Data Visualization

Import required packages

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

Matplotlib

> Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. Wikipedia

Scatter Plot

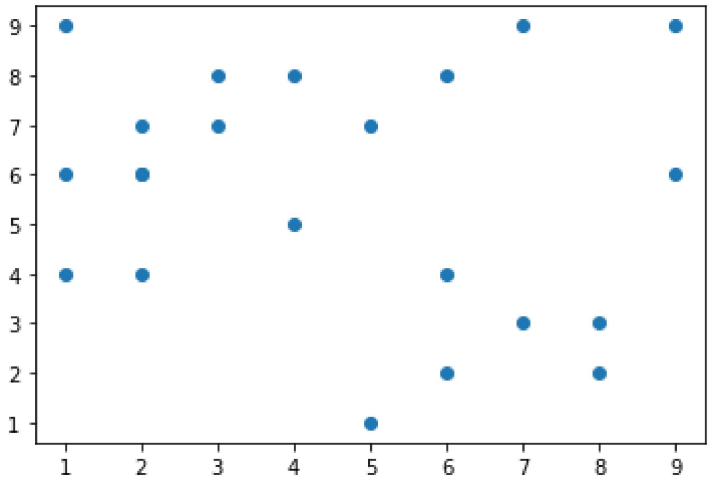
> A **scatter plot** (also called as ***scatterplot***, ***scatter graph***, ***scattergram*** or ***scatter graph***) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data. Wikipedia

Scatter plots can be created in Python by using `matplotlib.pyplot.scatter()` or `matplotlib.axes._subplots.AxesSubplot.scatter()` function where matplotlib is a commonly used Python package for producing 2D and 3D graphs.

Scatter Plot using hardcoded data

In [2]:
```
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, 3, 6, 8, 9, 2, 5, 7, 9, 4, 6, 8, 9, 1, 3, 6, 7]

plt.scatter(x,y)
```

Out[2]:
```
<matplotlib.collections.PathCollection at 0x24c2751f9d0>
```



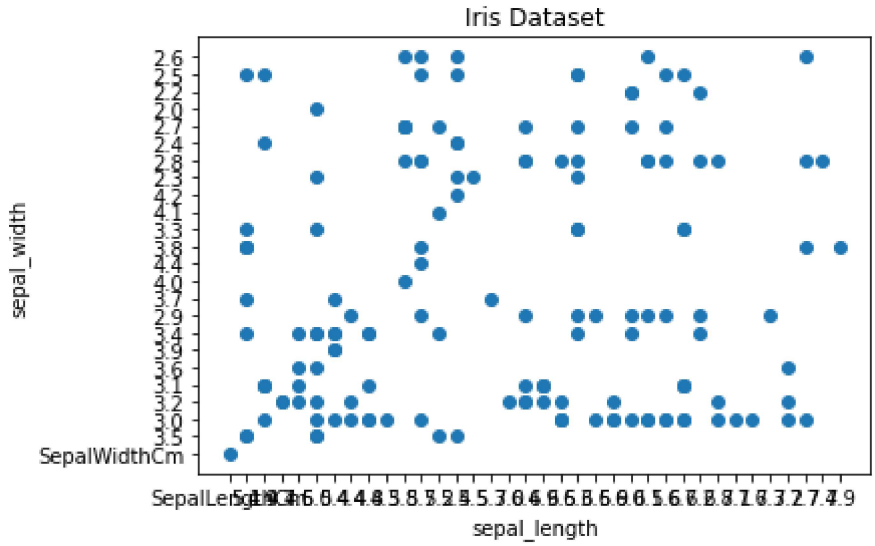
Scatter plot from a CSV file

In [3]:
```
iris = pd.read_csv('data_sets/Iris.csv', names=['sepal_length', 'sepal_width', 'petal_l
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 [4]:
```
fig, ax = plt.subplots()

ax.scatter(iris['sepal_length'], iris['sepal_width'])
ax.set_title('Iris Dataset')
ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
```

Out[4]:
```
Text(0, 0.5, 'sepal_width')
```



Line Chart

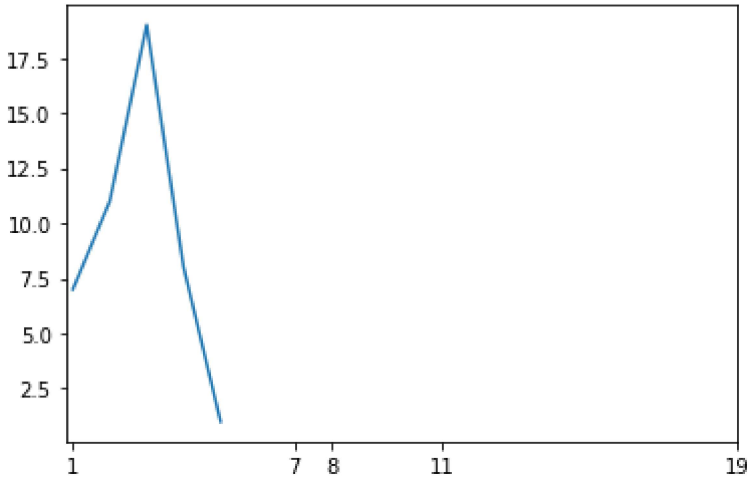
> A **line chart** (or ***line plot*** or ***line graph*** or ***curve chart***) is a type of chart which displays information as a series of data points called '*markers*' connected by straight line segments. It is similar to a scatter plot except that the measurement points are ordered (typically with their x-axis value) and joined with straight line segments. Wikipedia

Scatter plots can be created in Python by using `matplotlib.pyplot.plot()` or `matplotlib.axes._subplots.AxesSubplot.plot()` function.

Line chart using hardcoded data and possessing an ordered x-axis values

In [5]:
```
x = range(1, 6)
y = np.random.randint(1, 20, 5)
plt.plot(x, y)

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

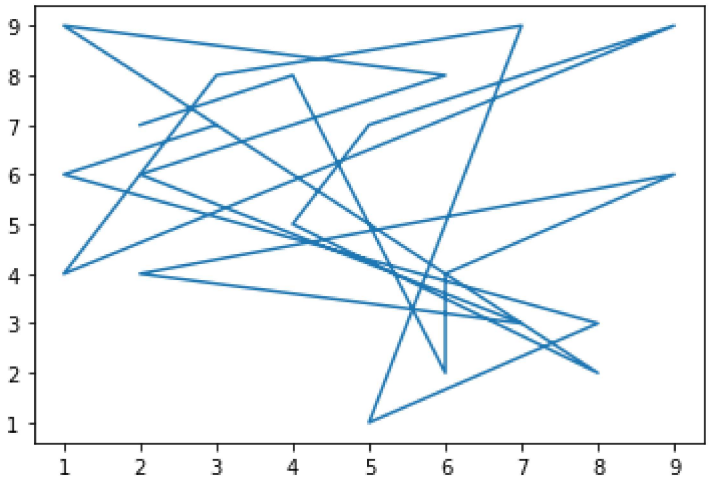
Out[5]:
```
([<matplotlib.axis.XTick at 0x24c29757e80>,
  <matplotlib.axis.XTick at 0x24c29757e50>,
  <matplotlib.axis.XTick at 0x24c29757520>,
  <matplotlib.axis.XTick at 0x24c29780c10>,
  <matplotlib.axis.XTick at 0x24c297903a0>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



Line chart using hardcoded data and with an unordered x-axis values

In [6]:
```
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, 3, 6, 8, 9, 2, 5, 7, 9, 4, 6, 8, 9, 1, 3, 6, 7]

plt.plot(x,y)
```

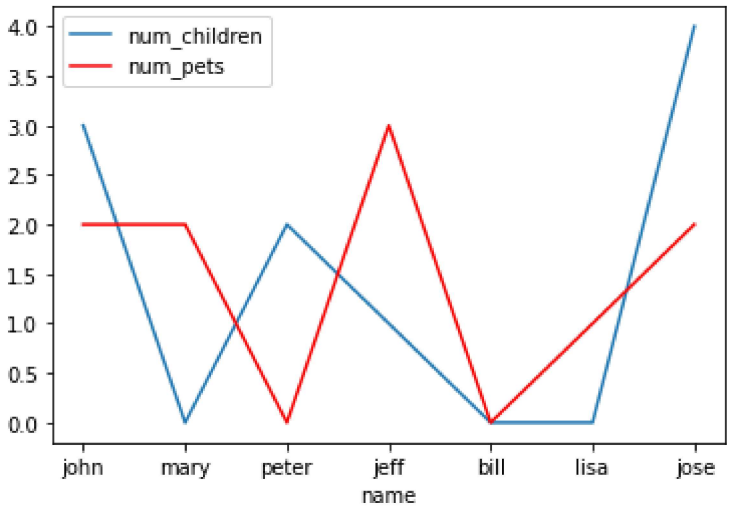
Out[6]:
```
[<matplotlib.lines.Line2D at 0x24c297e7520>]
```



Plotting multiple line charts within a single graph from a pandas DataFrame

In [7]:
```
df = pd.DataFrame({
    'name' : ['john', 'mary', 'peter', 'jeff', 'bill', 'lisa', 'jose'],
    'age' : [23, 22, 27, 33, 20, 19, 45],
    'gender' : ['M', 'F', 'M', 'M', 'M', 'F', 'M'],
    'state' : ['California', 'dc', 'California', 'uk', 'California', 'dc', 'California'
    'num_children' : [3, 0, 2, 1, 0, 0, 4],
    'num_pets' : [2, 2, 0, 3, 0, 1, 2],
})

# gca stands for Get Current Axes
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[7]:
```
<AxesSubplot:xlabel='name'>
```

