=labels,
        autopct='%1.1f%%',
        shadow=True,
        startangle=90)
ax.axis('equal')  # Equal aspect ratio ensures the pie chart is circular.
ax.set_title('Engineering Diciplines')

plt.show()
```



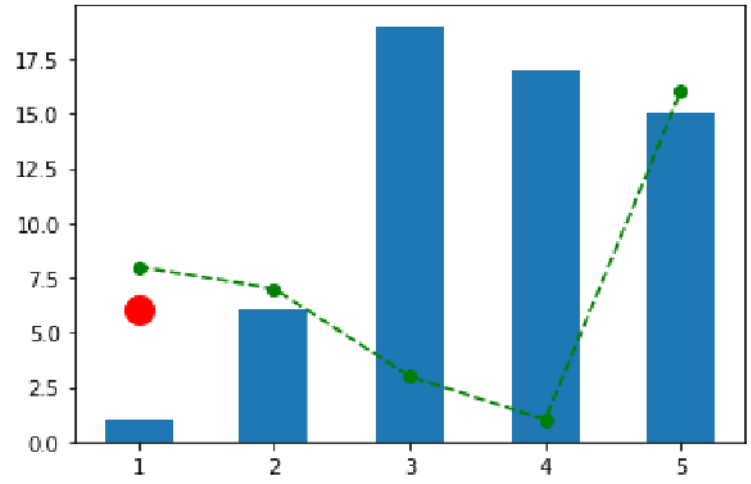

```
plt.figure(figsize=(20,10))
plt.subplot(2,2,1)
plt.bar(range(1,6), np.random.randint(1,20,5))
plt.title("2,2,1")
plt.subplot(2,2,2)
plt.bar(range(1,6), np.random.randint(1,20,5))
plt.title("2,2,2")
plt.subplot(2,2,3)
# s is the size of dot
plt.scatter(range(1,6), np.random.randint(1,20,5), s=100, color="r")
plt.title("2,2,3")
plt.subplot(2,2,4)
plt.plot(range(1,6), np.random.randint(1,20,5), marker='o', color='g', linestyle='--')
plt.title("2,2,4")
```

```
Text(0.5, 1.0, '2,2,4')
```




```
plt.bar(range(1,6), np.random.randint(1,20,5), width=0.5)
plt.scatter(range(1,6), np.random.randint(1,20,5), s=200, color="r")
plt.plot(range(1,6), np.random.randint(1,20,5), marker='o', color='g', linestyle='--')
```

```
[<matplotlib.lines.Line2D at 0x7f6296fa1310>]
```



```
import seaborn as sns
import os
```

```
os.chdir('/content/drive/MyDrive/Colab Notebooks')
cars_data=pd.read_csv('Toyota.csv',index_col=0,na_values=["??","????"])
cars_data.size
```

```
14360
```

```
cars_data.dropna(axis=0,inplace=True)
cars_data.size
```

```
10960
```

```
cars_data=pd.read_csv('Toyota.csv')
cars_data.head()
```

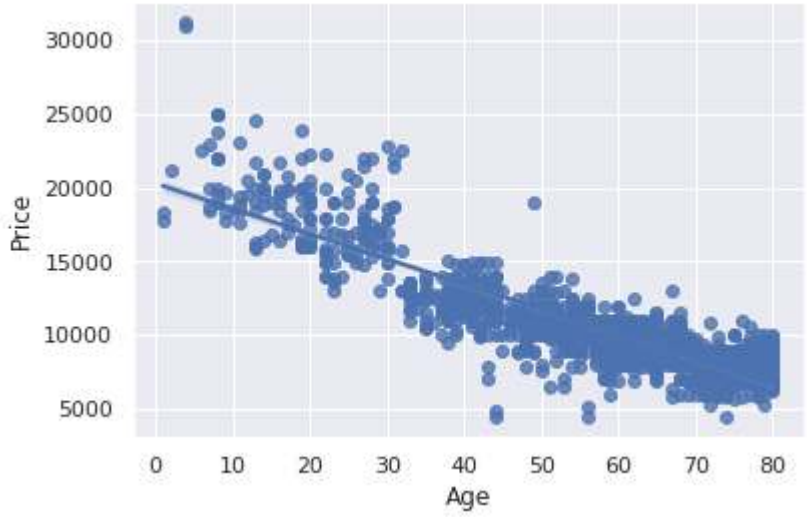
	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165

