In [6]:

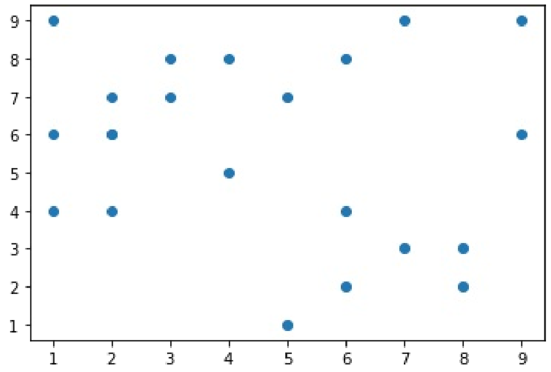
```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

In [7]:

```python
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[7]:

```
<matplotlib.collections.PathCollection at 0x1cec150bf70>
```



In [8]:

```python
iris = pd.read_csv(r'C:\Users\acer\Desktop\Sem 1\data science\DataSet\iris.csv', names=['sepal_length', 'sepal_wi
print(iris.head())
```

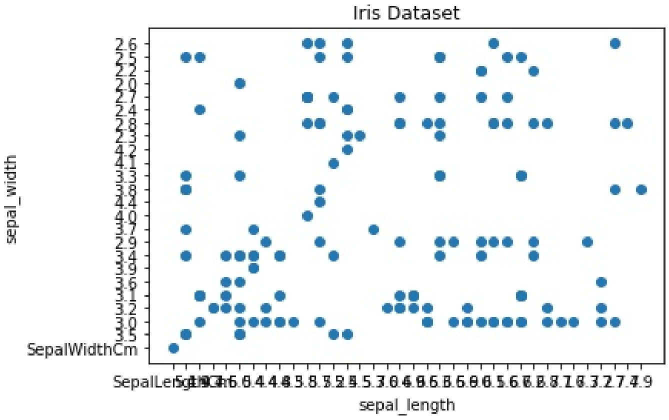
```
     sepal_length   sepal_width   petal_length   petal_width          class
Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm        Species
1             5.1           3.5            1.4           0.2    Iris-setosa
2             4.9           3.0            1.4           0.2    Iris-setosa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

In [9]:

```python
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[9]:

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



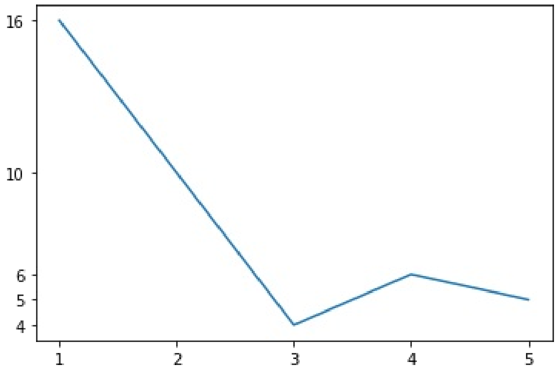
Line Chart

In [10]:

```python
x=range(1,6)
y=np.random.randint(1,20,5)
plt.plot(x,y)
plt.xticks(x)
plt.yticks(y)
```

Out[10]:

```
([<matplotlib.axis.YTick at 0x1aa72ccdc10>,
  <matplotlib.axis.YTick at 0x1aa72ccd820>,
  <matplotlib.axis.YTick at 0x1aa72ccb670>,
  <matplotlib.axis.YTick at 0x1aa72cf8ee0>,
  <matplotlib.axis.YTick at 0x1aa72d00430>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```

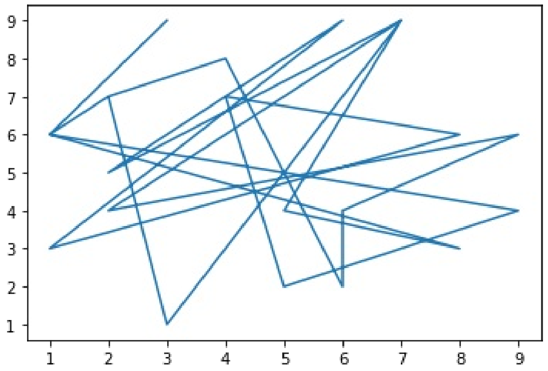


In [12]:

```
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[12]:

```
[<matplotlib.lines.Line2D at 0x1aa72d37c70>]
```

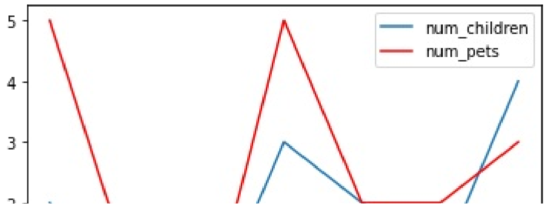


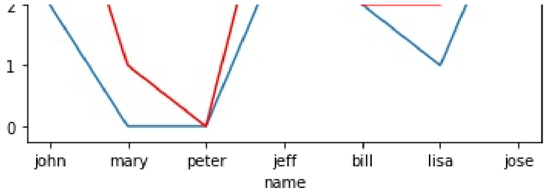
In [13]:

```
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[13]:

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





In [10]:

```
iris = pd.read_csv(r'C:\Users\acer\Desktop\Sem 1\data science\DataSet\iris.csv', names=['sepal_length', 'sepal_wi
print(iris.head())
```

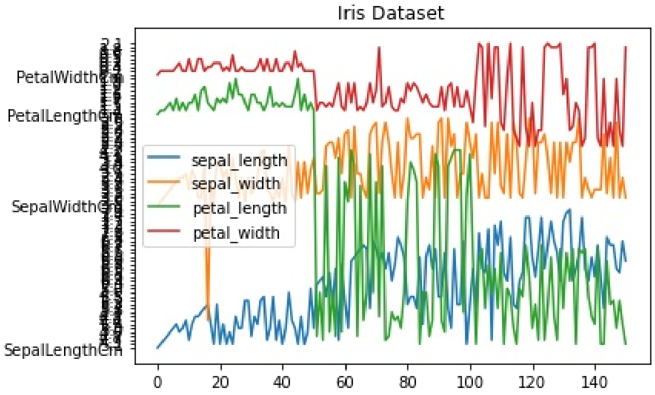
```
     sepal_length   sepal_width  petal_length   petal_width        class
Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
1             5.1           3.5            1.4           0.2  Iris-setosa
2             4.9           3.0            1.4           0.2  Iris-setosa
3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