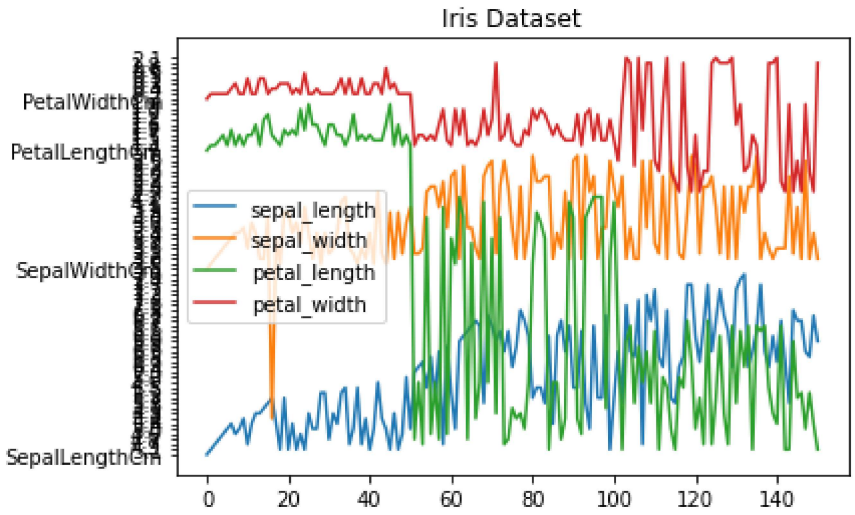


Plotting multiple line charts within a single graph from a CSV file

In [8]:
```
columns = iris.columns.drop(['class'])
x_data = range(0, iris.shape[0])
```

```
fig, ax = plt.subplots()
for column in columns:
  ax.plot(x_data, iris[column], label = column)
ax.set_title('Iris Dataset')
ax.legend()
```

Out[8]:
```
<matplotlib.legend.Legend at 0x24c298cd3d0>
```



Histogram

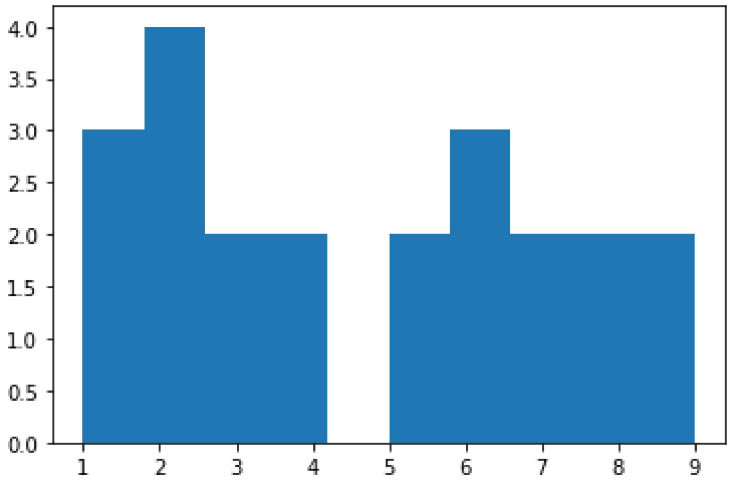
> A **histogram** is an approximate representation of the distribution of numerical data.
> Wikipedia

A histogram can be created in Python by using `matplotlib.pyplot.hist()` or `matplotlib.axes._subplots.AxesSubplot.hist()` function.

Histogram using hardcoded data

In [9]:
```
x = [2, 4, 6, 6, 9, 2, 7, 2, 6, 1, 8, 4, 5, 9, 1, 2, 3, 7, 5, 8, 1, 3]

plt.hist(x)
```

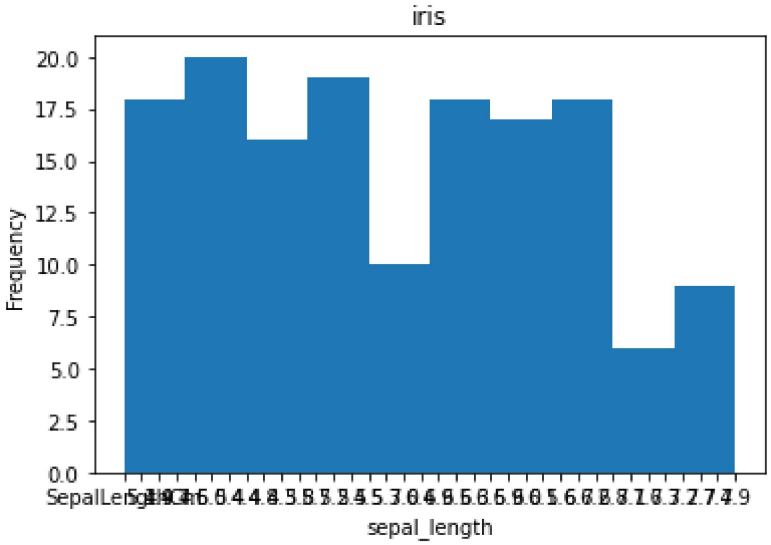
Out[9]:
```
(array([3., 4., 2., 2., 0., 2., 3., 2., 2., 2.]),
 array([1. , 1.8, 2.6, 3.4, 4.2, 5. , 5.8, 6.6, 7.4, 8.2, 9. ]),
 <BarContainer object of 10 artists>)
```



Histogram from a pandas DataFrame

In [10]:
```
fig, ax = plt.subplots()

ax.hist(iris['sepal_length'])
ax.set_title('iris')
ax.set_xlabel('sepal_length')
ax.set_ylabel('Frequency')
```

Out[10]:
```
Text(0, 0.5, 'Frequency')
```



Bar chart

> A **bar chart** or ***bar graph*** is a chart or graph that represents categorical data with rectangular bars with heights or lengths proportional to the values they represent. These bars can be plotted vertically or horizontally. A vertical bar chart is sometimes called a ***column chart***. Wikipedia

A bar chart can be created in Python by using `matplotlib.pyplot.bar()` or `matplotlib.axes._subplots.AxesSubplot.bar()` function.

Bar plot from a CSV file

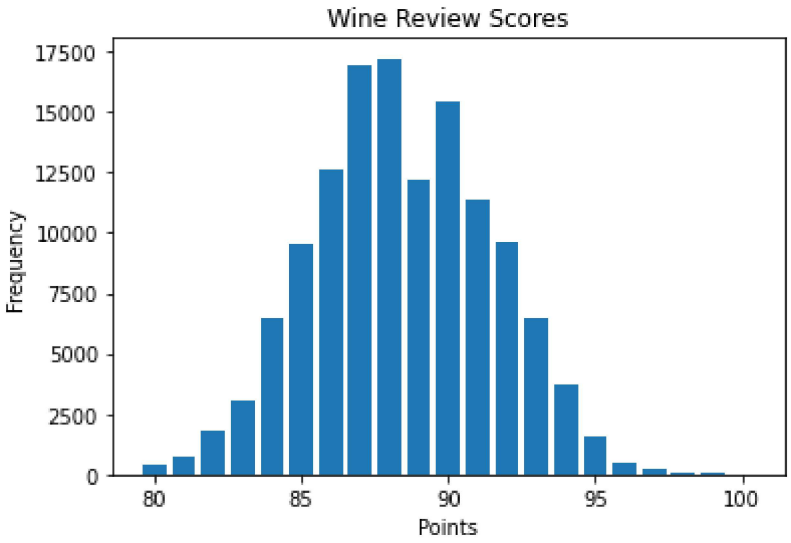
In [11]:
```
wine_reviews = pd.read_csv('data_sets/winemag-data-130k-v2.csv')
wine_reviews.head()
```

Out[11]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	taster_
0	0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	O
1	1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roge
2	2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul G
3	3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alex Pe
4	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 G

In [12]:
```
data = wine_reviews['points'].value_counts()
points = data.index
frequency = data.values

fig, ax = plt.subplots()
ax.bar(points, frequency)
ax.set_title('Wine Review Scores')
ax.set_xlabel('Points')
ax.set_ylabel('Frequency')
```

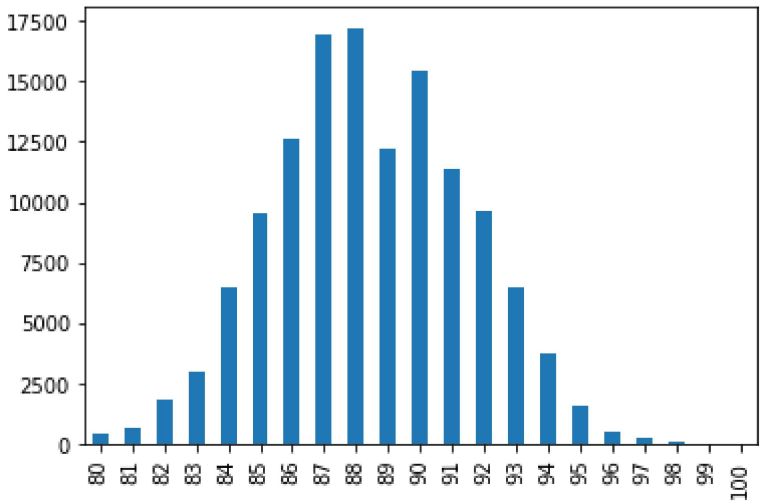
Out[12]:
```
Text(0, 0.5, 'Frequency')
```



Bar plot for wine_reviews DataFrame mapping the score to the number of wines obtaining the score

In [13]:
```
wine_reviews['points'].value_counts().sort_index().plot.bar()
```

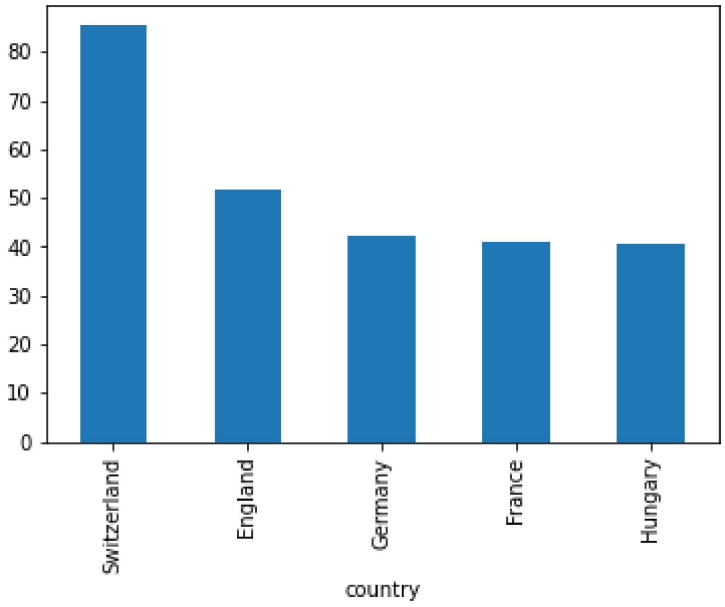
Out[13]:
```
<AxesSubplot:>
```



Bar plot for wine_reviews DataFrame mapping the average price of wine to the countries in a descending order (first 5 entries)