```
cars_data=pd.read_csv('Toyota.csv',index_col=0)
cars_data.head()
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165

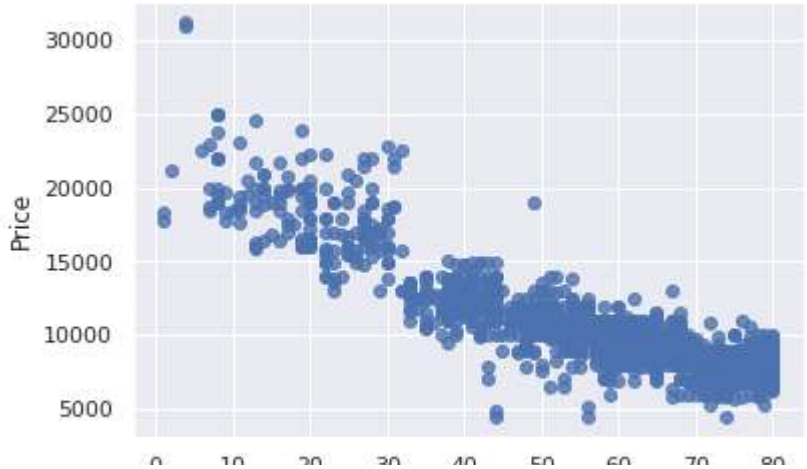
```
sns.set(style="darkgrid")
sns.regplot(x=cars_data['Age'],y=cars_data['Price'])

#It estimates and plots a regression model relating the x and y variables
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f628ae52cd0>
```

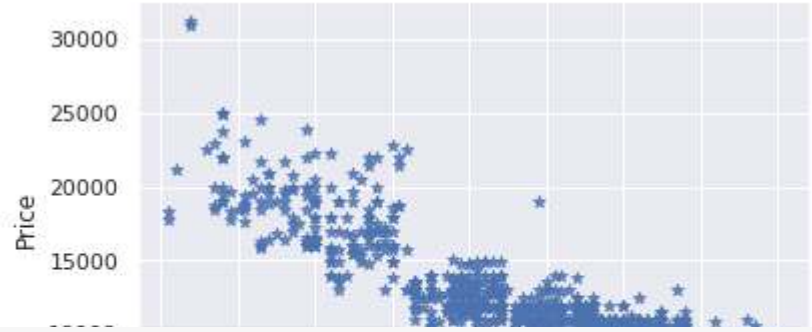



```
#Scatter plot of Price vs Age without the regression fit line
sns.regplot(x=cars_data['Age'],y=cars_data['Price'],fit_reg=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f628ad38190>
```

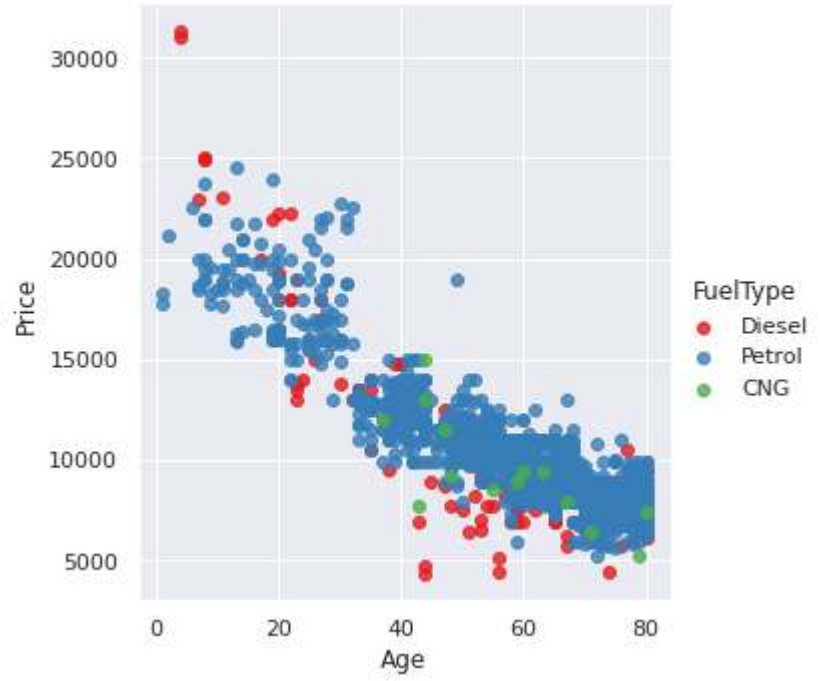


```
<matplotlib.axes._subplots.AxesSubplot at 0x7f628ad22310>
```