In [16]:

```
# 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[16]:

```
<matplotlib.legend.Legend at 0x1aa72e2b340>
```



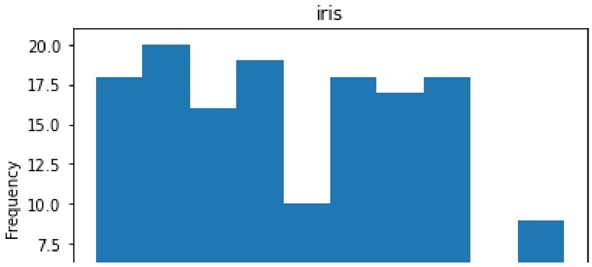
Histogram

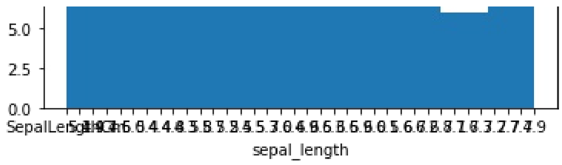
In [17]:

```
# 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[17]:

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





Bar Chart

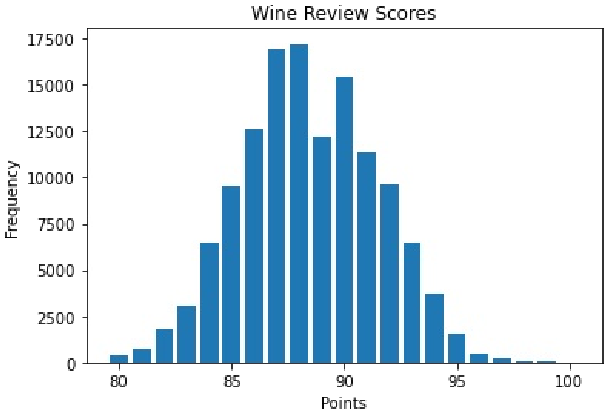
In [11]:
```
wine_reviews = pd.read_csv(r'C:\Users\acer\Desktop\Sem 1\data science\DataSet\winemag-data-130k-v2.csv', index_co
wine_reviews.head()
```

Out[11]:

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety
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
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red
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
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
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

In [23]:
```
#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')
```

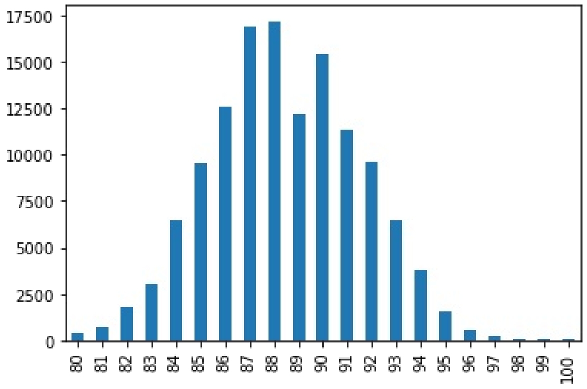
Out[23]:
```
Text(0, 0.5, 'Frequency')
```



In [24]:

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

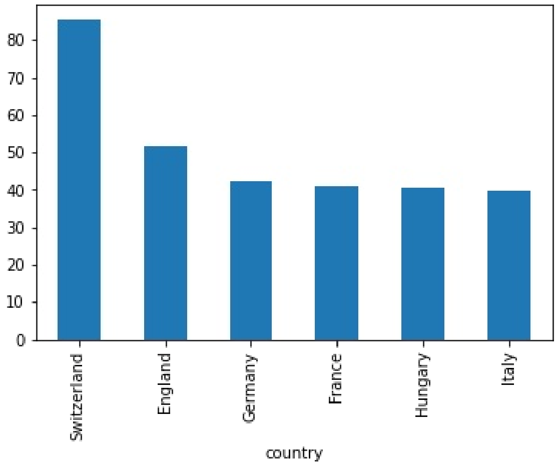
Out[24]: <AxesSubplot:>



In [25]:

```
wine_reviews.groupby("country").price.mean().sort_values(ascending=False)[:6].plot.bar()
```

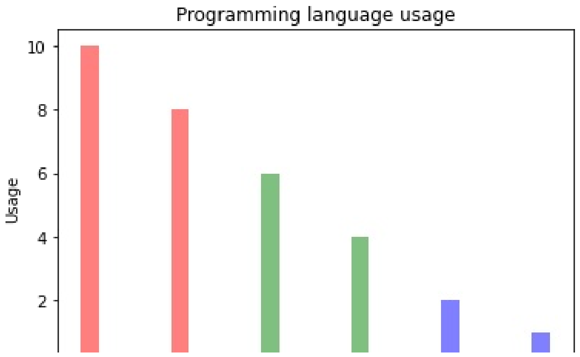
Out[25]: <AxesSubplot:xlabel='country'>



In [26]:

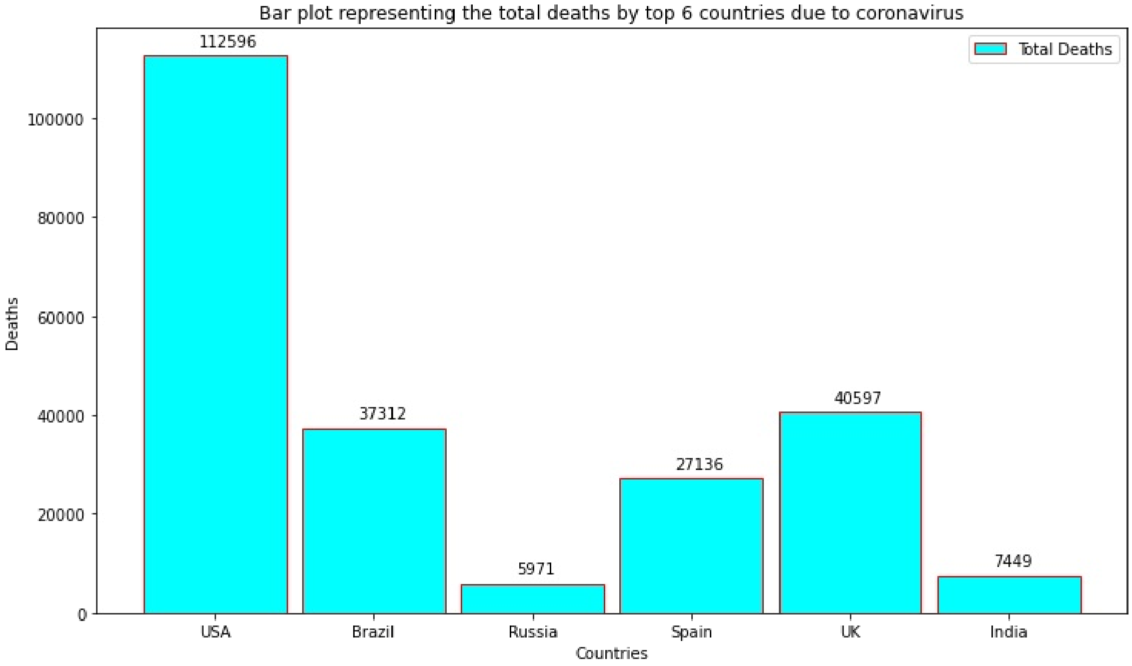
```
objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]
# Bar Chart
# X Axis positions as first parameter list, it can be floating point numbers also
# Y Values as 2nd parameter list
# Alpha is transparency,
# Align can be center or edge
# Color can be single value or a list of color codes, one for each bar.
plt.bar(y_pos, performance, width=0.2, align='edge', 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[26]: Text(0.5, 1.0, 'Programming language usage')



In [27]:

```
# 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')
```

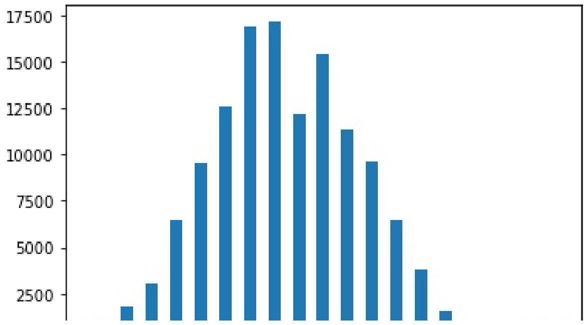


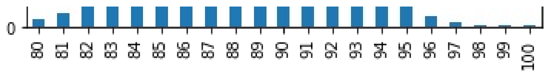
In [28]:

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

Out[28]:

```
<AxesSubplot:>
```



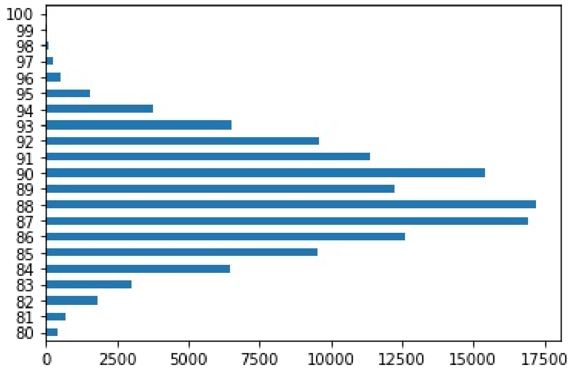


In [30]:

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

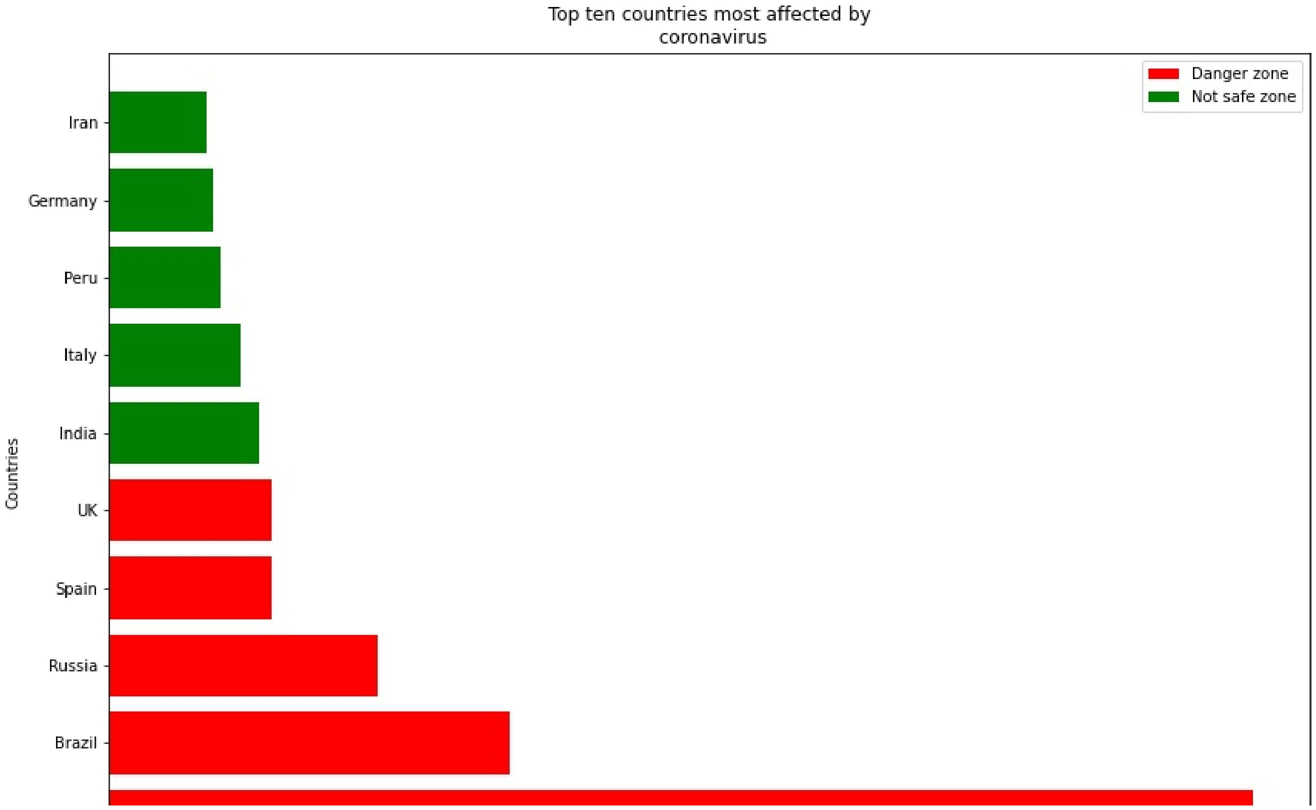
Out[30]:

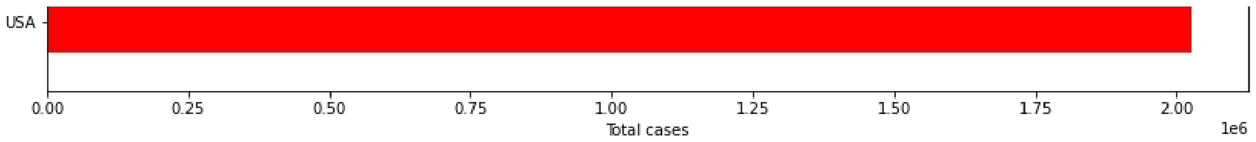
```
<AxesSubplot:>
```



In [31]:

```
# 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 z
plt.barh(['India', 'Italy', 'Peru', 'Germany', 'Iran'], [265928, 235278, 199696, 186205, 173832], label = "Not sa

# 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()
```



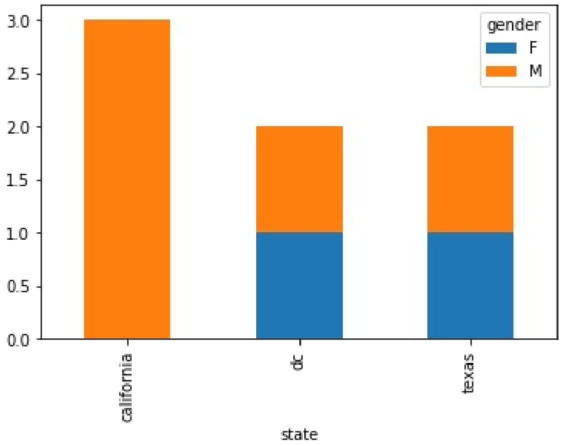


In [32]:

```
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)
```

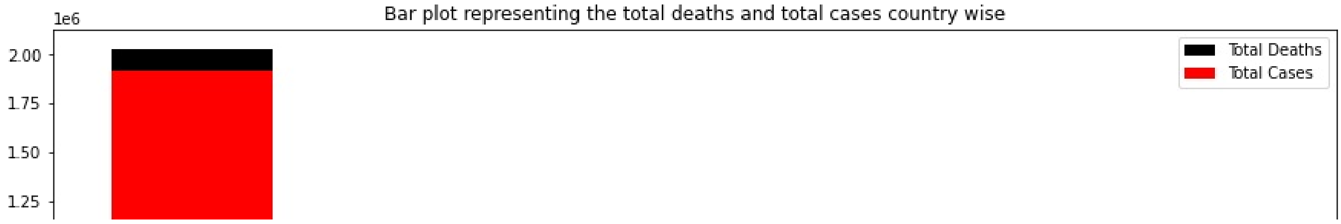
Out[32]:

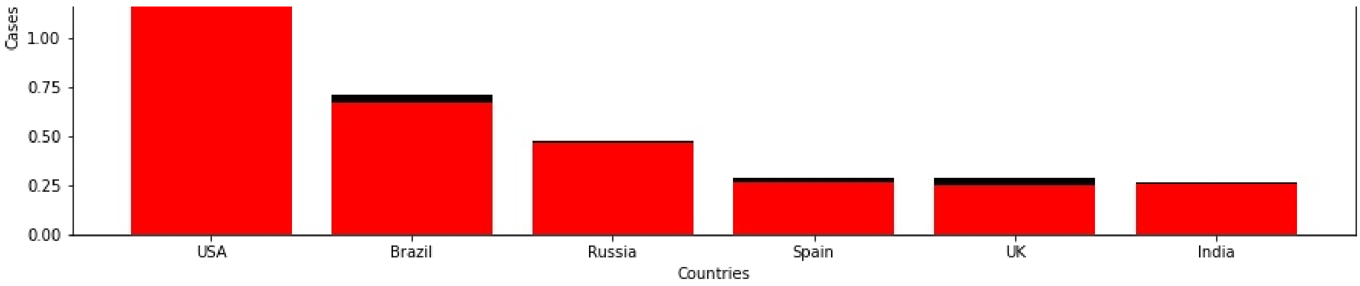
```
<AxesSubplot:xlabel='state'>
```



In [33]:

```
# 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()
```



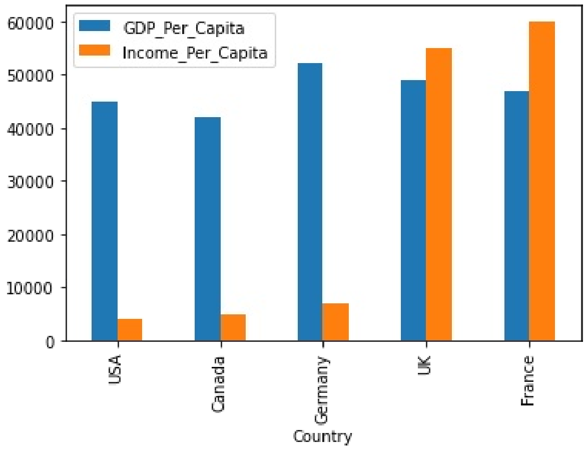


In [34]:

```
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]:

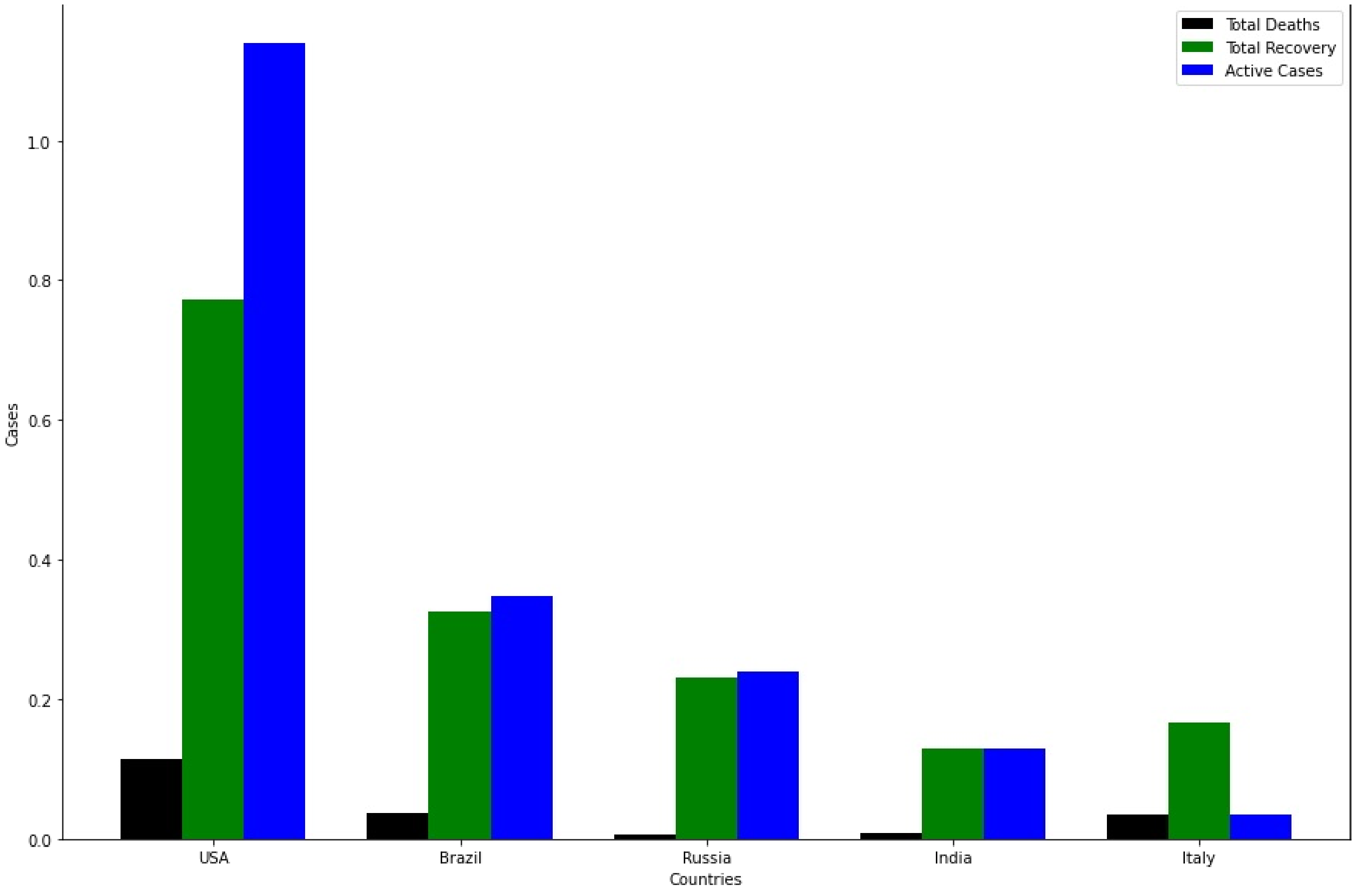
```
<AxesSubplot:xlabel='Country'>
```



In [35]:

```
# 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()
```

Bar plot representing the total deaths, total recovered cases and active cases country wise



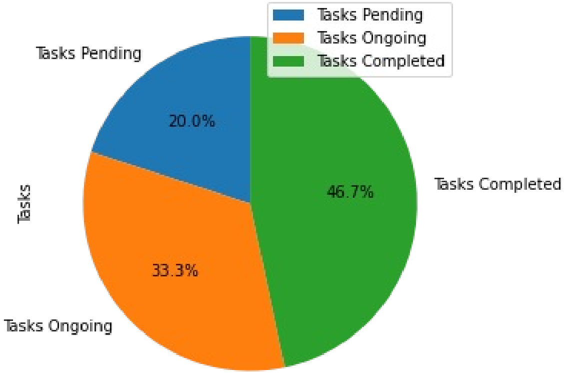
Pie chart

In [37]:

```
# 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=(5,5),autopct='%1.1f%%', startangle=90)
```

Out[37]:

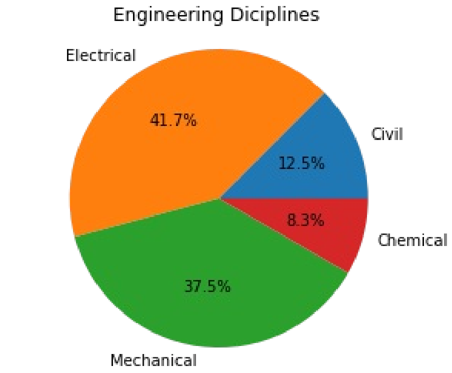
```
<AxesSubplot:ylabel='Tasks'>
```



In [38]:

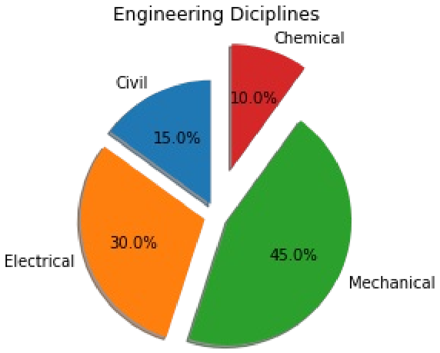
```
%matplotlib inline


# 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()
```