In [14]:
```
wine_reviews.groupby("country").price.mean().sort_values(ascending = False)[ : 5].plot.
```

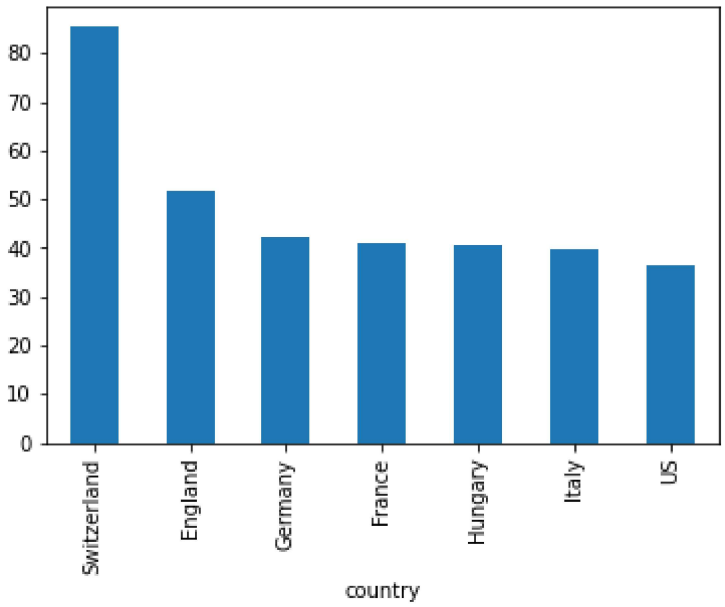
Out[14]:
```
<AxesSubplot:xlabel='country'>
```



Bar plot for wine_reviews DataFrame mapping the average price of wine to the countries in a descending order (first 7 entries)

In [15]:
```
wine_reviews.groupby("country").price.mean().sort_values(ascending = False)[ : 7].plot.
```

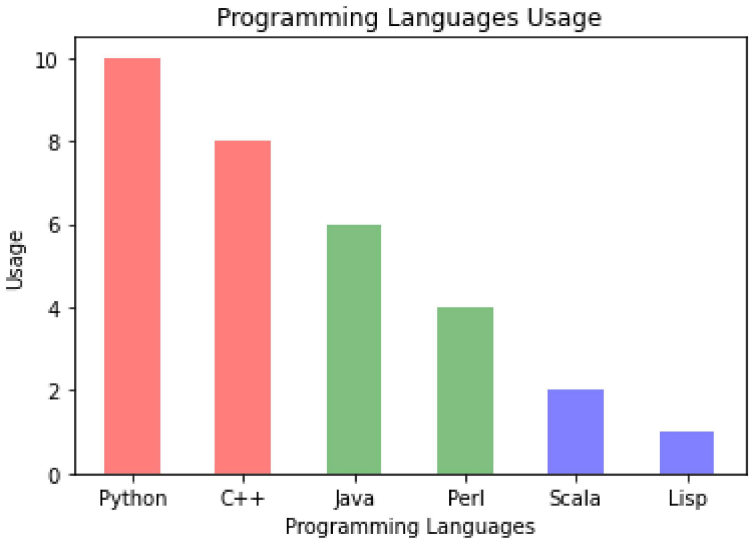
Out[15]: <AxesSubplot:xlabel='country'>



Bar charts with hardcoded data

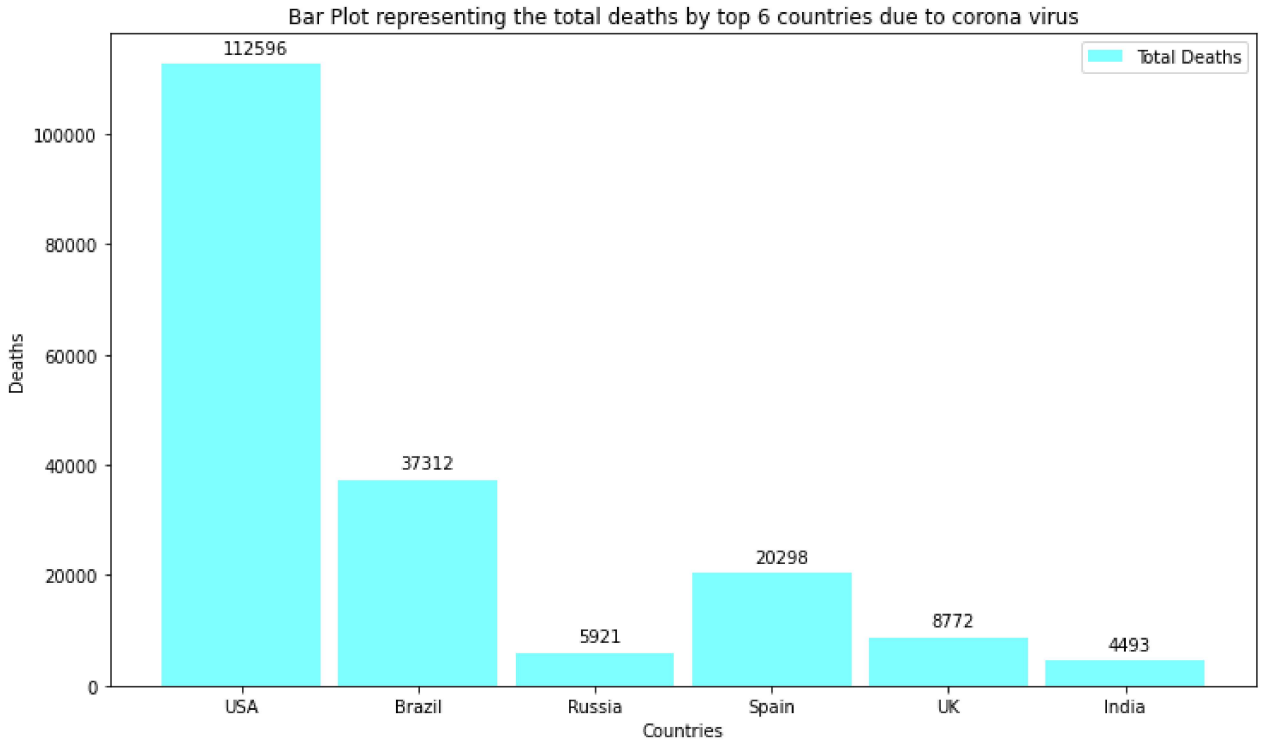
In [16]:
```
objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10, 8, 6, 4, 2, 1]

plt.bar(y_pos, performance, width = 0.5, align = 'center', alpha = 0.5, color = ['r', '
plt.xticks(y_pos, objects)
plt.ylabel('Usage')
plt.xlabel('Programming Languages')
plt.title('Programming Languages Usage')
```

Out[16]:
```
Text(0.5, 1.0, 'Programming Languages Usage')
```



Bar charts with hardcoded data and annotations

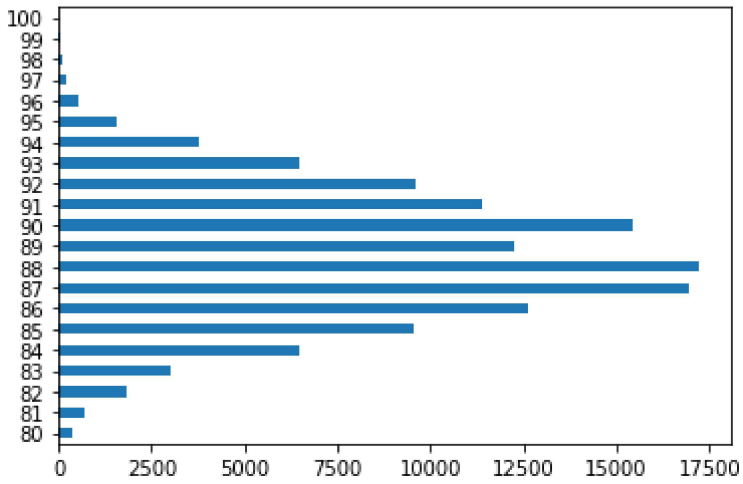
In [17]:
```
countries = ['USA', 'Brazil', 'Russia', 'Spain', 'UK', 'India']
totalDeaths=[112596, 37312, 5921, 20298, 8772, 4493]

i = 1.0
j = 2000

plt.figure(figsize = (12, 7))
plt.bar(countries, totalDeaths, width = 0.9, align = 'center', alpha = 0.5, color = 'cy
for i in range(len(countries)):
    plt.annotate(totalDeaths[i], (-0.1 + i, totalDeaths[i] + j))
plt.legend(labels = ['Total Deaths'])
plt.title('Bar Plot representing the total deaths by top 6 countries due to corona viru
plt.xlabel('Countries')
plt.ylabel('Deaths')
plt.savefig('output_files/bar_plot_1.png')
```



Horizontal Bar plot for wine_reviews DataFrame mapping the score to the number of wines obtaining the score