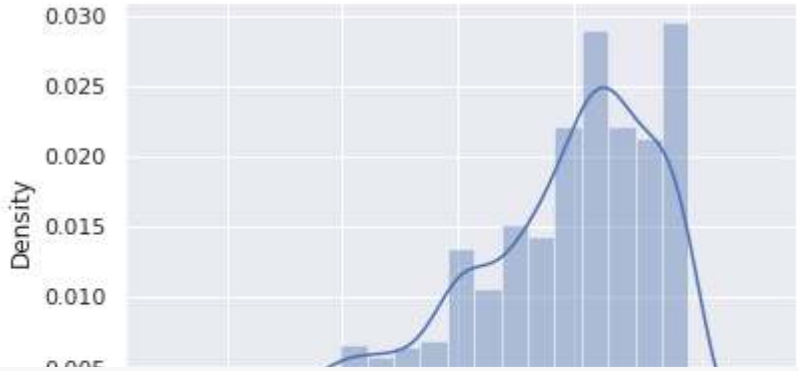

```
# Scatter plot of Price vs Age by FuelType

#Using hue parameter, including another variable to show the fuel types categories with diffe

sns.lmplot(x='Age', y='Price', data=cars_data, fit_reg=False, hue='FuelType', legend=True, pa
```

```
<seaborn.axisgrid.FacetGrid at 0x7f628ac8d210>
```



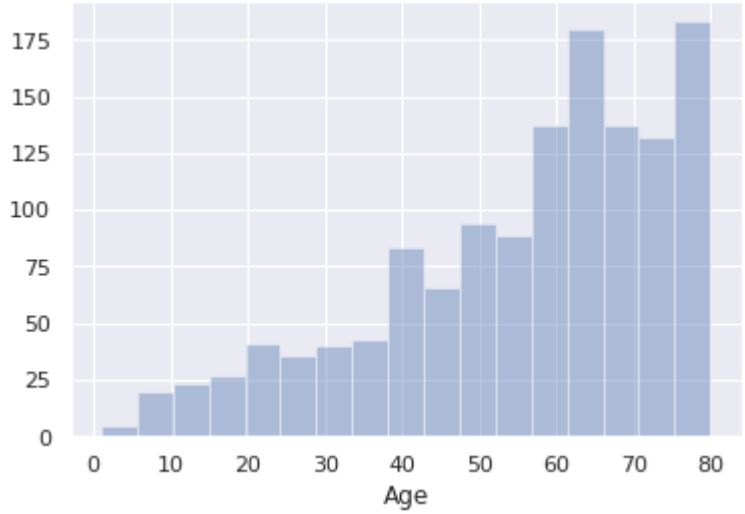
Histogram

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f62883e5a10>
```



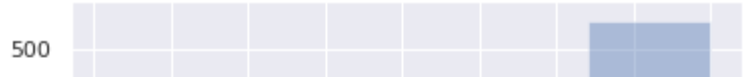

```
#Histogram without kernel density estimate
sns.distplot(cars_data['Age'],kde=False)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f62882ffd10>
```




