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

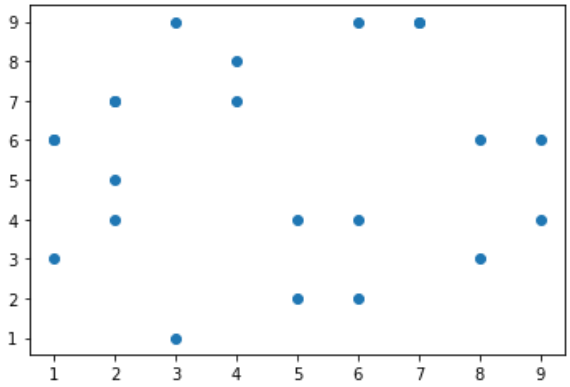
Scatter plots are sometimes called correlation plots because they show how two variables are correlated

In [1]:

```
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.scatter(x, y)
```

Out[1]:

<matplotlib.collections.PathCollection at 0x1108ae5c608>



In [2]:

```
import pandas as pd
iris = pd.read_csv('D:/AnitaRJ/DATA SCIENCE/MScI_DataSci_Practicals/Practical7/iris.csv', names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])
print(iris.head())
```

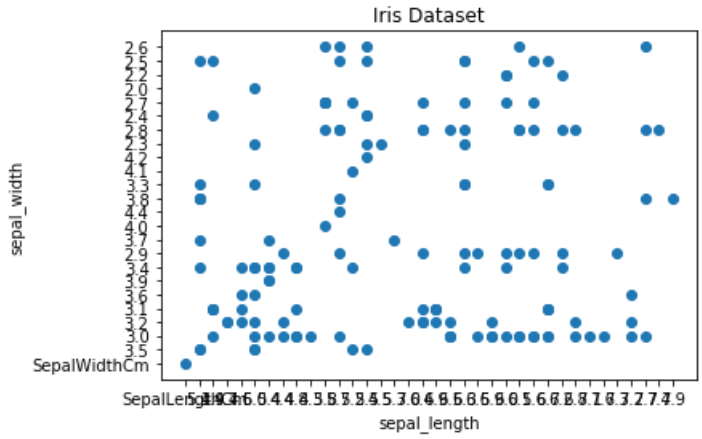
	sepal_length	sepal_width	petal_length	petal_width	class
Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa

In [3]:

```
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')
```

Out[3]:

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



In [8]:

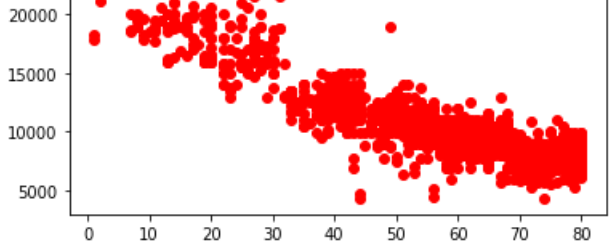
```
import pandas as pd
cars_data=pd.read_csv('D:/AnitaRJ/DATA SCIENCE/MScI_DataSci_Practicals/Practical7/Toyota.csv',index_col=0)
cars_data.head()
```

Out[8]:

	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

In [10]:

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



Line Chart

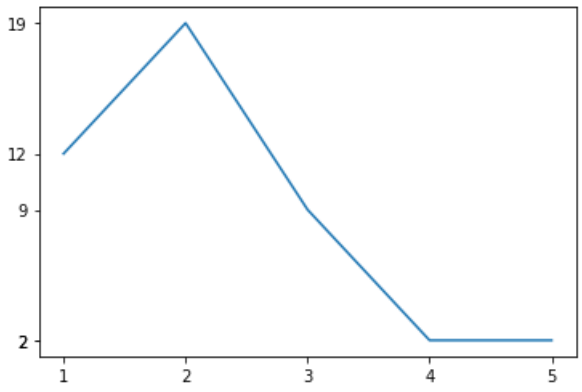
In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want and plotting each column on the same axis.

In [8]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
x=range(1,6)
y=np.random.randint(1,20,5)
plt.plot(x,y)

plt.xticks(x)
plt.yticks(y)
```

Out[8]:

```
([<matplotlib.axis.YTick at 0x12989b8e448>,
  <matplotlib.axis.YTick at 0x129898e4f88>,
  <matplotlib.axis.YTick at 0x12989b88408>,
  <matplotlib.axis.YTick at 0x12989bb92c8>,
  <matplotlib.axis.YTick at 0x12989bb99c8>],
 <a list of 5 Text yticklabel objects>)
```

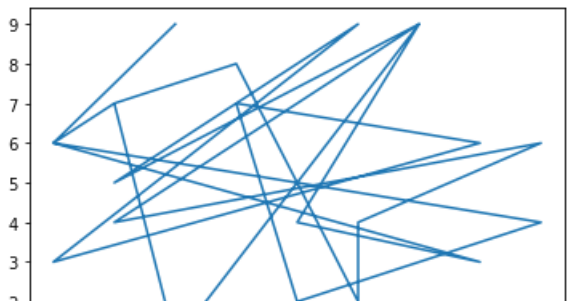


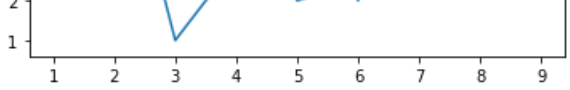
In [7]:

```
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)
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x12989b27088>]
```



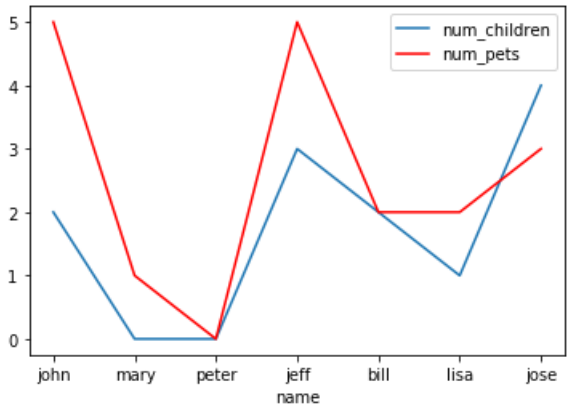


In [35]:

```
import pandas as pd
df = pd.DataFrame({
    'name':['john','mary','peter','jeff','bill','lisa','jose'],
    'age':[23,78,22,19,45,33,20],
    'gender':['M','F','M','M','M','F','M'],
    'state':['california','dc','california','dc','california','texas','texas'],
    'num_children':[2,0,0,3,2,1,4],
    'num_pets':[5,1,0,5,2,2,3]
})
# 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)
```

Out[35]:

<matplotlib.axes._subplots.AxesSubplot at 0x11090c193c8>



In [9]:

```
import pandas as pd
iris = pd.read_csv('D:/AnitaRJ/DATA SCIENCE/MScI_DataSci_Practicals/Practical7/iris.csv', names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])
print(iris.head())
```

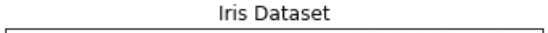
	sepal_length	sepal_width	petal_length	petal_width	class
Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa

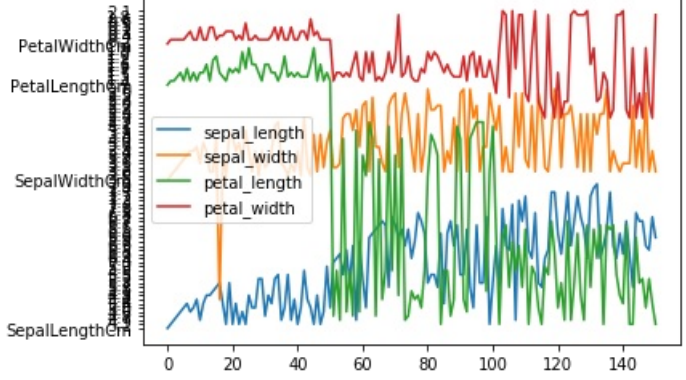
In [5]:

```
# get columns to plot
columns = iris.columns.drop(['class'])
# create x data
x_data = range(0, iris.shape[0])
# create figure and axis
fig, ax = plt.subplots()
# plot each column
for column in columns:
    ax.plot(x_data, iris[column], label=column)
# set title and legend
ax.set_title('Iris Dataset')
ax.legend()
```

Out[5]:

<matplotlib.legend.Legend at 0x1108aaf9888>





Histogram

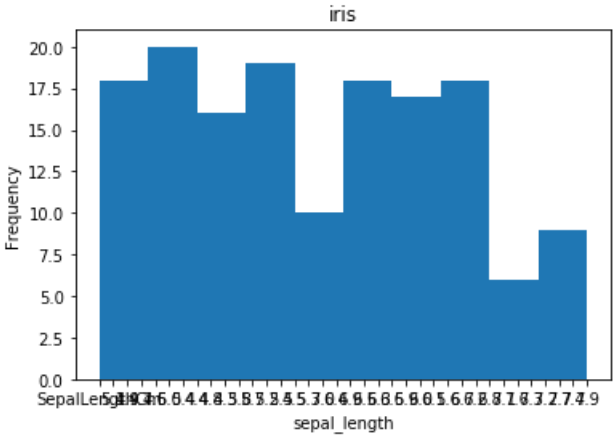
In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.

In [6]:

```
# 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')
```

Out[6]:

Text(0, 0.5, 'Frequency')



Bar Chart

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.

In [8]:

```
wine_reviews = pd.read_csv('D:/AnitaRJ/DATA SCIENCE/MScI_DataSci_Practicals/Practical7/winemag-data-130k-v2.csv', index_col=0)
wine_reviews.head()
```

Out[8]:

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	wine
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicos
		This is ripe									Quinta dos Avidagos		Quir

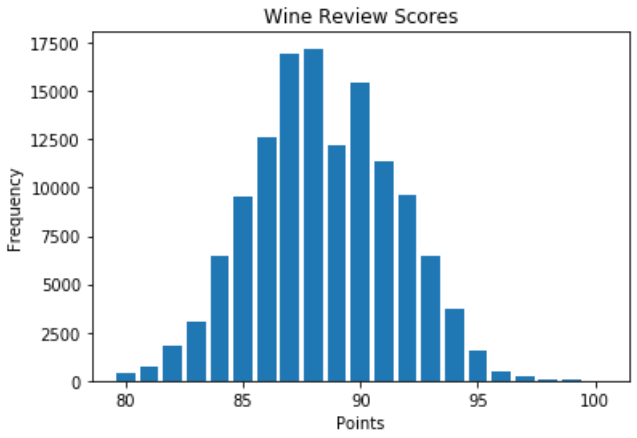
	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	wine
1	Portugal	and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	2011 Avidagos Red (Douro)	Portuguese Red	Avidag
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainsto
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Juli
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Swe Chee

In [9]:

```
#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)
# set title and labels
ax.set_title('Wine Review Scores')
ax.set_xlabel('Points')
ax.set_ylabel('Frequency')
```

Out[9]:

Text(0, 0.5, 'Frequency')

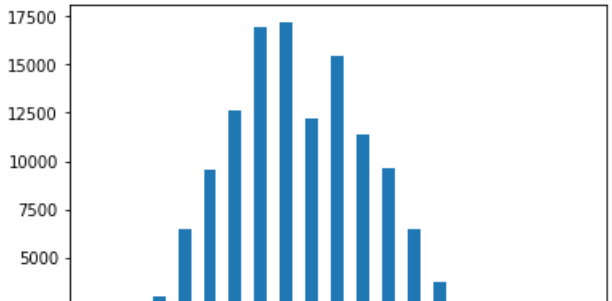


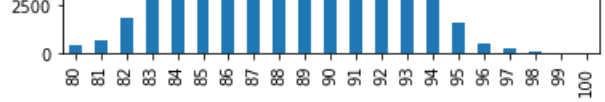
In [10]:

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

Out[10]:

<matplotlib.axes._subplots.AxesSubplot at 0x1108ea89088>





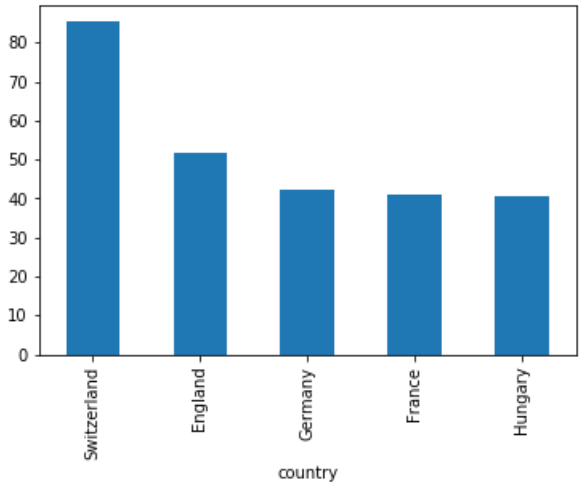
We can also plot other data then the number of occurrences.

In [12]:

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

Out[12]:

<matplotlib.axes._subplots.AxesSubplot at 0x1108eb548c8>



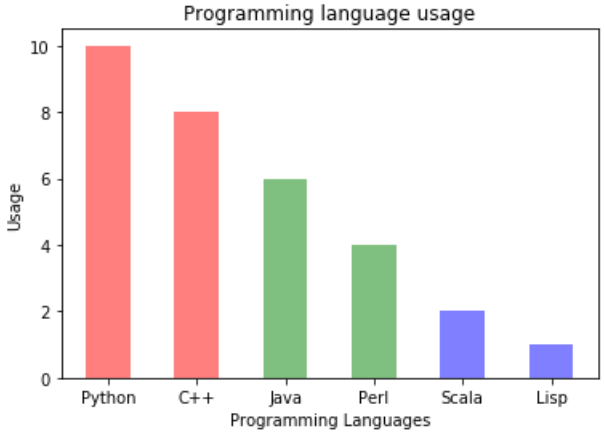
Adding more charecteristics to bar graph

In [32]:

```
import numpy as np
import matplotlib.pyplot as plt
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', 'b', 'b'])
# To define labels for x axis values.
plt.xticks(y_pos, objects)
plt.ylabel('Usage')
plt.xlabel('Programming Languages')
plt.title('Programming language usage')
```

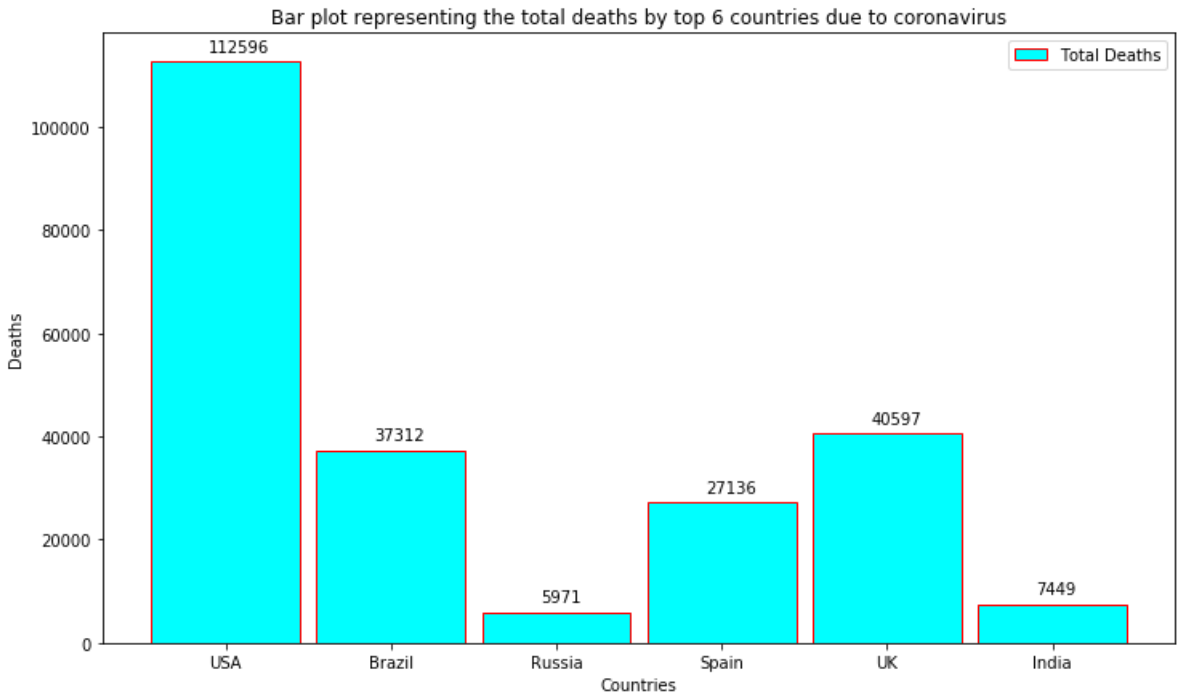
Out[32]:

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



In [26]:

```
# 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))

# Categorical data: Country names
countries = ['USA', 'Brazil', 'Russia', 'Spain', 'UK', 'India']

# Integer value interms of death counts
totalDeaths = [112596, 37312, 5971, 27136, 40597, 7449]

# Passing the parameters to the bar function, this is the main function which creates the bar plot
plt.bar(countries, totalDeaths, width= 0.9, align='center',color='cyan', edgecolor = 'red')

# This is the location for the annotated text
i = 1.0
j = 2000

# 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()
```



Horizontal bar plot

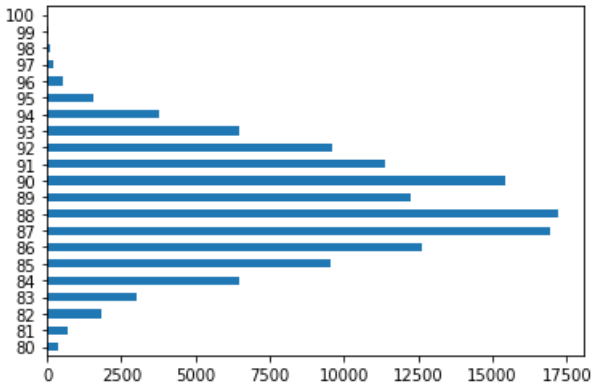
It's also really simple to make a horizontal bar-chart using the plot.barh() method. By adding one extra character 'h', we can align the bars horizontally. Also, we can represent the bars in two or more different colors, this will increase the readability of the plots.

In [11]:

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

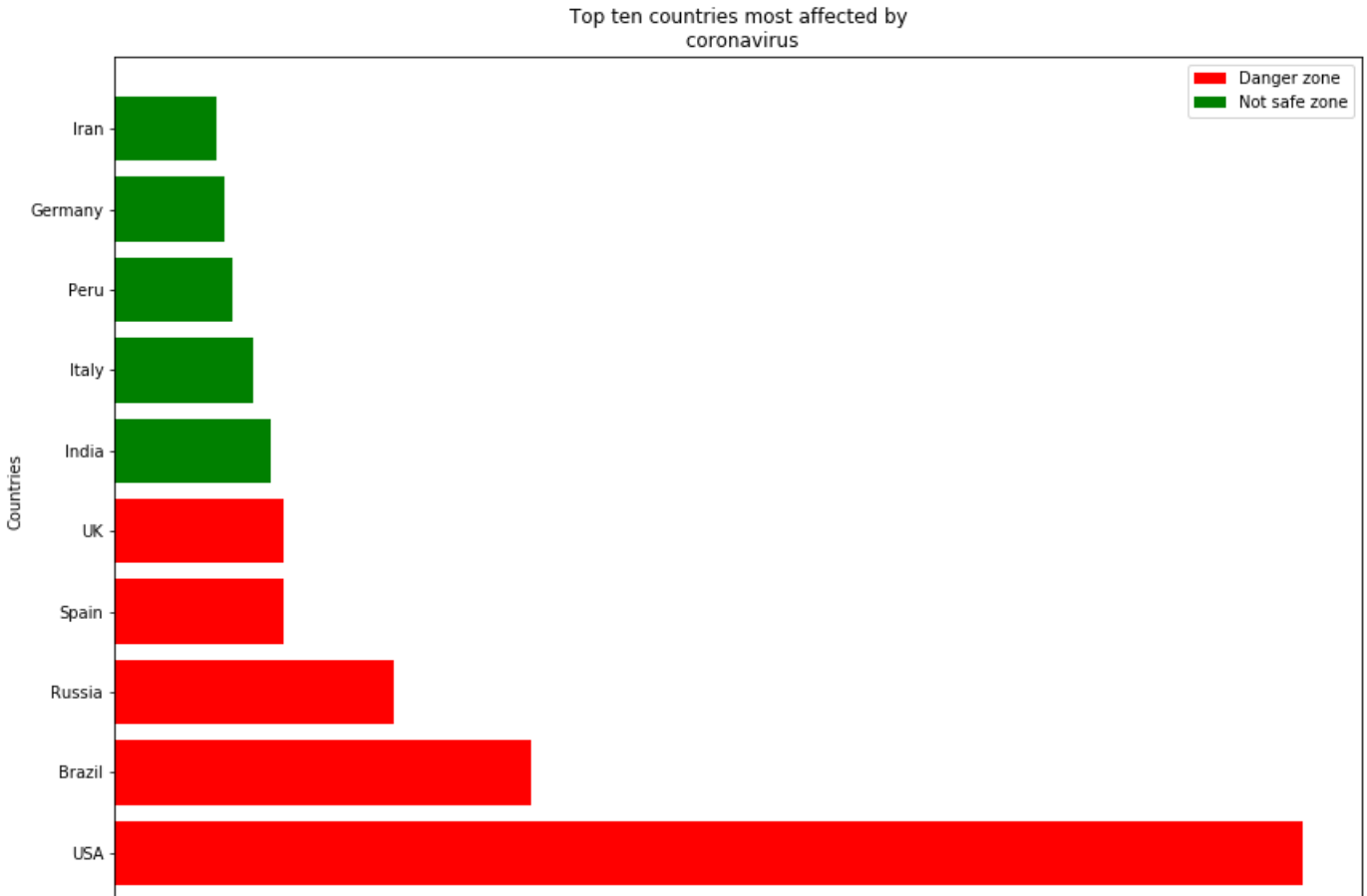
Out[11]:

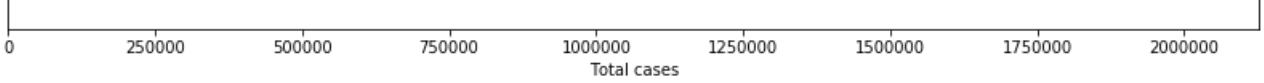
<matplotlib.axes._subplots.AxesSubplot at 0x1108c825248>



In [28]:

```
# 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 plot
# 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], label = "Danger zone", color = 'r')
plt.barh(['India', 'Italy', 'Peru', 'Germany', 'Iran'], [265928, 235278, 199696, 186205, 173832], label = "Not safe zone", color = 'g')

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

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

# Giving the tilte for the plot
plt.title('Top ten countries most affected by\n coronavirus')

# Saving the plot as a 'png'
plt.savefig('2BarPlot.png')

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





Stacking two bar plots on top of each other

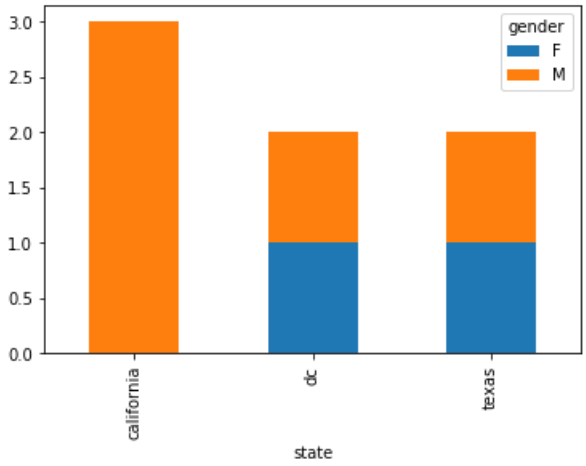
At times you might want to stack two or more bar plots on top of each other. With the help of this, you can differentiate two separate quantities visually. To do this just follow.