In [18]:
```
wine_reviews['points'].value_counts().sort_index().plot.barh()
```

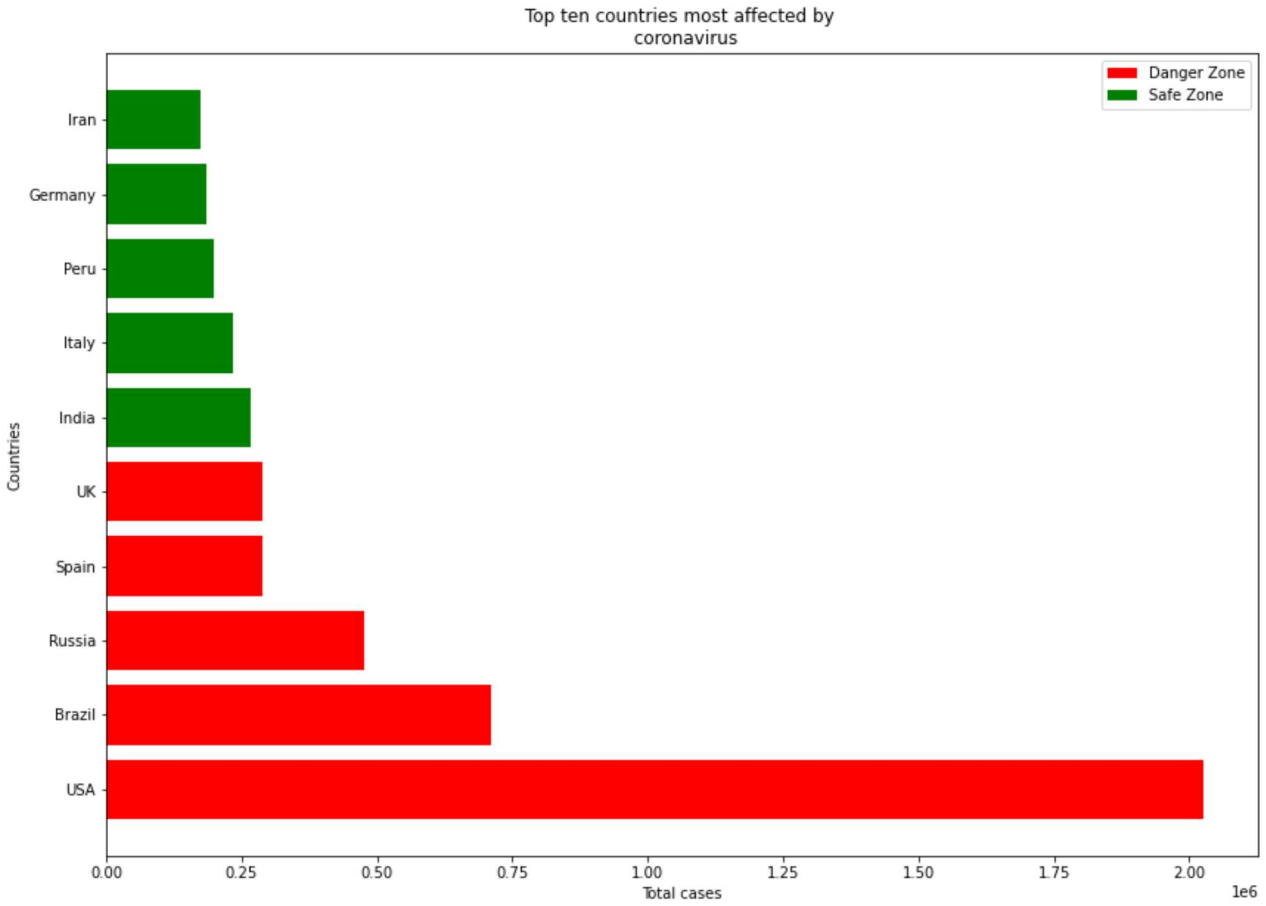
Out[18]: <AxesSubplot:>



Adding multiple labels to a bar plot

In [19]:
```
plt.figure(figsize = [14, 10])

plt.barh(['USA', 'Brazil', 'Russia', 'Spain', 'UK'], [2026493, 710887, 476658, 288797,
plt.barh(['India', 'Italy', 'Peru', 'Germany', 'Iran'], [265928, 235278, 199696, 186205

plt.legend()

plt.xlabel('Total cases')
```

```
plt.ylabel('Countries')

plt.title('Top ten countries most affected by \n coronavirus')

plt.savefig("output_files/bar_plot_2.png")

plt.show()
```



Stacked bar plots

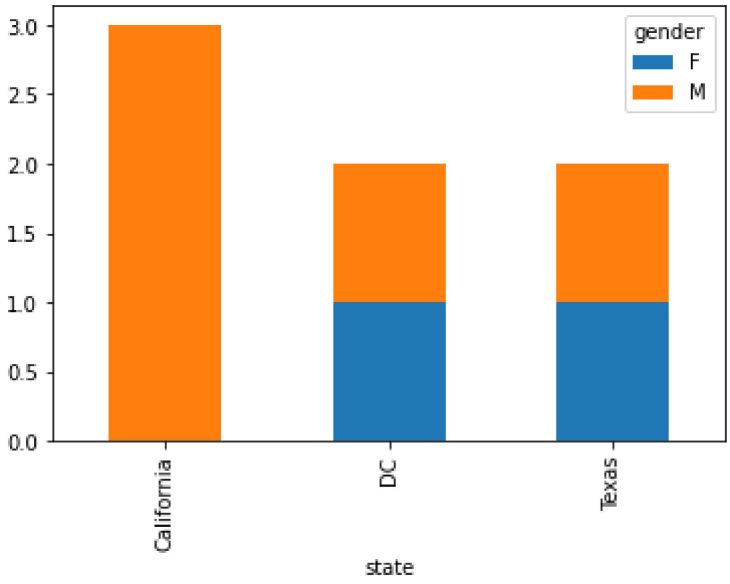
Bar plot representing the gender wise distribution of participants and the state they belong to

In [20]:

```
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],
})

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

Out[20]:

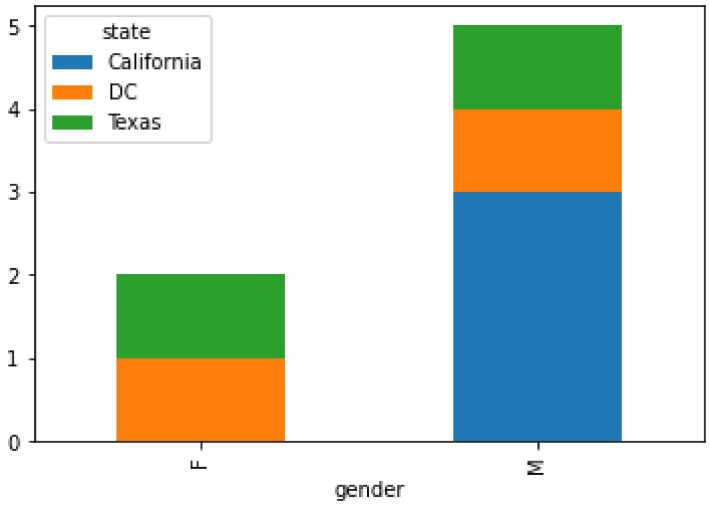
```
<AxesSubplot:xlabel='state'>
```



Bar plot representing the state wise distribution of participants and their gender

In [21]:
```
df.groupby(['gender', 'state']).size().unstack().plot(kind = "bar", stacked = True)
```

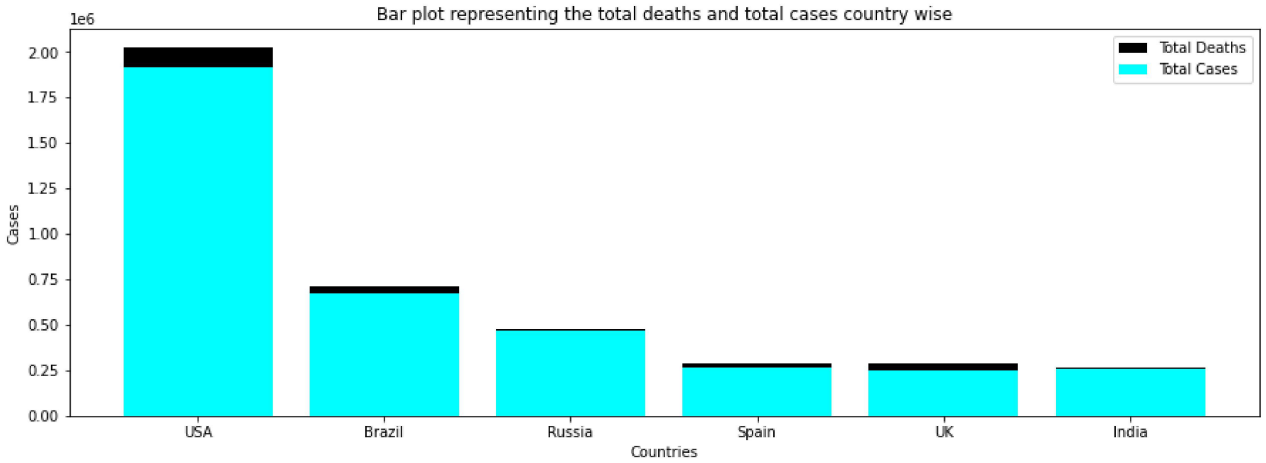
Out[21]:
```
<AxesSubplot:xlabel='gender'>
```



Bar plot representing the total deaths and total cases country wise

In [22]:
```
plt.figure(figsize = [15, 5])

countries = ['USA', 'Brazil', 'Russia', 'Spain', 'UK', 'India']

totalCases = (2026493, 710887, 476658, 288797, 287399, 265928)

totalDeaths = (113055, 37312, 5971, 27136, 40597, 7473)

for i in range(len(countries)):
    plt.bar(countries[i], totalDeaths[i], bottom = totalCases[i] - totalDeaths[i], colo
    plt.bar(countries[i], totalCases[i] - totalDeaths[i], color = "cyan")

plt.legend(labels = ['Total Deaths', 'Total Cases'])
```

```
plt.title("Bar plot representing the total deaths and total cases country wise")

plt.xlabel("Countries")
plt.ylabel("Cases")

plt.savefig("output_files/bar_plot_3.png")

plt.show()
```



Plotting bars next to each other (No stacking)

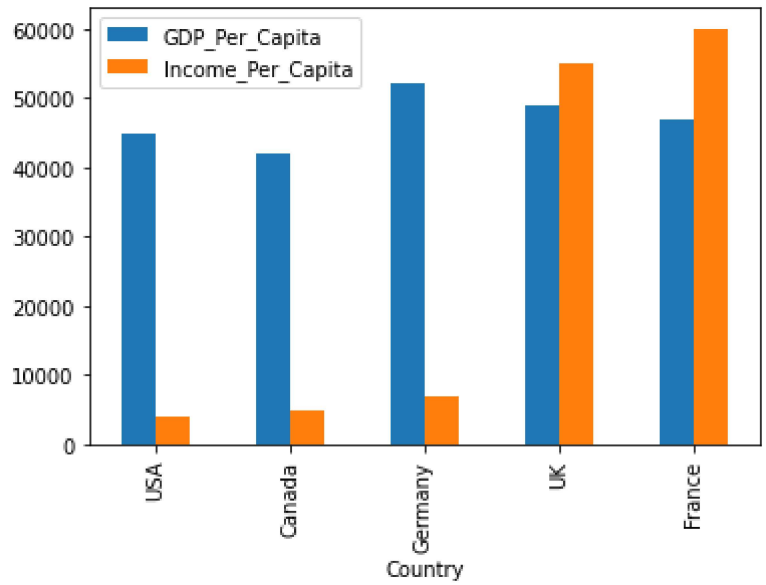
Two bars next to each other

In [23]:

```
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)
df.plot(x = 'Country', y = ['GDP_Per_Capita', 'Income_Per_Capita'], kind = 'bar')
```

Out[23]:

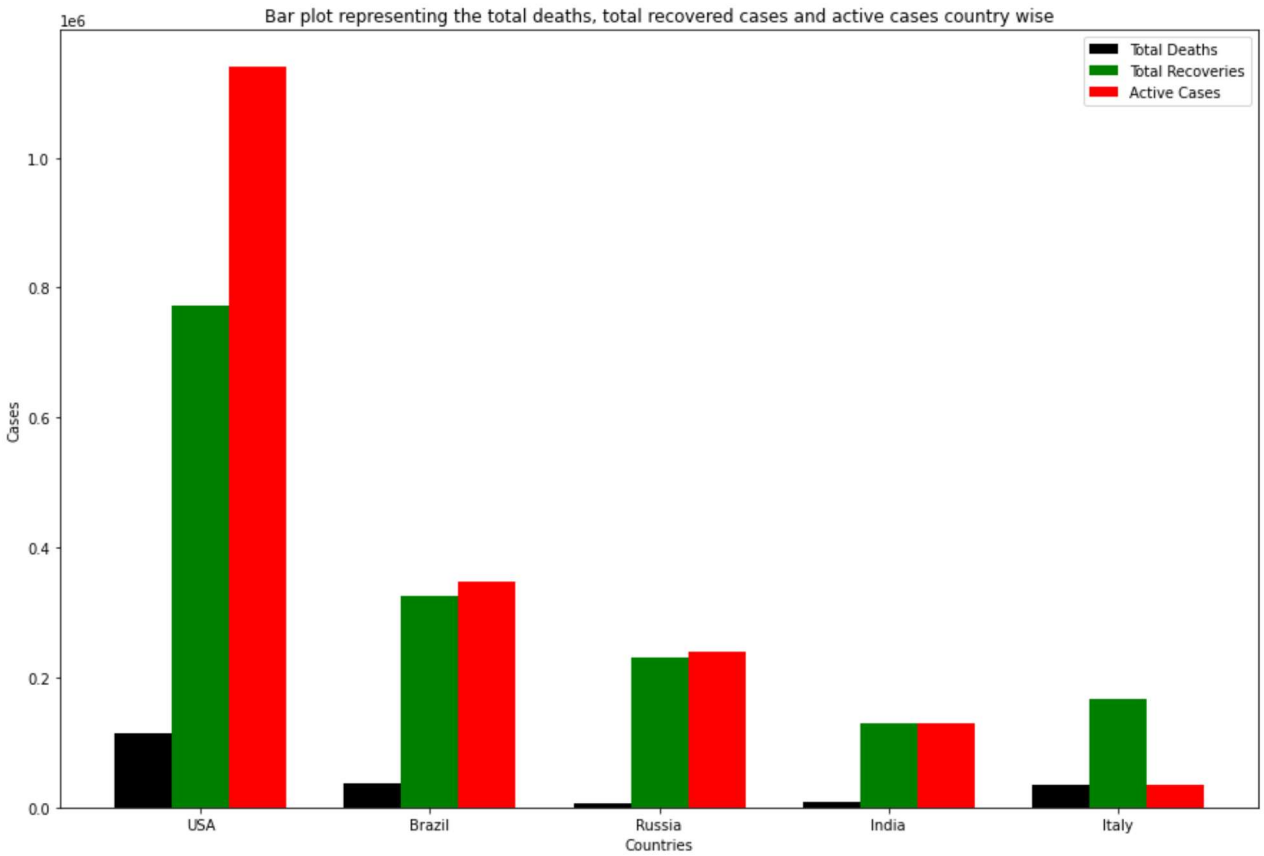
```
<AxesSubplot:xlabel='Country'>
```



Three bars next to each other

In [24]:

```
plt.figure(figsize = [15, 10])

totalDeath = [113055, 37312, 5971, 7473, 33964]
totalRecovery = [773480, 325602, 230688, 129095, 166584]
activeCases = [1139958, 347973, 239999, 129360, 34730]
country = ['USA', 'Brazil', 'Russia', 'India', 'Italy']

X = np.arange(len(totalDeath))

plt.bar(X, totalDeath, color = 'black', width = 0.25)
plt.bar(X + 0.25, totalRecovery, color = 'green', width = 0.25)
plt.bar(X + 0.50, activeCases, color = 'red', width = 0.25)

plt.legend(['Total Deaths', 'Total Recoveries', 'Active Cases'])

plt.xticks([i + 0.25 for i in range(5)], country)

plt.title("Bar plot representing the total deaths, total recovered cases and active cas

plt.xlabel("Countries")

plt.ylabel("Cases")

plt.savefig("output_files/bar_plot_4.png")

plt.show()
```



Pie Chart

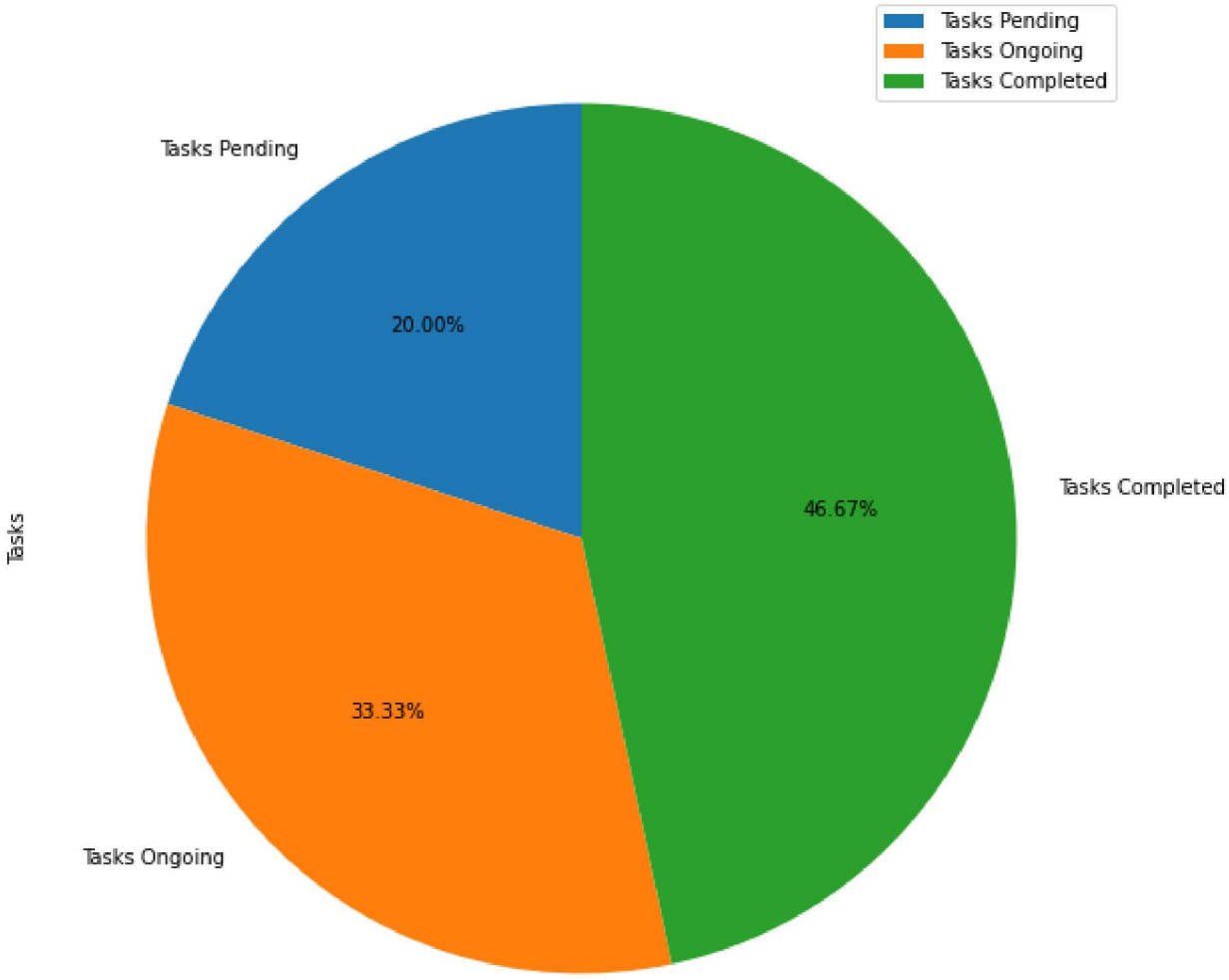
> A pie chart (or a circle chart) is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice (and consequently its central angle and area), is proportional to the quantity it represents. Wikipedia

A pie chart can be drawn in Python by using the `matplotlib.pyplot.pie()` or `matplotlib.axes._subplots.AxesSubplot.pie()` function.

Standard Pie Chart from a pandas DataFrame

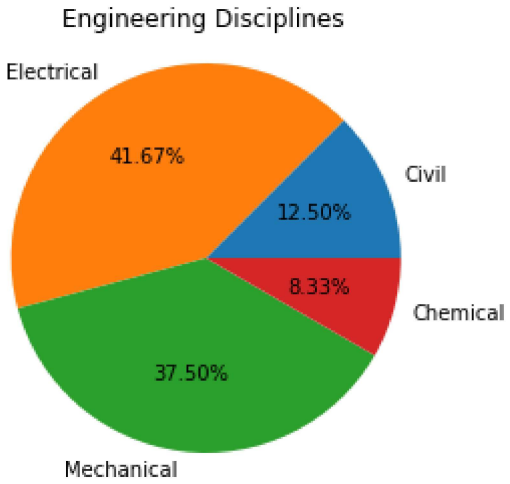
In [25]:
```
data = {
    'Tasks' : [300, 500, 700],
    'Task Type' : ['Tasks Pending', 'Tasks Ongoing', 'Tasks Completed']
}

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

df.plot.pie(y = 'Tasks', figsize = (10, 10), autopct = '%1.2f%%', startangle = 90)
```

Out[25]: <AxesSubplot:ylabel='Tasks'>



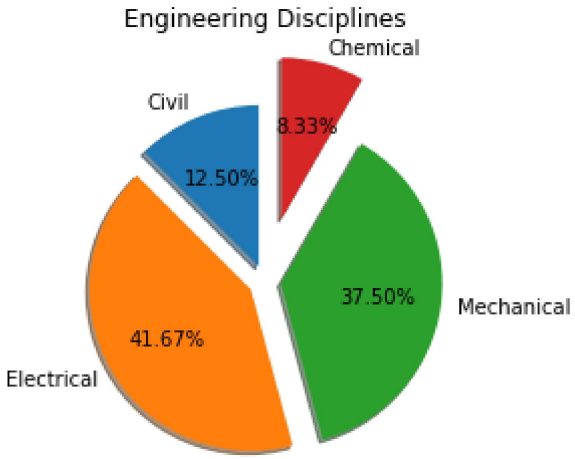
Pie chart with labels

In [26]:
```
%matplotlib inline

labels = ['Civil', 'Electrical', 'Mechanical', 'Chemical']
sizes = [15, 50, 45, 10]

fig, ax = plt.subplots()
ax.pie(sizes, labels = labels, autopct = "%1.2f%%")
ax.axis('equal')
ax.set_title('Engineering Disciplines')

plt.show()
```



Exploded Pie Chart

In [27]:
```
labels = ['Civil', 'Electrical', 'Mechanical', 'Chemical']
sizes = [15, 50, 45, 10]

explode = (0.1, 0.1, 0.1, 0.4)

fig, ax = plt.subplots()
ax.pie(sizes,
       explode = explode,
       labels = labels,
       autopct = "%1.2f%%",
       shadow = True,
       startangle = 90)

ax.axis('equal')
ax.set_title('Engineering Disciplines')

plt.show()
```

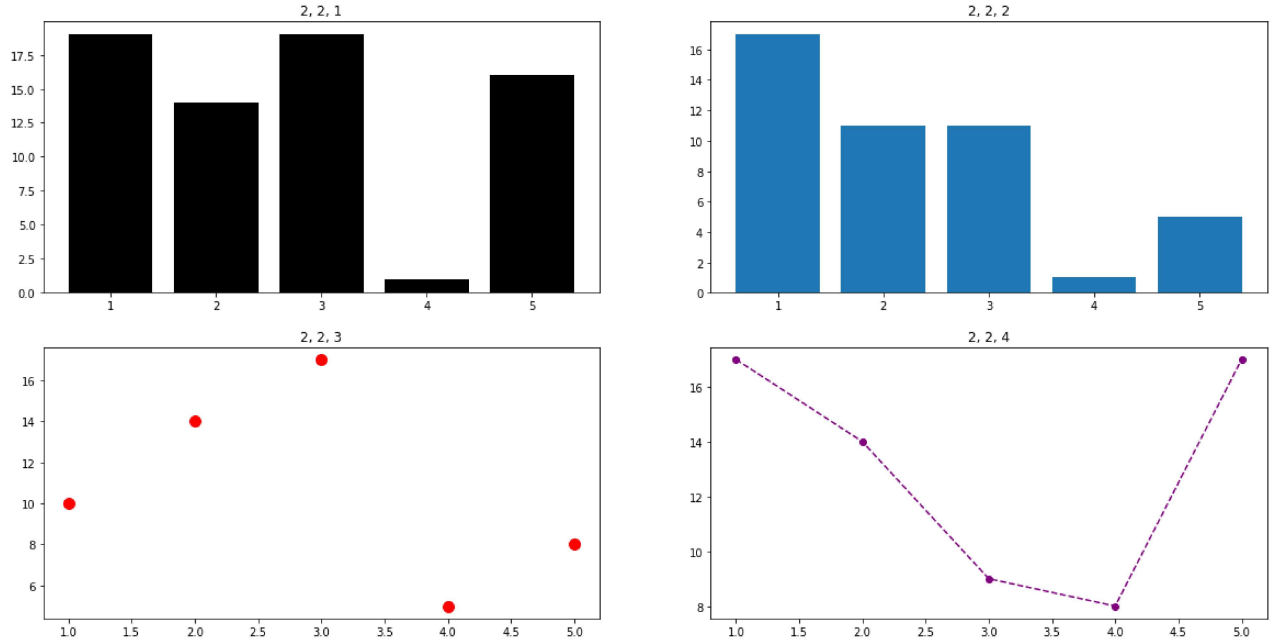


Multiple Subplots

Plotting multiple plots into a single figure

In [28]:
```
plt.figure(figsize = (20, 10))
plt.subplot(2, 2, 1)

plt.bar(range(1, 6), np.random.randint(1, 20, 5), color = "black")
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)
plt.scatter(range(1, 6), np.random.randint(1, 20, 5), s = 100, color = 'red')
plt.title("2, 2, 3")

plt.subplot(2, 2, 4)
plt.plot(range(1, 6), np.random.randint(1, 20, 5), marker = "o", color = "purple", line
plt.title("2, 2, 4")
```

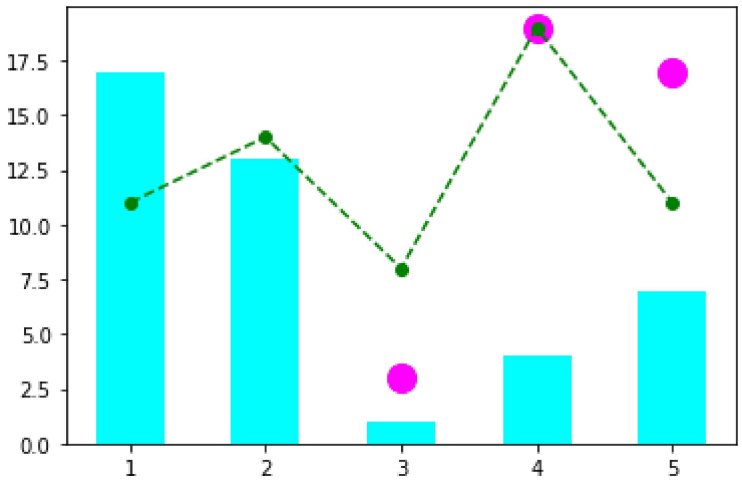
Out[28]:
```
Text(0.5, 1.0, '2, 2, 4')
```



Plotting multiple plots into a single plot

In [29]:
```
plt.bar(range(1, 6), np.random.randint(1, 20, 5), width = 0.5, color = 'cyan')
plt.scatter(range(1, 6), np.random.randint(1, 20, 5), s = 200, color = 'magenta')
plt.plot(range(1, 6), np.random.randint(1, 20, 5), marker = "o", color = 'green', lines
```

Out[29]:
```
[<matplotlib.lines.Line2D at 0x24c2b15a9a0>]
```



Seaborn

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

Load and process data to be used and store it in a dataframe

In [30]:
```
# The os.chdir() is used to change the current working directory.
# It is not used here as all related files are stored in a folder
# which is present in the current working directory i.e.
# ~/Practicals/Semester%201/Fundamentals%20of%20Data%20Science/
```

```
# and the folder in question is data_sets.
# The %20 is a placeholder for the ' ' (space) character.

# os.chdir("~/Practicals/Semester%201/Fundamentals%20of%20Data%20Science/")
cars_data = pd.read_csv('data_sets/Toyota.csv', index_col = 0, na_values = ("??", "????
cars_data.size
```

Out[30]:
```
14360
```

In [31]:
```
cars_data.dropna(axis = 0, inplace = True)
cars_data.size
```

Out[31]:
```
10960
```

In [32]:
```
cars_data = pd.read_csv('data_sets/Toyota.csv', index_col = 0)
cars_data.head()
```

Out[32]:

	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

Scatter Plot

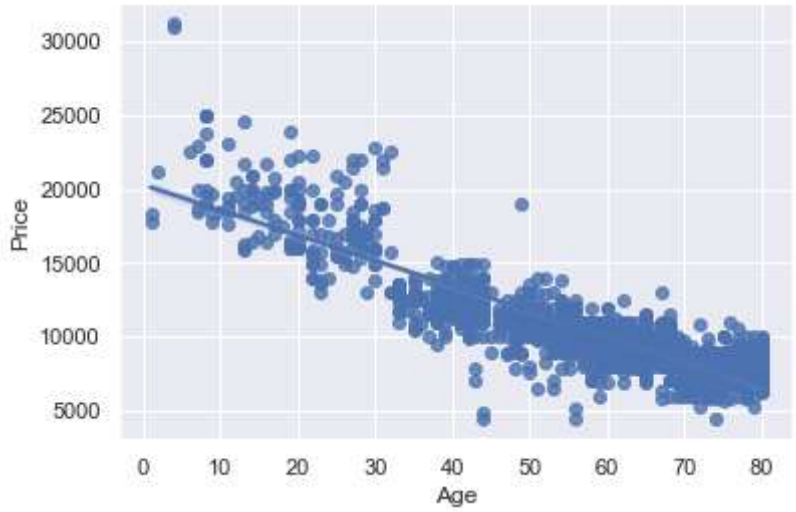
> A **scatter plot** (also called a ***scatterplot***, ***scatter graph***, ***scatter chart***, ***scattergram***, or ***scatter diagram***) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data. Wikipedia

A scatter plot can be drawn in Python by using the `seaborn.regplot()` function.

Scatter Plot with Regression line

In [33]:
```
sns.set(style = 'darkgrid')
sns.regplot(x = cars_data['Age'], y = cars_data['Price'])
```

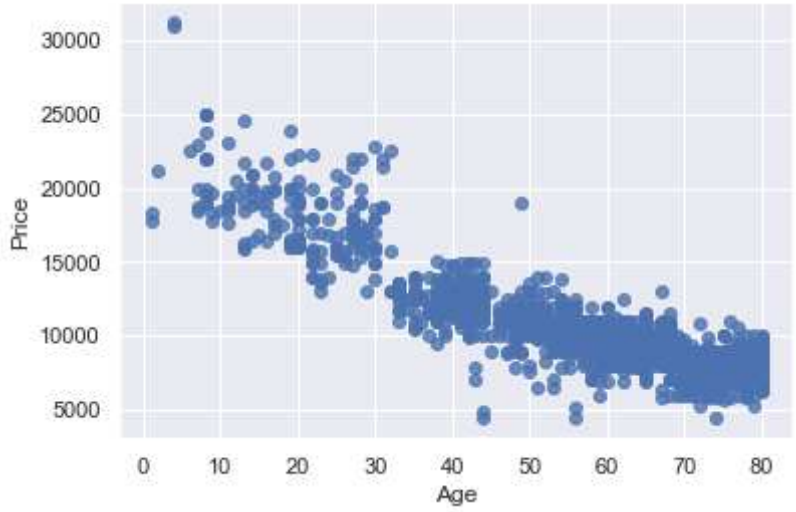
Out[33]:
```
<AxesSubplot:xlabel='Age', ylabel='Price'>
```



Scatter plot without Regression line

In [34]:
```
sns.regplot(x = cars_data['Age'], y = cars_data['Price'], fit_reg = False)
```

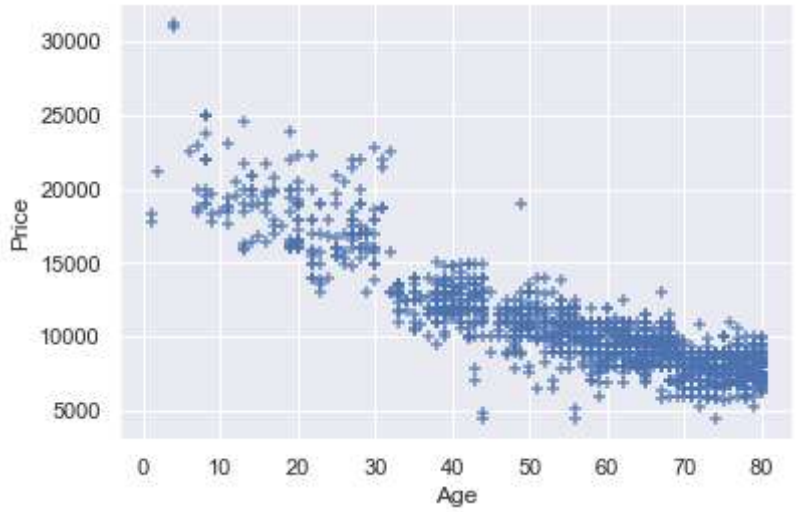
Out[34]:
```
<AxesSubplot:xlabel='Age', ylabel='Price'>
```



Scatter Plot with a different marker and without Regression line

In [35]:
```
sns.regplot(x = cars_data['Age'], y = cars_data['Price'], marker = '+', fit_reg = False
```

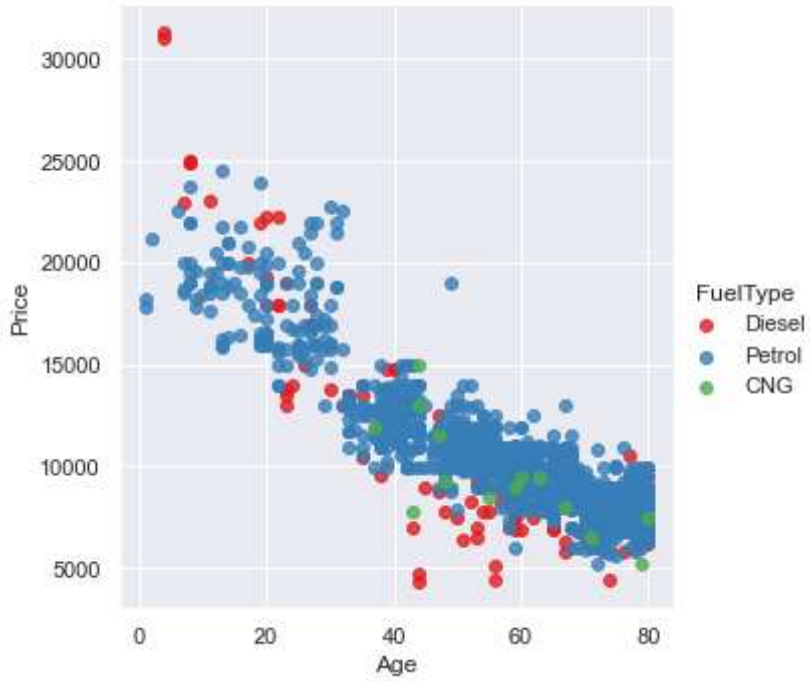
Out[35]:
```
<AxesSubplot:xlabel='Age', ylabel='Price'>
```



Plotting Age of Cars vs Prices of Cars differentiated on Fuel Type

In [36]:
```
sns.lmplot(x = 'Age', y = 'Price', data = cars_data, fit_reg = False, hue = 'FuelType',
```

Out[36]:
```
<seaborn.axisgrid.FacetGrid at 0x24c2a9c8460>
```



Histogram

> A **histogram** is an approximate representation of the distribution of numerical data.
> Wikipedia

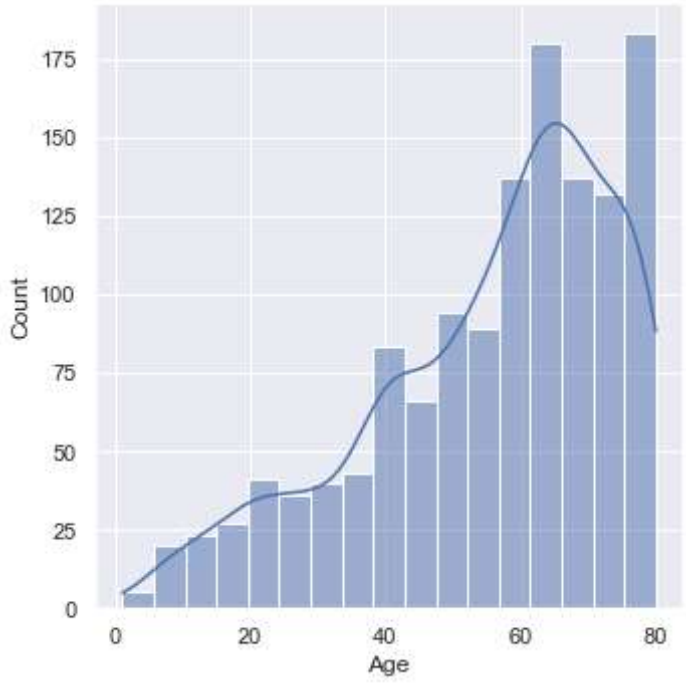
A histogram can be created in Python by using `seaborn.distplot()` (Deprecated since 0.11.2) and `seaborn.displot()` function.

Histogram with a KDE plot

Note: All examples of histogram moving forward will use the `seaborn.displot()` method instead of `seaborn.distplot()`. The only difference between the two functions pertaining to the practical is that **`displot()`** does not generate a KDE plot with the histogram. Essentially, `displot(X, *args) == distplot(X, kde = False, *args)` for the scope of this practical. Replace displot() with distplot() and invert the kde parameter for Seaborn v0.11.2 or lower. All other parameters are the same.

In [37]:
```
sns.displot(cars_data['Age'], kde = True)