```
#Histogram with fixed no. of bins
sns.distplot(cars_data['Age'],kde=False, bins=5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f62882a6c50>
```

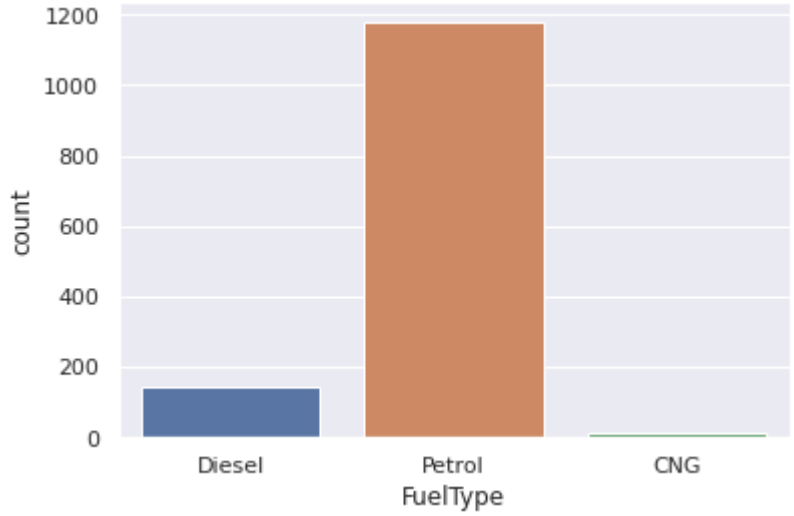


Bar plot

Frequency distribution of fuel type of the cars

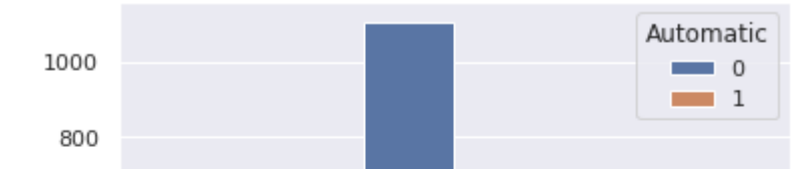
```
sns.countplot(x="FuelType", data=cars_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6288222150>
```




```
###Grouped bar plot
#Grouped bar plot of FuelType and Automatic

sns.countplot(x="FuelType", data=cars_data, hue="Automatic")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f62881968d0>
```



FuelType	CNG	Diesel	Petrol
Automatic			
0	15	144	1104
1	0	0	73

Box and whiskers plot

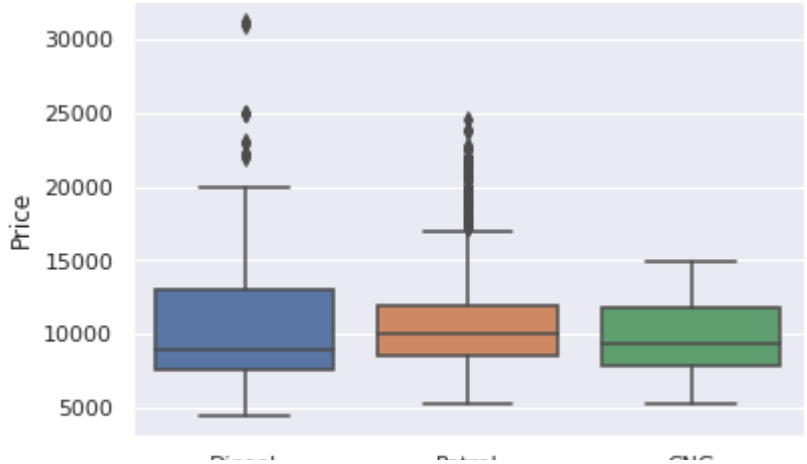
Box and whiskers plot for numerical vs categorical variable

A Box Plot is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum. In the box plot, a box is created from the first quartile to the third quartile, a vertical line is also there which goes through the box at the median. Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution.

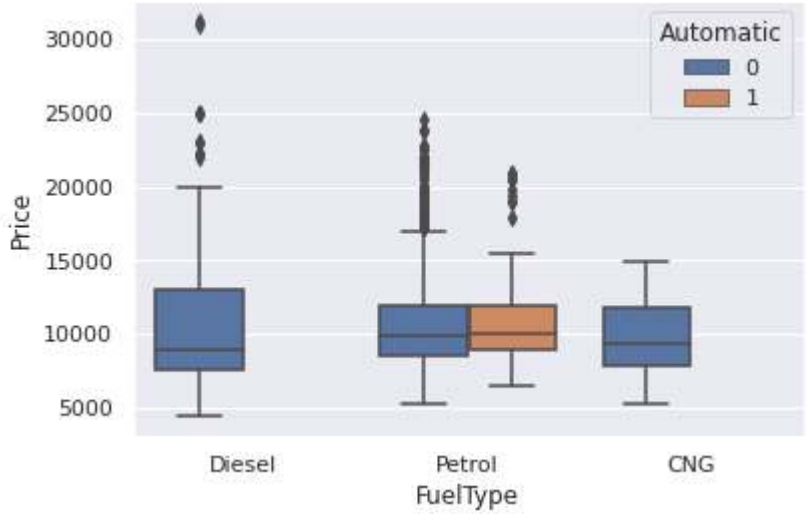
Price of the cars for various fuel types