In [37]:

```
import pandas as pd
df = pd.DataFrame({
    'name':['john','mary','peter','jeff','bill','lisa','jose'],
    'age':[23,78,22,19,45,33,20],
    'gender':['M','F','M','M','M','F','M'],
    'state':['california','dc','california','dc','california','texas','texas'],
    'num_children':[2,0,0,3,2,1,4],
    'num_pets':[5,1,0,5,2,2,3]
})
# From pandas to plot multiple plots on same figure

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

Out[37]:

<matplotlib.axes._subplots.AxesSubplot at 0x1109171ea88>

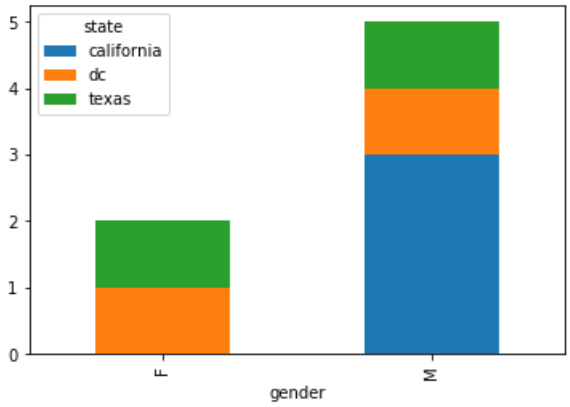


In [38]:

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

Out[38]:

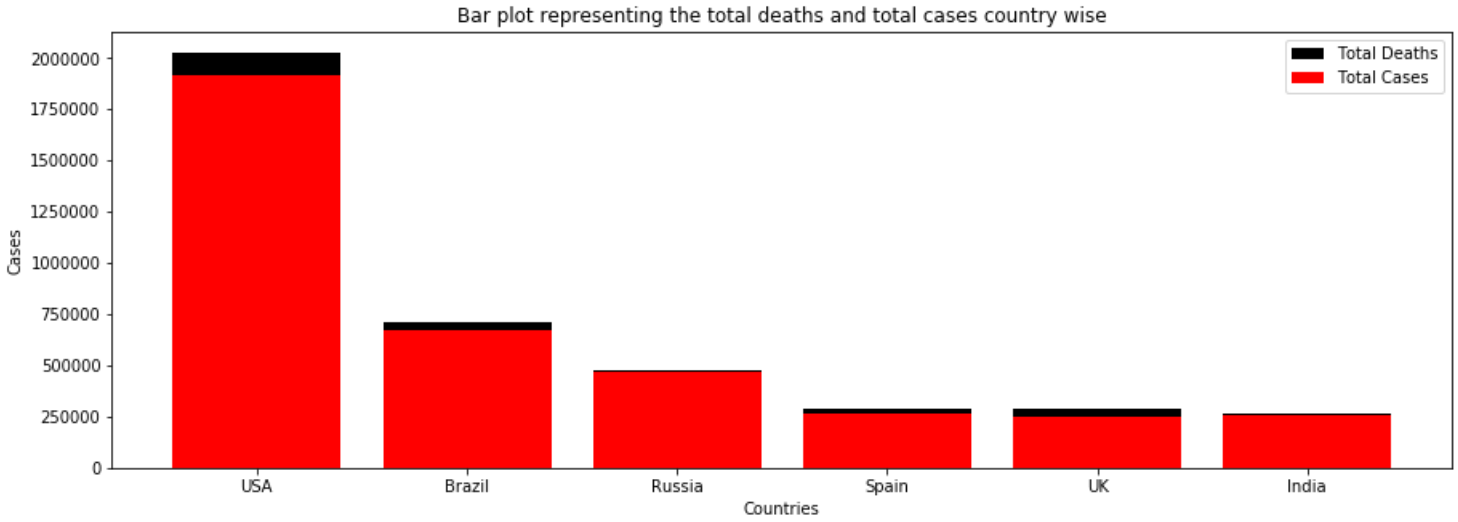
<matplotlib.axes._subplots.AxesSubplot at 0x1108ee54508>



In [30]:

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

# Declaring the figure or the plot (y, x) or (width, height)
```

```
plt.figure(figsize=[15, 5])

# Categorical data: Country names
countries = ['USA', 'Brazil', 'Russia', 'Spain', 'UK', 'India']

# Integer value interms of total cases
totalCases = (2026493, 710887, 476658, 288797, 287399, 265928)

# Integer value interms of death counts
totalDeaths = (113055, 37312, 5971, 27136, 40597, 7473)

# Plotting both the total death and the total cases in a single plot. Formula total cases - total deaths
for i in range(len(countries)):
    plt.bar(countries[i], totalDeaths[i], bottom = totalCases[i] -  totalDeaths[i], color='black')
    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()
```



In [ ]:

```
```

Plotting two or bar plot next to another (Grouping)

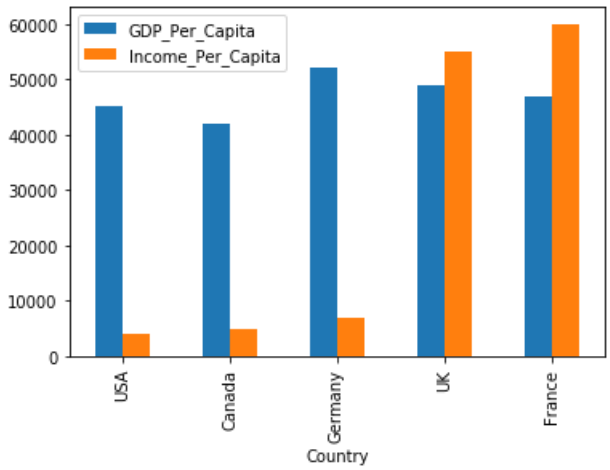
Often many-a-times you might want to group two or more plots just to represent two or more different quantities or whatever. Also in the below code, you can learn to override the name of the x-axis with the name of your choice.