In [39]:

```
# 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()
```

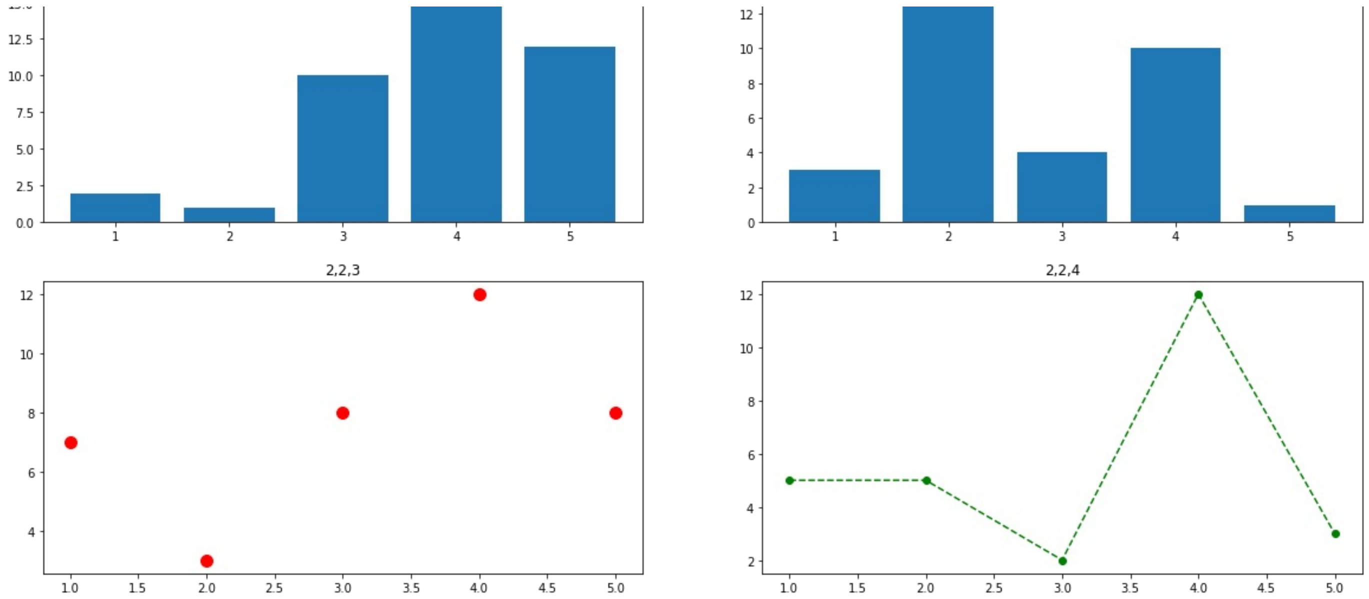


In [40]:

```
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[40]:

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

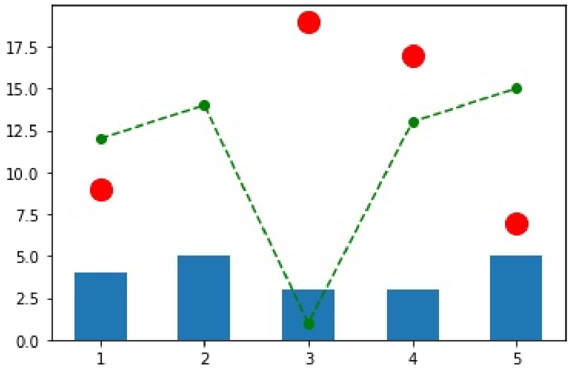


In [41]:

```
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[41]:

```
[<matplotlib.lines.Line2D at 0x1aa740da160>]
```



Seaborn

In [12]:

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

In [14]:

```
os.chdir(r'C:\Users\acer\Desktop\Sem 1\data science\DataSet')
cars_data=pd.read_csv('Toyota.csv',index_col=0,na_values=["??","????"])
cars_data.size
```

Out[14]:

```
14360
```

In [47]:

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

Out[47]:

```
10960
```

In [48]:

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

Out[48]:

Out[48]:

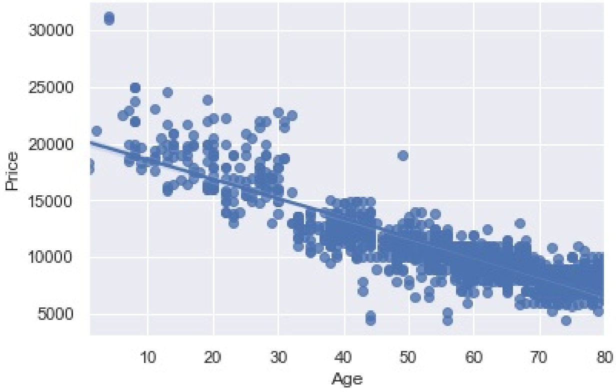
	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170

In [49]:

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

Out[49]:

```
<AxesSubplot:xlabel='Age', ylabel='Price'>
```

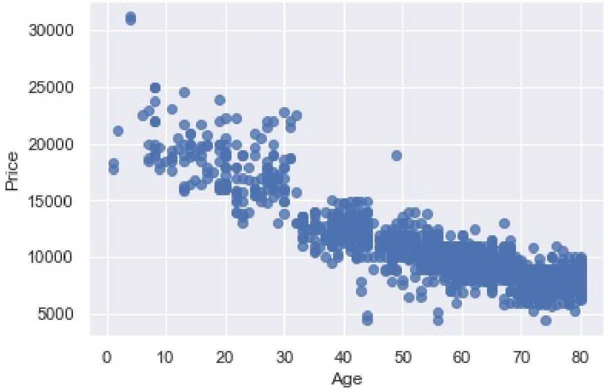


In [50]:

```
#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[50]:

```
<AxesSubplot:xlabel='Age', ylabel='Price'>
```

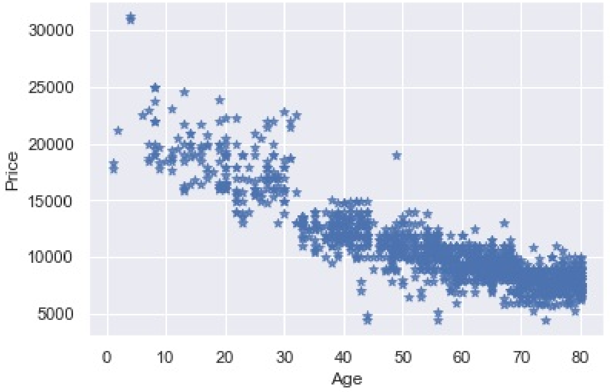


In [51]:

```
sns.regplot(x=cars_data['Age'], y=cars_data['Price'], marker="*", fit_reg=False)
```

Out[51]:

```
<AxesSubplot:xlabel='Age', ylabel='Price'>
```

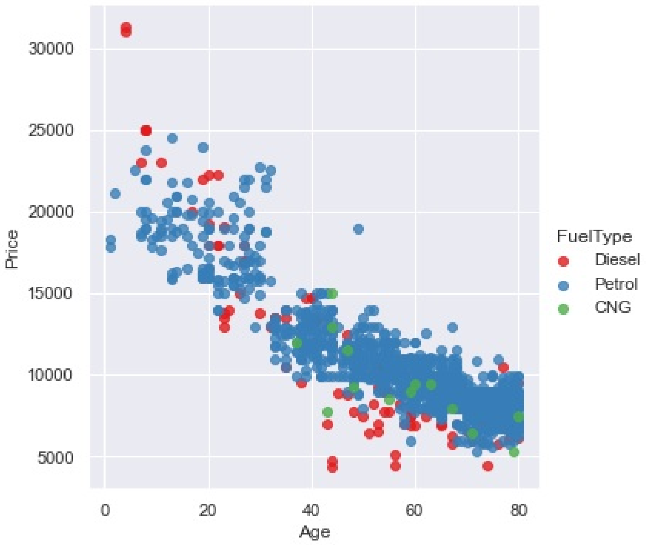


In [52]:

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

Out[52]:

```
<seaborn.axisgrid.FacetGrid at 0x1aa74518b80>
```



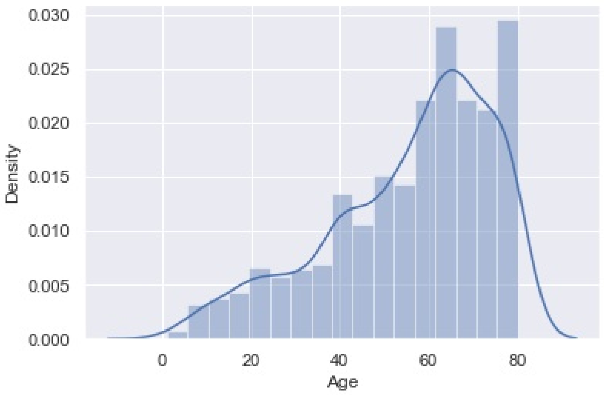
In [53]:

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

```
C:\Users\acer\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecate
d function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[53]:

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



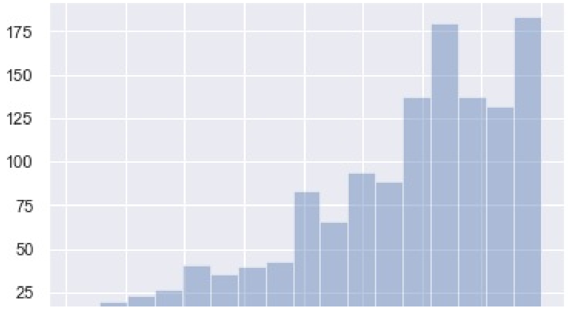
In [54]:

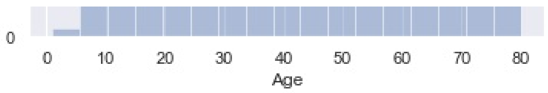
```python
sns.distplot(cars_data['Age'],kde=False)
```

```
C:\Users\acer\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecate
d function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[54]:

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





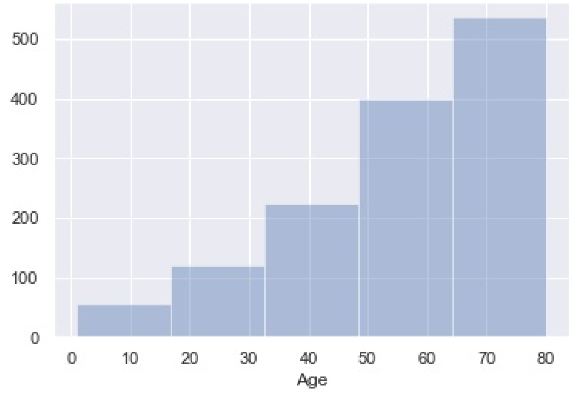
In [60]:

```
sns.distplot(cars_data['Age'],kde=False, bins=5)
```

```
C:\Users\acer\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecate
d function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[60]:

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

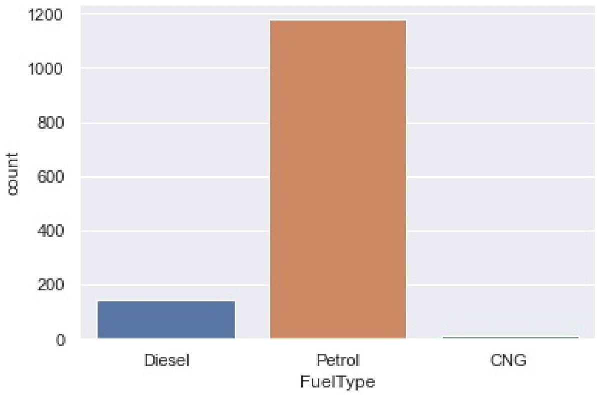


In [61]:

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

Out[61]:

```
<AxesSubplot:xlabel='FuelType', ylabel='count'>
```

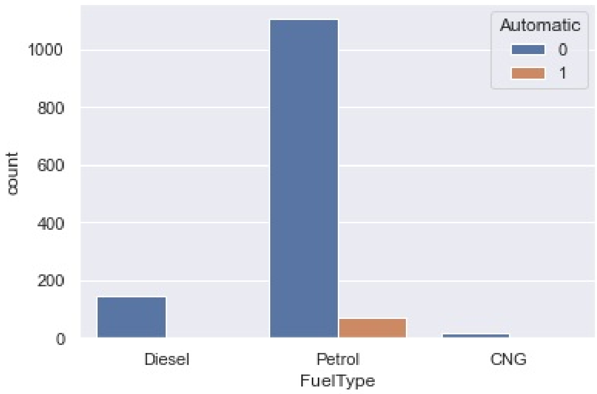


In [62]:

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

Out[62]:

```
<AxesSubplot:xlabel='FuelType', ylabel='count'>
```

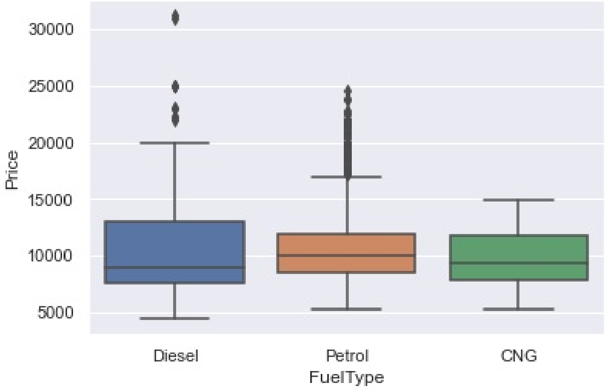


In [64]:

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

Out[64]:

```
<AxesSubplot:xlabel='FuelType', ylabel='Price'>
```



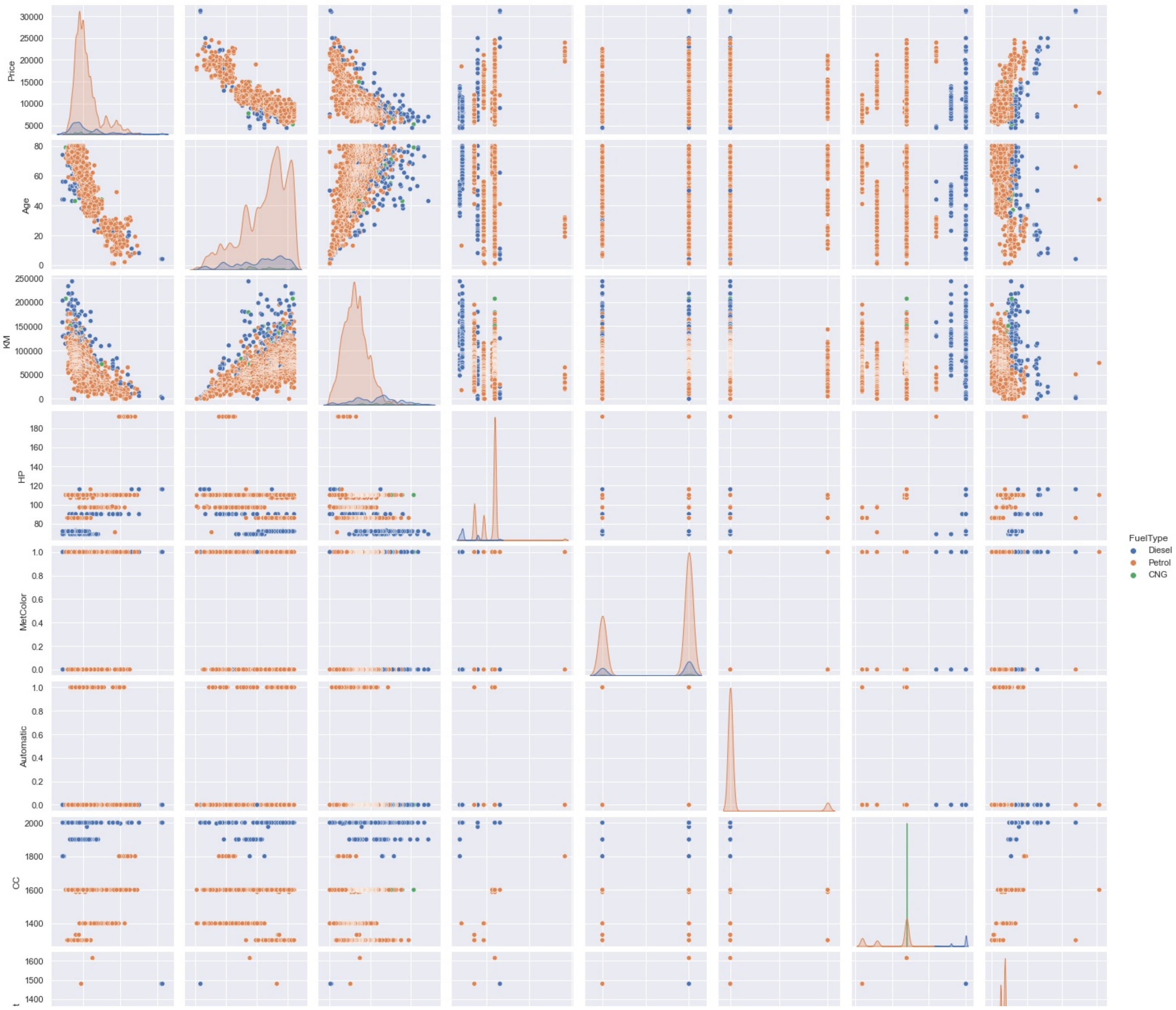
In [66]:

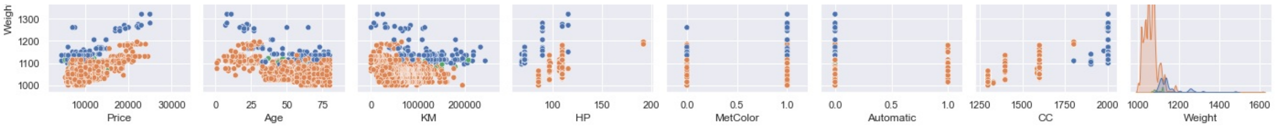
```
sns.pairplot(cars_data, kind = "scatter", hue = "FuelType", diag_kws = {'bw_method' : 0.1})
plt.plot()
```

```
C:\Users\acer\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWarning: Dataset has 0 variance; skip
ping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\acer\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWarning: Dataset has 0 variance; skip
ping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\acer\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWarning: Dataset has 0 variance; skip
ping density estimate.
  warnings.warn(msg, UserWarning)
```

Out[66]:

```
[]
```





In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js