```
sns.boxplot(x=cars_data['FuelType'],y=cars_data["Price"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6288105bd0>
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f62880a7690>
```



Box

whiskers plot and Histogram

Let's plot box whiskers plot and histogram on the same window

Split the plotting window into 2 parts

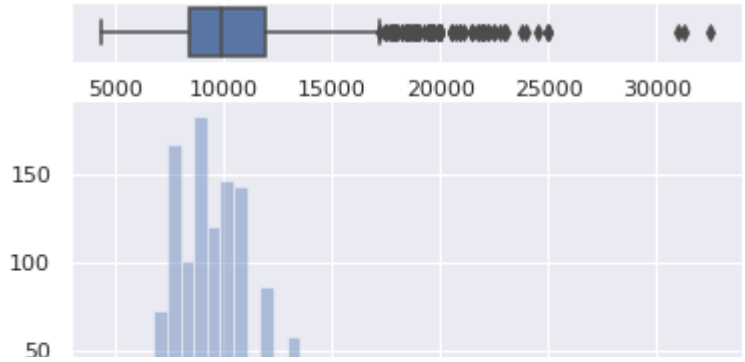
```
f,(ax_box,ax_hist)=plt.subplots(2,gridspec_kw={"height_ratios": (.15, .85)})
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-515edc54e648> in <module>
----> 1 f,(ax_box,ax_hist)=plt.subplots(2,gridspec_kw={"height_ratios": (.15, .85)})

NameError: name 'plt' is not defined
```

SEARCH STACK OVERFLOW

Now, add create two plots

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f6287f71050>
```