In [34]:

```
import pandas as pd
from matplotlib import pyplot as plt

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')
```

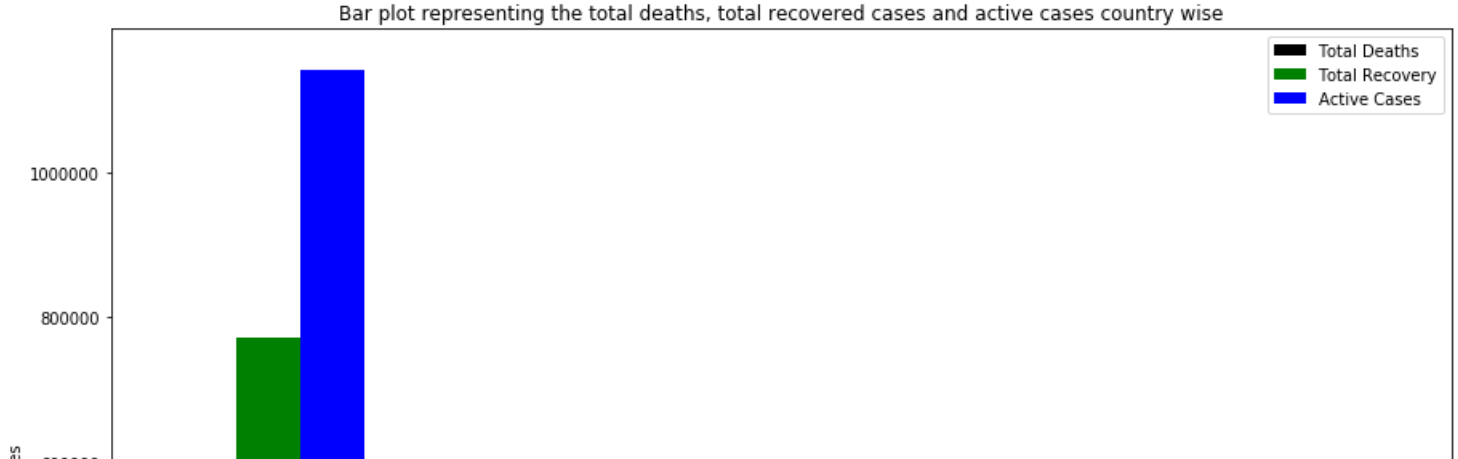
Out[34]:

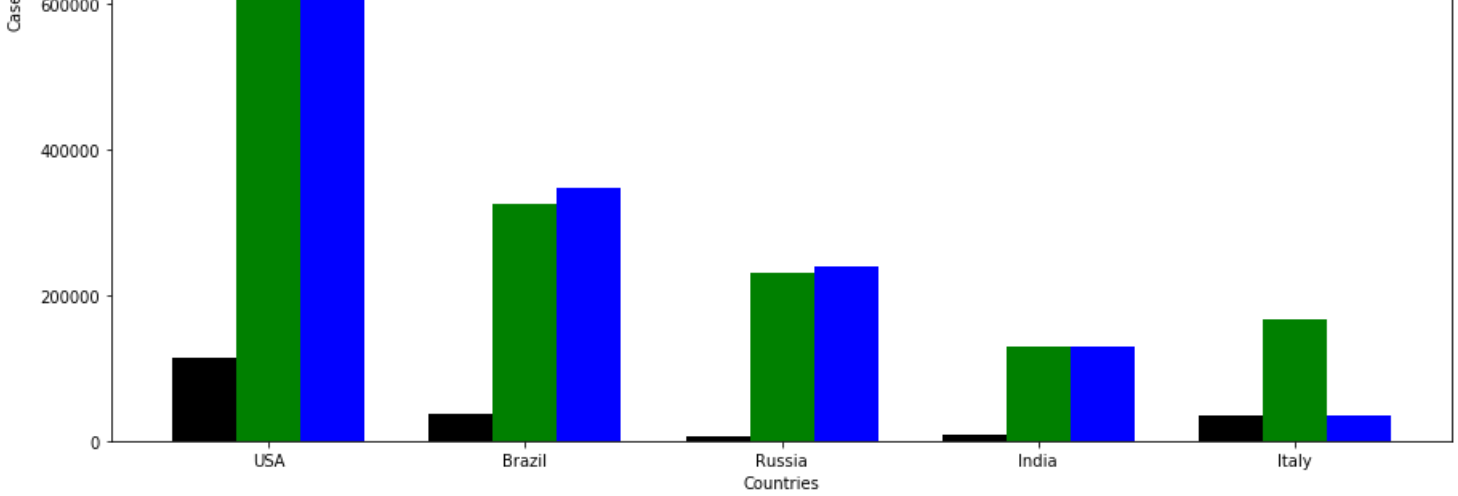
<matplotlib.axes._subplots.AxesSubplot at 0x11090f8c048>



In [31]:

```
# 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]
totalRecovery = [773480, 325602, 230688, 129095, 166584]
activeCases = [1139958, 347973, 239999, 129360, 34730]
country = ['USA', 'Brazil', 'Russia', 'India', 'Italy']

# Using numpy to group 3 different data with bars
X = np.arange(len(totalDeath))

# Passing the parameters to the bar function, this is the main function which creates the bar plot
# Using X now to align the bars side by side
plt.bar(X, totalDeath, color = 'black', width = 0.25)
plt.bar(X + 0.25, totalRecovery, color = 'g', width = 0.25)
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 country wise")

# 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

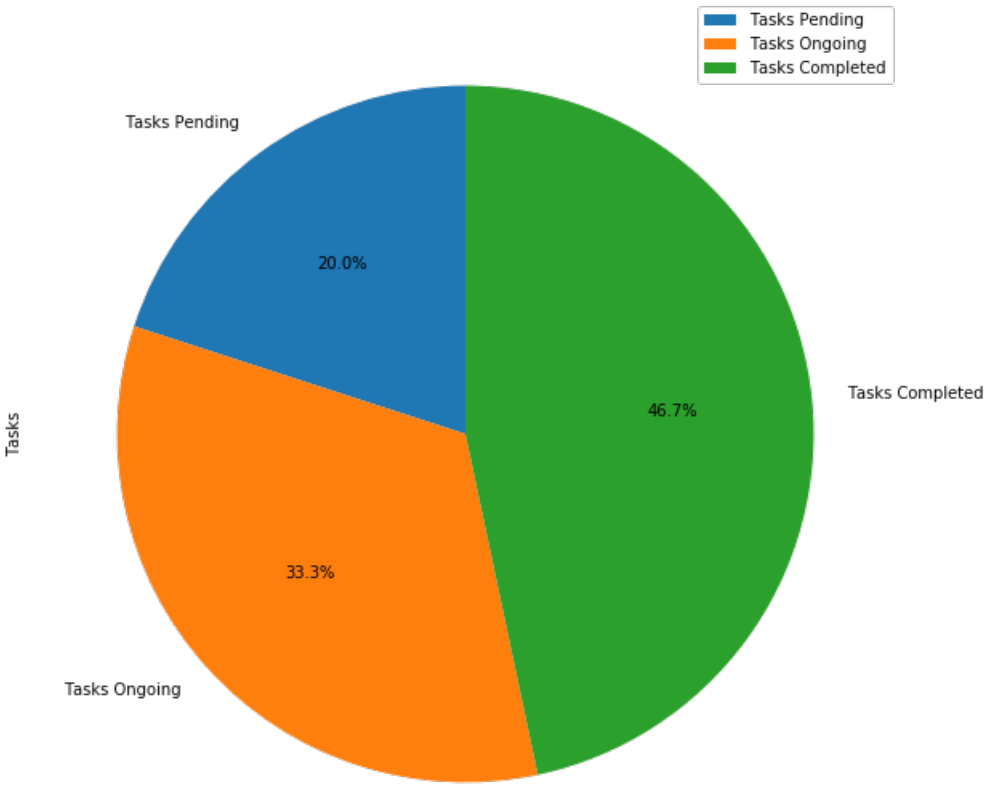
A pie chart is a type of data visualization that is used to illustrate numerical proportions in data.

In [40]:

```
# 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 default.
# Only pie chart needs labels to be data frame index
df.plot.pie(y='Tasks', figsize=(10,10),autopct='%1.1f%%', startangle=90)
```

Out[40]:

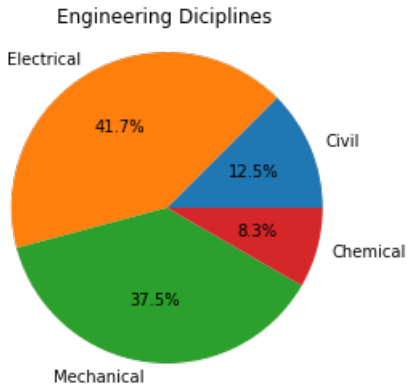
<matplotlib.axes._subplots.AxesSubplot at 0x1108eea4dc8>



We will plot this data on a pie chart with Matplotlib's ax.pie() method. The pie piece labels are defined as a list of strings, and the pie piece sizes are defined as a list of integers. The code section below builds a pie chart with four pie pieces, each pie piece labeled with a relative size auto-calculated to the nearest 10th of a percent.

In [41]:

```
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()
```

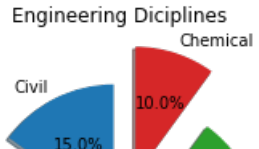


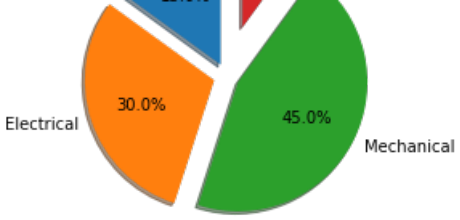
Pie pieces can be highlighted by "exploding" them out. Exploded pie pieces are applied to a Matplotlib pie chart by supplying the explode= keyword argument to the ax.pie() method. shadow=True and startangle= are two additional keyword arguments that can be passed to the ax.pie() method to control the angle and rotation of the pieces on a pie chart.

The code section below creates a pie chart with the pie pieces separated and the "Chemical" piece exploded out.

In [42]:

```
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, 30, 45, 10]

# Explode out the 'Chemical' pie piece by offsetting it a greater amount
explode = (0.1, 0.1, 0.1, 0.4)

fig, ax = plt.subplots()
ax.pie(sizes,
       explode=explode,
       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()
```





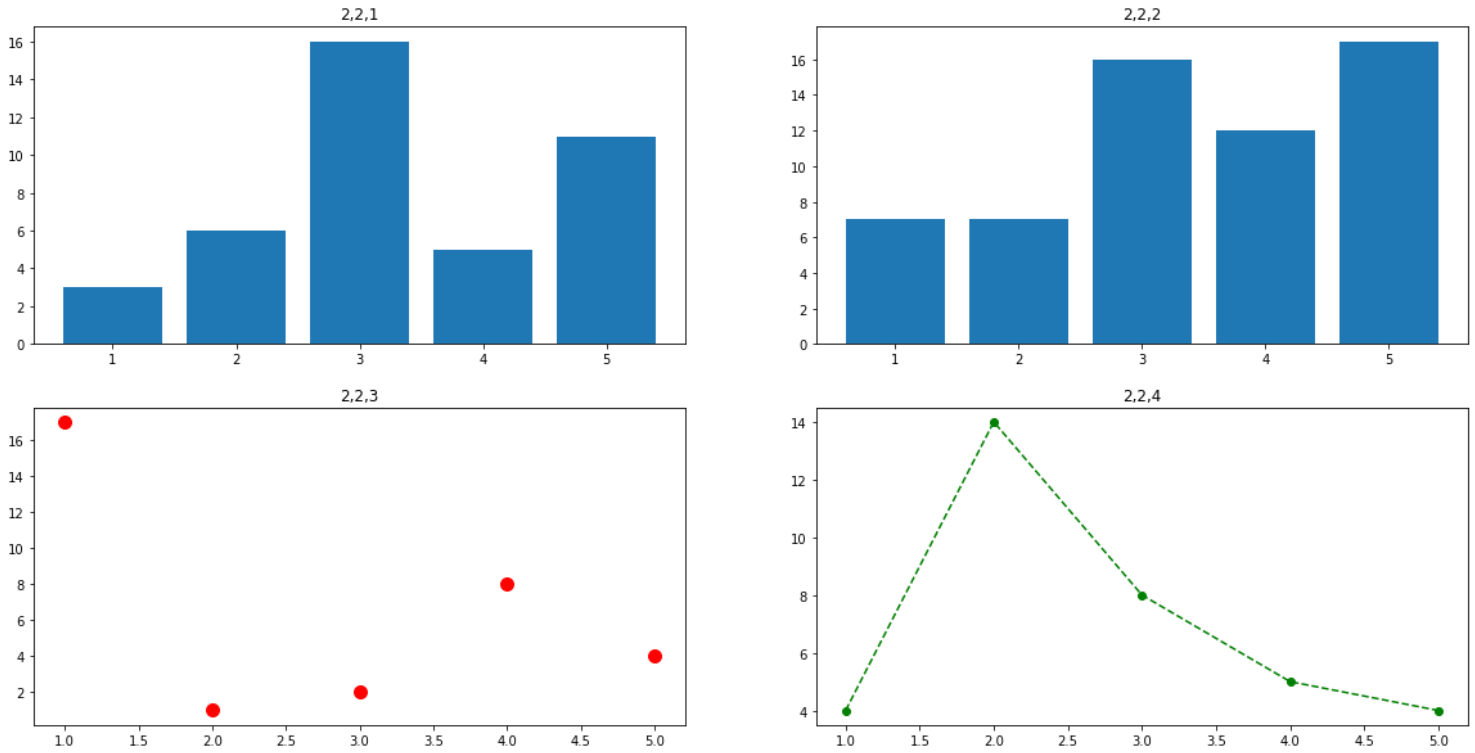
Subplots

In [43]:

```
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")
```

Out[43]:

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

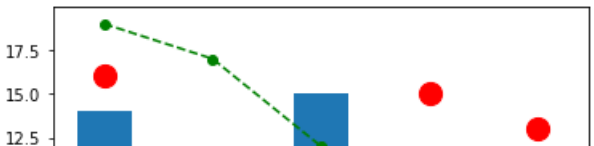


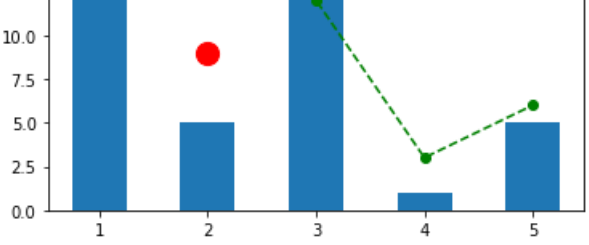
In [44]:

```
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='--')
```

Out[44]:

[<matplotlib.lines.Line2D at 0x1108ee23c88>]





Seaborn

• Seaborn is a Python data visualization library based on matplotlib • It provides a high level interface for drawing attractive and informative statistical graphics

In [18]:

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

In [19]:

```
os.chdir('D:/AnitaRJ/DATA SCIENCE/MScI_DataSci_Practicals/Practical7')
cars_data=pd.read_csv('Toyota.csv',index_col=0,na_values=["??","????"])
cars_data.size
```

Out[19]:

14360

In [16]:

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

Out[16]:

10960

In [22]:

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

Out[22]:

	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
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170

In [23]:

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

Out[23]:

	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

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170

Scatter plot

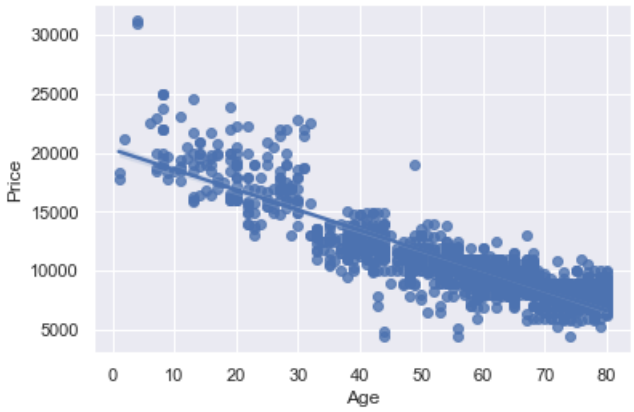
Scatter plot of Price vs Age with default arguments

In [25]:

```
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
```

Out[25]:

<matplotlib.axes._subplots.AxesSubplot at 0x21787092b08>

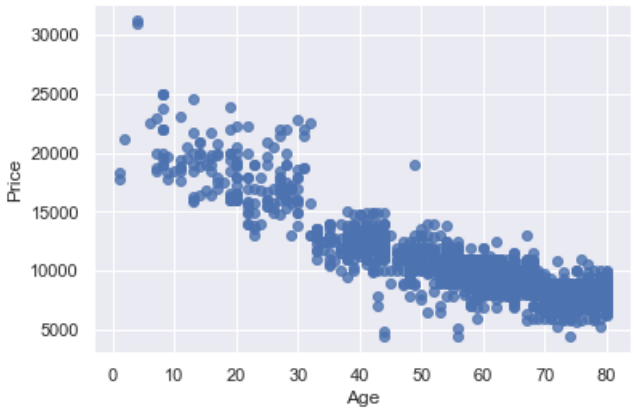


In [27]:

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

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x217870e4408>

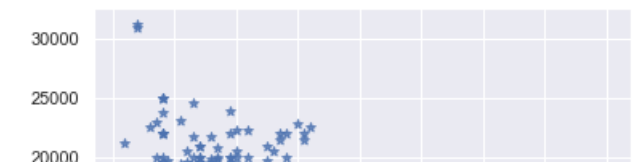


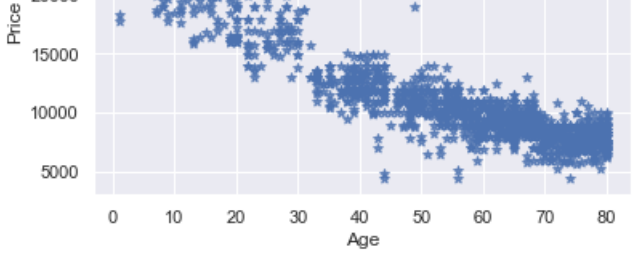
In [29]:

```
#Scatter plot of Price vs Age by customizing the appearance of markers
sns.regplot(x=cars_data['Age'], y=cars_data['Price'], marker="*", fit_reg=False)
```

Out[29]:

<matplotlib.axes._subplots.AxesSubplot at 0x217871da588>





In [30]:

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

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

sns.lmplot(x='Age', y='Price', data=cars_data, fit_reg=False, hue='FuelType', legend=True, palette="Set1")
```

Out[30]:

<seaborn.axisgrid.FacetGrid at 0x2178722bbc8>

Histogram

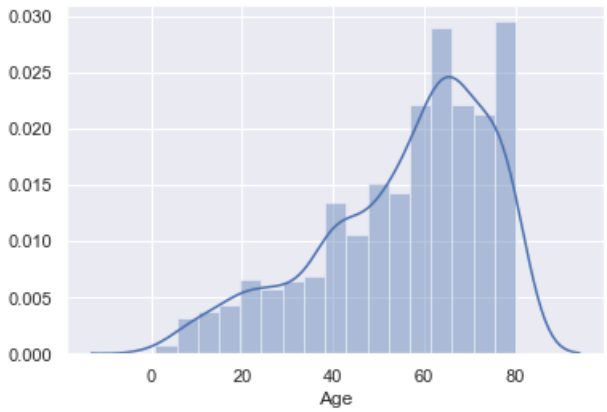
Histogram with default kernel density estimate

In [31]:

```
sns.distplot(cars_data['Age'])
```

Out[31]:

<matplotlib.axes._subplots.AxesSubplot at 0x217872a9b88>

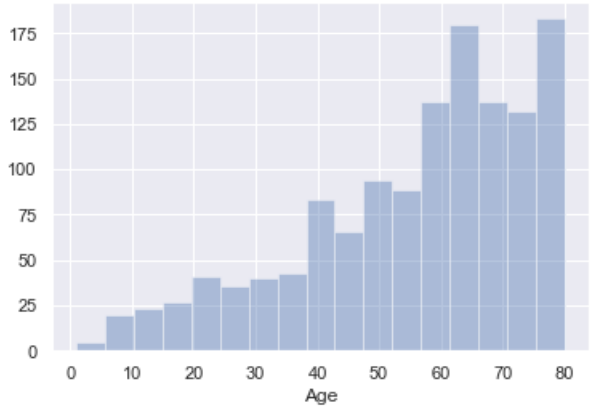


In [32]:

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

Out[32]:

<matplotlib.axes._subplots.AxesSubplot at 0x217871da488>

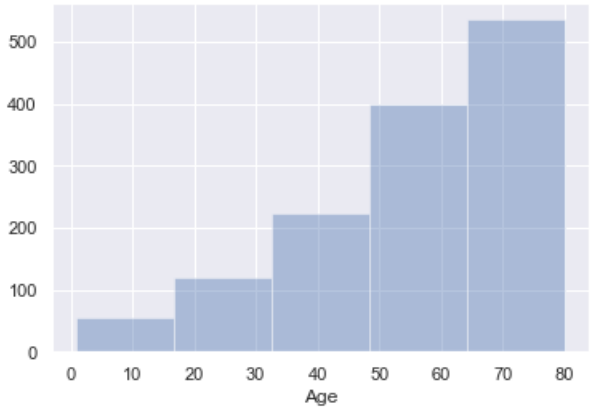


In [33]:

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

Out[33]:

<matplotlib.axes._subplots.AxesSubplot at 0x217873c21c8>



Bar plot

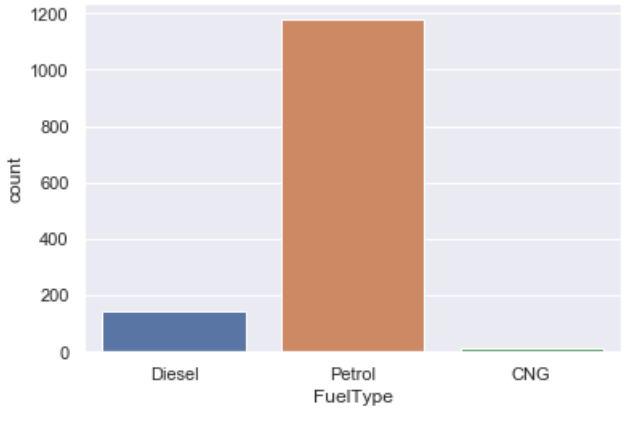
Frequency distribution of fuel type of the cars

In [34]:

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

Out[34]:

<matplotlib.axes._subplots.AxesSubplot at 0x21787434908>

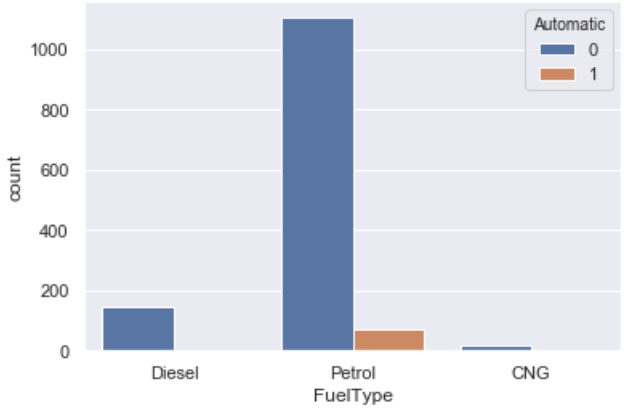


In [35]:

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

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

Out[35]:

<matplotlib.axes._subplots.AxesSubplot at 0x217874ab308>



In [37]:

```
pd.crosstab(index=cars_data['Automatic'], columns=cars_data['FuelType'],dropna=True)
```

Out[37]:

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.