```

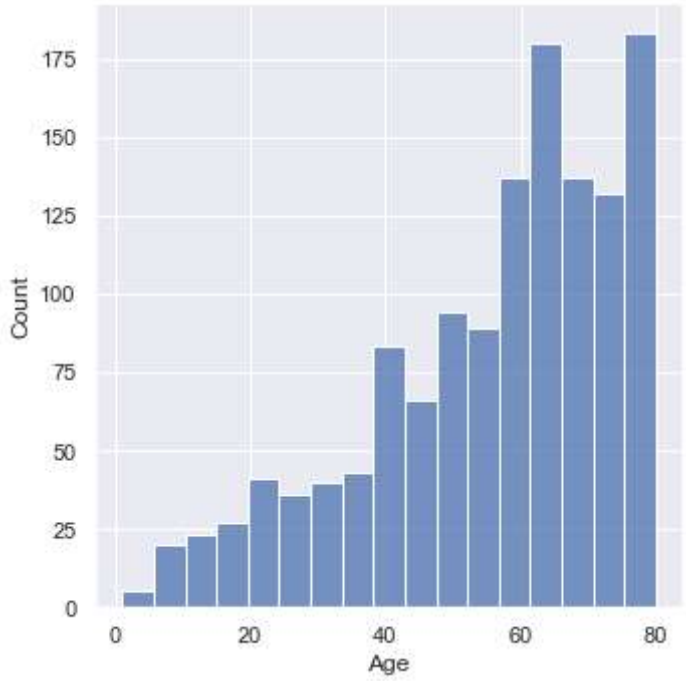
Out[37]:
```
<seaborn.axisgrid.FacetGrid at 0x24c2a909820>
```



Histogram without a KDE plot

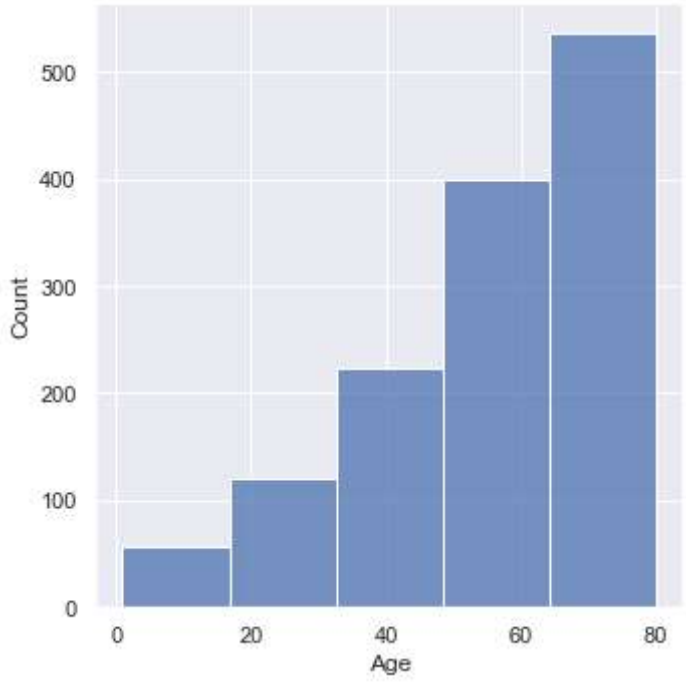
In [38]:
```
sns.displot(cars_data['Age'])
```

Out[38]:
```
<seaborn.axisgrid.FacetGrid at 0x24c2a7b02b0>
```



In [39]:
```
sns.displot(cars_data['Age'], bins = 5)
```

Out[39]:
```
<seaborn.axisgrid.FacetGrid at 0x24c2acd9550>
```



Bar Chart

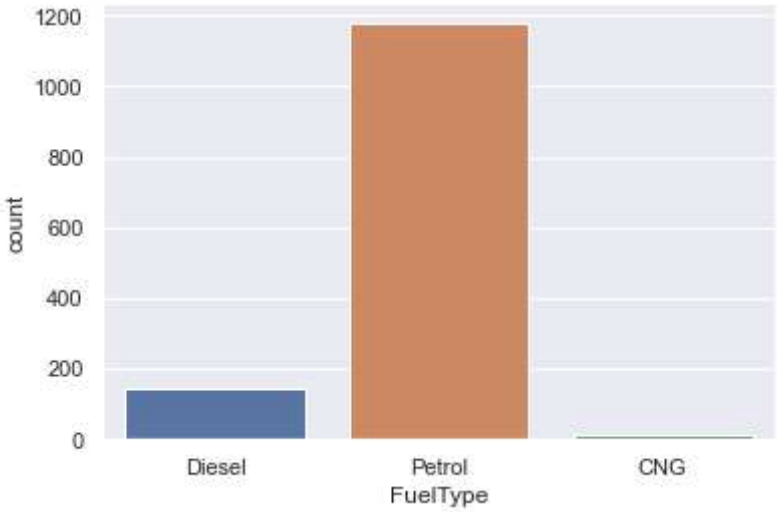
> A **bar chart** or ***bar graph*** is a chart or graph that represents categorical data with rectangular bars with heights or lengths proportional to the values they represent. These bars can be plotted vertically or horizontally. A vertical bar chart is sometimes called a ***column chart***. Wikipedia

A bar chart can be created in Python by using `seaborn.countplot()` function.

Plot between Fuel type and Number of Cars

In [40]:
```
sns.countplot(x = "FuelType", data = cars_data)
```

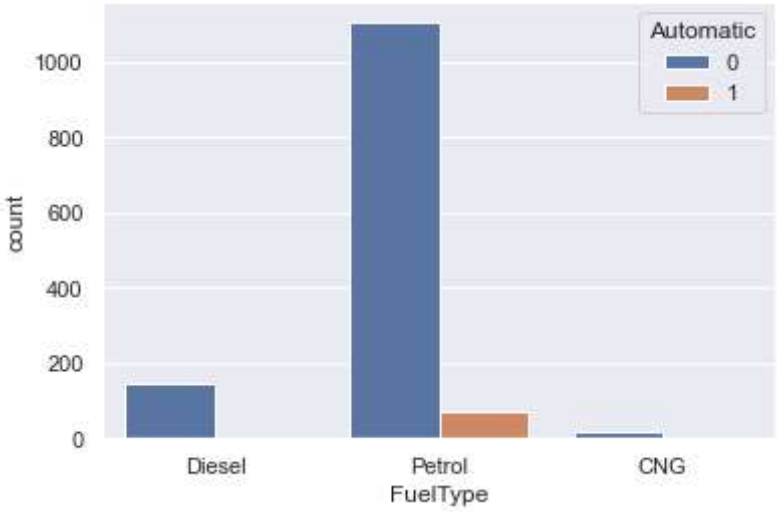
Out[40]:
```
<AxesSubplot:xlabel='FuelType', ylabel='count'>
```



Plot between the Fuel type and number of cars differentiated by transmission method (Manual or Automatic)

In [41]:
```
sns.countplot(x = "FuelType", data = cars_data, hue = "Automatic")
```

Out[41]:
```
<AxesSubplot:xlabel='FuelType', ylabel='count'>
```



Box Plot

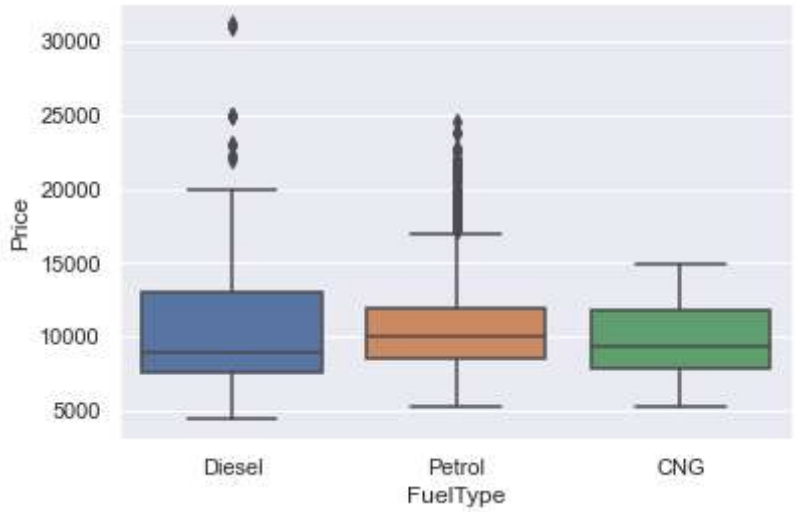
> A **box plot** or **boxplot** is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending from the boxes (*whiskers*) indicating variability outside the upper and lower quartiles, hence the term

> ***box-and-whisker plots*** and ***box-and-whisker diagram***. Outliers may be plotted as individual points. Wikipedia

A box plot can be drawn in Seaborn by using the `seaborn.boxpot()` function.

In [42]:
```
sns.boxplot(x = cars_data['FuelType'], y = cars_data['Price'])
```

Out[42]:
```
<AxesSubplot:xlabel='FuelType', ylabel='Price'>
```



Pair Plot

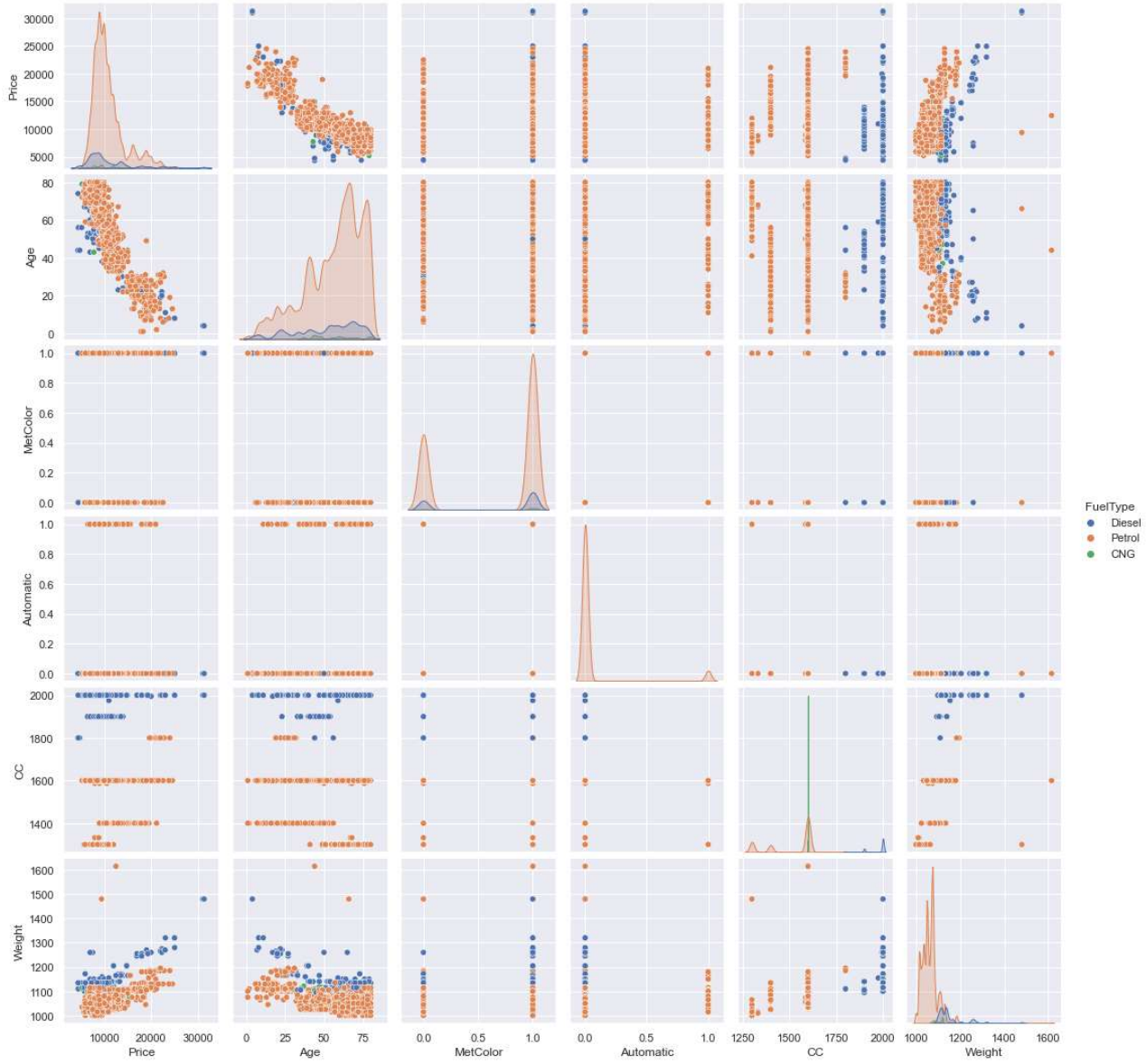
> Plot pairwise relationships in a dataset. By default, this will create a grid of Axes such that each numeric variable of dataset will be shared across the Y-axes across a single row and the X-axes across a single column. Seaborn

A pairplot can be drawn in Seaborn by using the `seaborn.pairplot()` function.

Note: The original practical uses bw instead of bw_method. But bw has been deprecated since v 0.11.2. Replace bw_method with bw for Seaborn v 0.11 or lower.

In [43]:
```
sns.pairplot(cars_data, kind = "scatter", hue = "FuelType", diag_kws = {'bw_method' : 0
plt.plot()
```

Out[43]:
```
[]
```


Price
Age
MetColor
Automatic
CC
Weight
FuelType
Diesel
Petrol
CNG