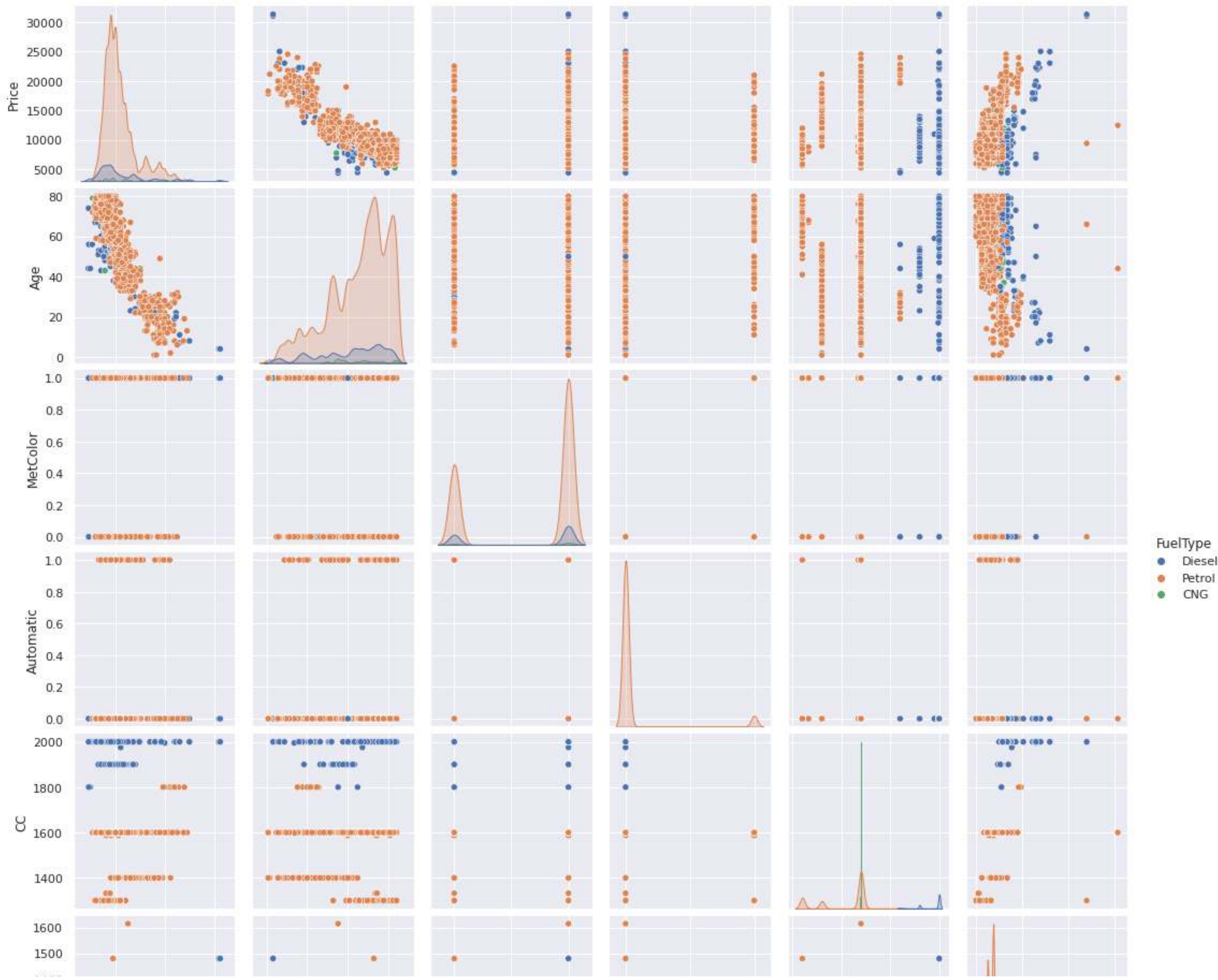
Pairwise plots

It is used to plot pairwise relationships in a dataset

Creates scatterplots for joint relationships and histograms for univariate distributions

```
sns.pairplot(cars_data,kind="scatter",hue="FuelType",diag_kws={'bw': 0.1})
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The
  warnings.warn(msg, FutureWarning)
```

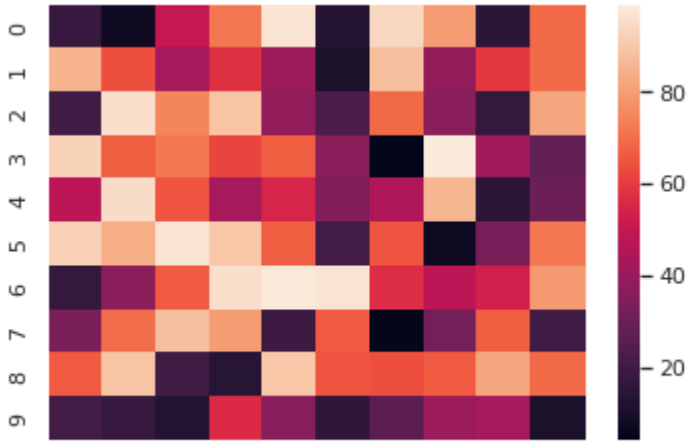


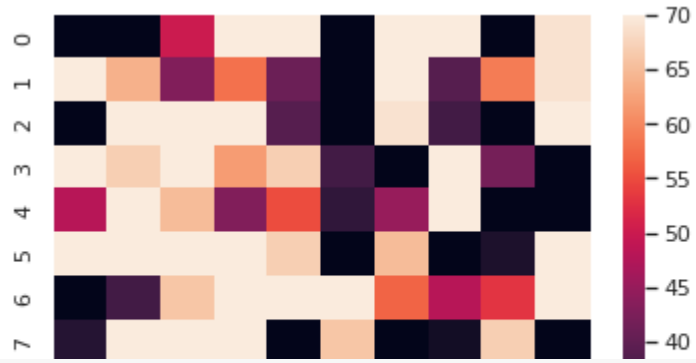
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
```

```
data=np.random.randint(1,100,size=(10,10))
print("The data to be plotted: \n")
print(data)
```

```
The data to be plotted:

[[18  8 50 72 97 13 94 80 15 69]
 [85 64 43 58 41 11 88 39 59 69]
 [20 96 75 89 39 23 69 37 17 82]
 [93 67 72 62 67 37  4 99 42 28]
 [48 95 65 43 55 35 45 86 15 30]
 [92 84 97 90 67 21 65  7 33 72]
 [17 37 66 96 98 97 57 48 53 79]
 [34 70 88 80 19 66  5 32 67 20]
 [66 89 20 14 90 65 64 66 82 69]
 [21 18 13 56 36 16 27 41 43 11]]
```

```
#Plotting Heatmap
hm=sns.heatmap(data=data)
plt.show()
```






```
# setting the parameter values
cmap = "tab20"
center = 0

# setting the parameter values
annot = True

# plotting the heatmap
hm = sns.heatmap(data=data, cmap=cmap, annot=annot)



# displaying the plotted heatmap
plt.show()
```