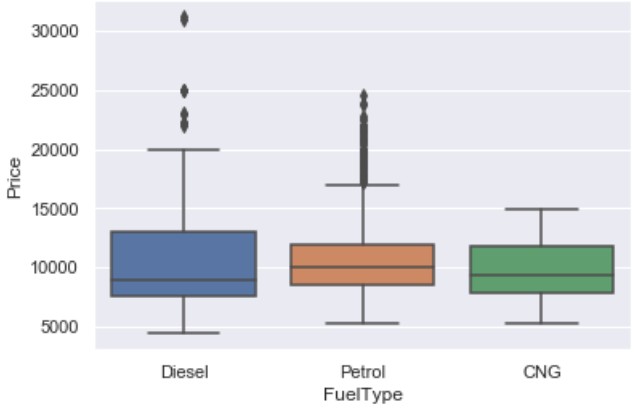
Price of the cars for various fuel types

In [38]:

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

Out[38]:

<matplotlib.axes._subplots.AxesSubplot at 0x2178750e4c8>



Grouped box and whiskers plot

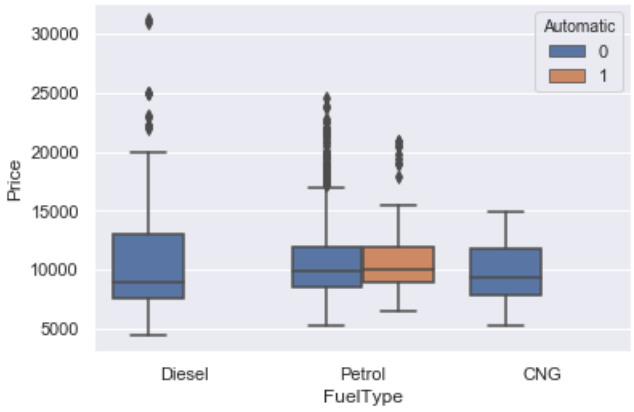
Grouped box and whiskers plot of Price vs FuelType and Automatic

In [39]:

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

Out[39]:

<matplotlib.axes._subplots.AxesSubplot at 0x21787267dc8>



Box

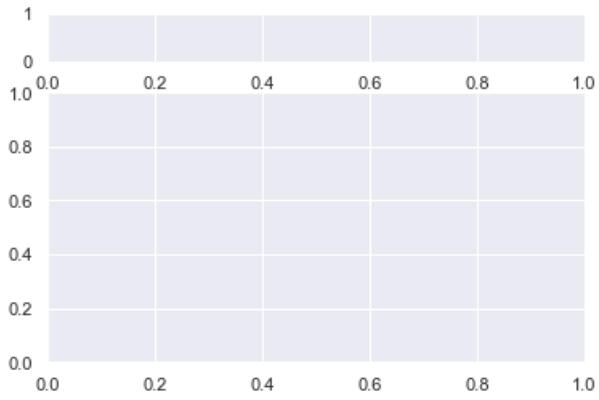
whiskers plot and Histogram

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

Split the plotting window into 2 parts

In [40]:

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



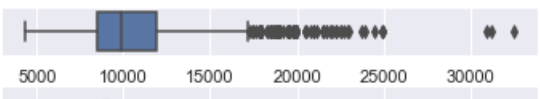
Now, add create two plots

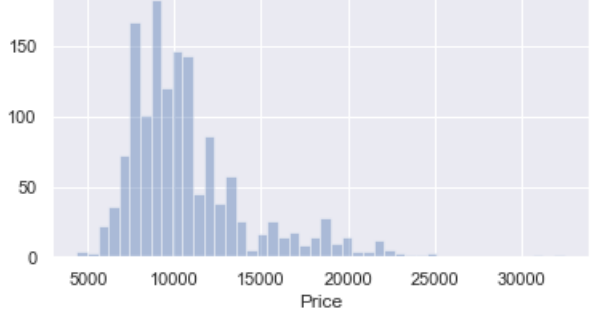
In [43]:

```
f,(ax_box,ax_hist)=plt.subplots(2,gridspec_kw={"height_ratios": (.15, .85)})
sns.boxplot(cars_data['Price'],ax=ax_box)
sns.distplot(cars_data['Price'],ax=ax_hist,kde=False)
```

Out[43]:

<matplotlib.axes._subplots.AxesSubplot at 0x217887d0d88>





Pairwise plots

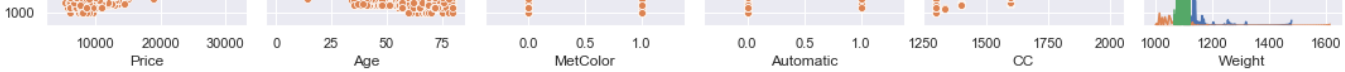
It is used to plot pairwise relationships in a dataset

Creates scatterplots for joint relationships and histograms for univariate distributions

In [48]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:288: UserWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:288: UserWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
```



Heatmap

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. I

In [16]:

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

In [17]:

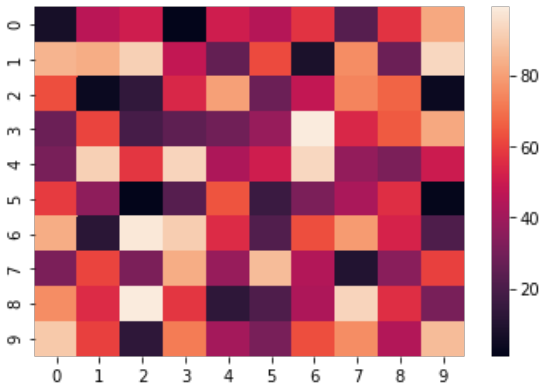
```
data=np.random.randint(1,100,size=(10,10))
print("The data to be plotted: \n")
print(data)
```

The data to be plotted:

```
[[ 7 46 51  1 51 45 57 23 57 82]
 [85 83 92 48 26 62  8 76 28 94]
 [63  4 14 54 80 28 48 74 67  4]
 [28 61 19 25 29 38 99 54 65 82]
 [31 92 58 93 43 51 94 37 32 50]
 [59 36  1 23 64 16 32 42 56  2]
 [83 12 98 91 55 22 63 79 53 21]
 [32 61 32 83 38 87 44 10 35 60]
 [76 55 99 58 13 21 43 93 56 31]
 [90 60 13 72 41 31 63 76 44 87]]
```

In [18]:

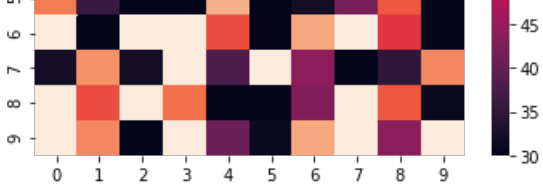
```
#Plotting Heatmap
hm=sns.heatmap(data=data)
plt.show()
```



If we set the vmin value to 30 and the vmax value to 70, then only the cells with values between 30 and 70 will be displayed. This is called anchoring the colormap.

In [22]:

```
hm = sns.heatmap(data=data,
                 vmin='30',
                 vmax='70')
plt.show()
```

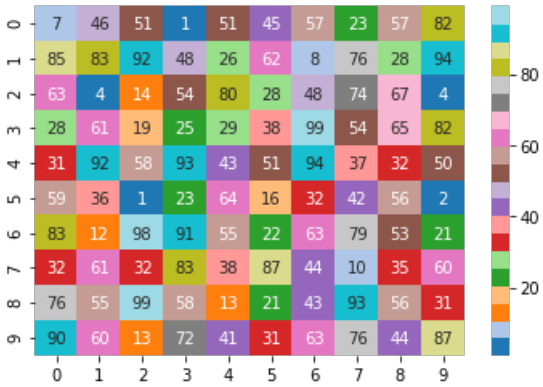


Choosing the colormap

In this, we will be looking at the cmap parameter. Matplotlib provides us with multiple colormaps, you can look at all of them here. Centering the cmap to 0 by passing the center parameter as 0.

In [46]:

```
# 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()
```



In [ ]: