# Learning Seaborn

- dist plot
- pair plot : whenever dimensionality falls in the range of 2<dim<7
- joint plot : For Bivariate Analysis

In [1]:
```
import seaborn as sns
import pandas as pd
import numpy as np
```

In [3]:
```
df= pd.read_csv('iris.csv')
df.head()
```

Out[3]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

# Heatmaps using Correlations

- used in corelating features during feature selection
- shows a 2D Correlation Matrix
- calculated for numerical features

In [4]:
```
df.corr()
```

Out[4]:

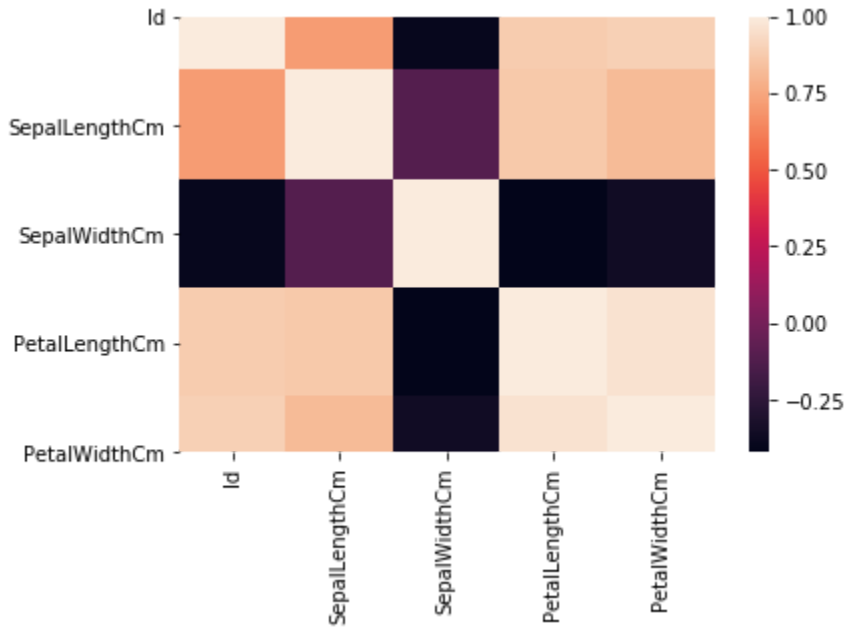
| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| Id | 1.000000 | 0.716676 | -0.397729 | 0.882747 | 0.899759 |
| SepalLengthCm | 0.716676 | 1.000000 | -0.109369 | 0.871754 | 0.817954 |
| SepalWidthCm | -0.397729 | -0.109369 | 1.000000 | -0.420516 | -0.356544 |
| PetalLengthCm | 0.882747 | 0.871754 | -0.420516 | 1.000000 | 0.962757 |
| PetalWidthCm | 0.899759 | 0.817954 | -0.356544 | 0.962757 | 1.000000 |

In [8]:
```
df.dtypes
```

Out[8]:
```
Id                int64
SepalLengthCm     float64
SepalWidthCm      float64
PetalLengthCm     float64
PetalWidthCm      float64
Species           object
dtype: object
```

In [10]:
```
sns.heatmap(df.corr())
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8800843108>
```




In [15]:
```
df1=df.drop(['Id'], axis=1)
df1.head()
```

Out[15]:

| | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

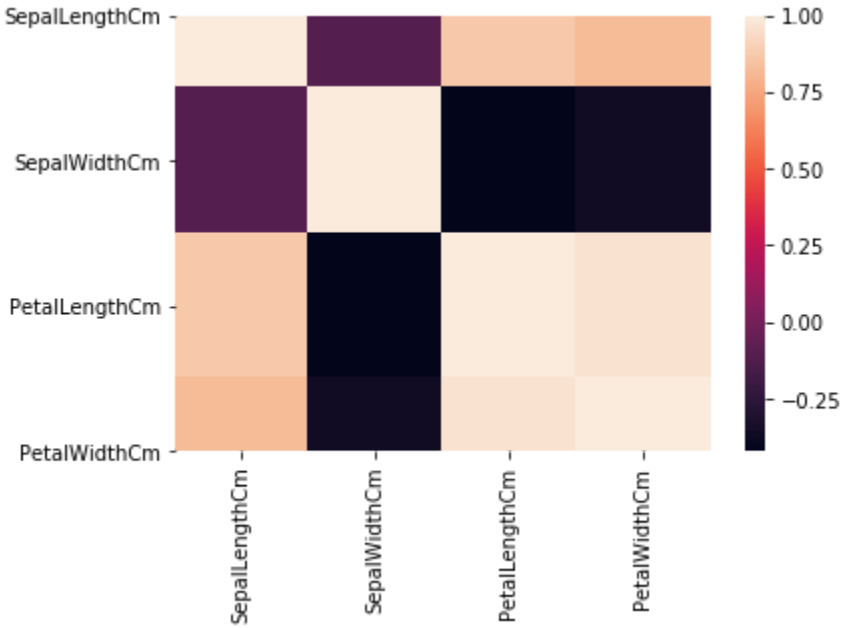
In [19]:
```
df1.corr()
```

Out[19]:

| | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|
| SepalLengthCm | 1.000000 | -0.109369 | 0.871754 | 0.817954 |
| SepalWidthCm | -0.109369 | 1.000000 | -0.420516 | -0.356544 |
| PetalLengthCm | 0.871754 | -0.420516 | 1.000000 | 0.962757 |
| PetalWidthCm | 0.817954 | -0.356544 | 0.962757 | 1.000000 |

In [18]:
```
sns.heatmap(df1.corr())
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8802ae1448>
```

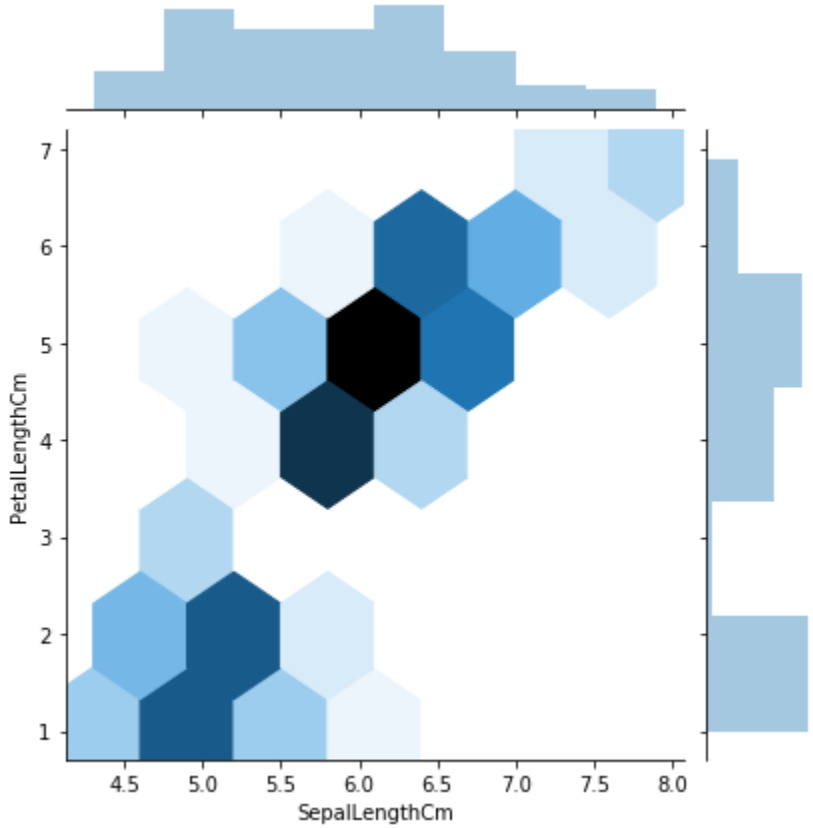



# joint Plots

- we plot one numeric feature against the other to visualise their span and relation.
- usually done between a single feature and the label.
- we also get to see the histograms.
- on using kind as 'reg' we get the pdf as well as the best fit lin regression line.

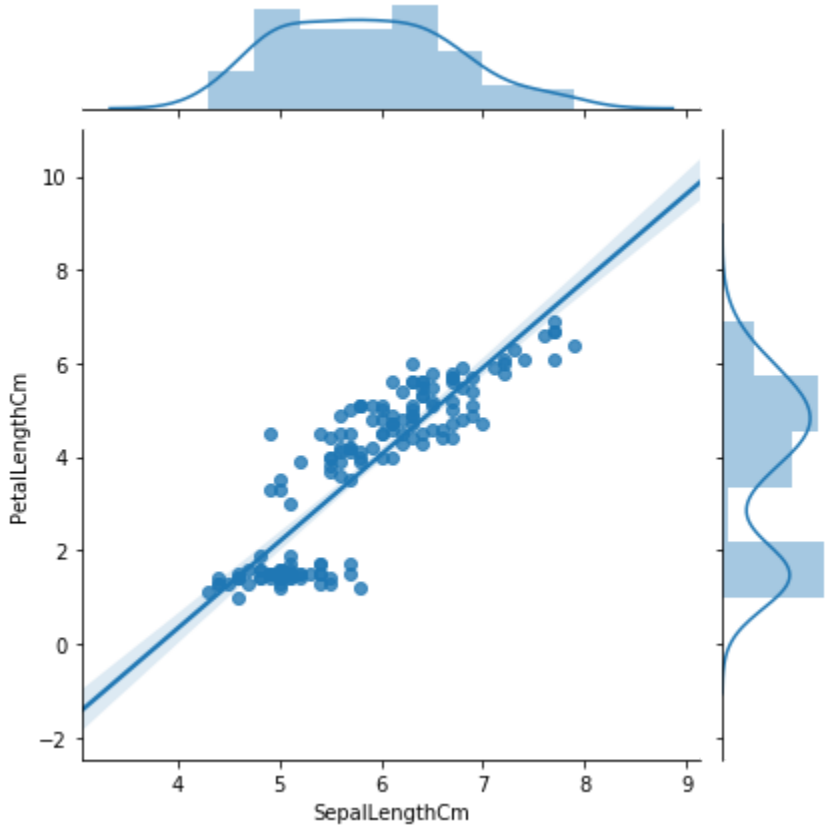
In [25]:
```
sns.jointplot(x='SepalLengthCm', y='PetalLengthCm', data=df, kind='hex')
```

Out[25]: <seaborn.axisgrid.JointGrid at 0x8802e33d48>




In [24]:
```
sns.jointplot(x='SepalLengthCm', y='PetalLengthCm', data=df, kind='reg')
```

Out[24]: <seaborn.axisgrid.JointGrid at 0x8803e64748>

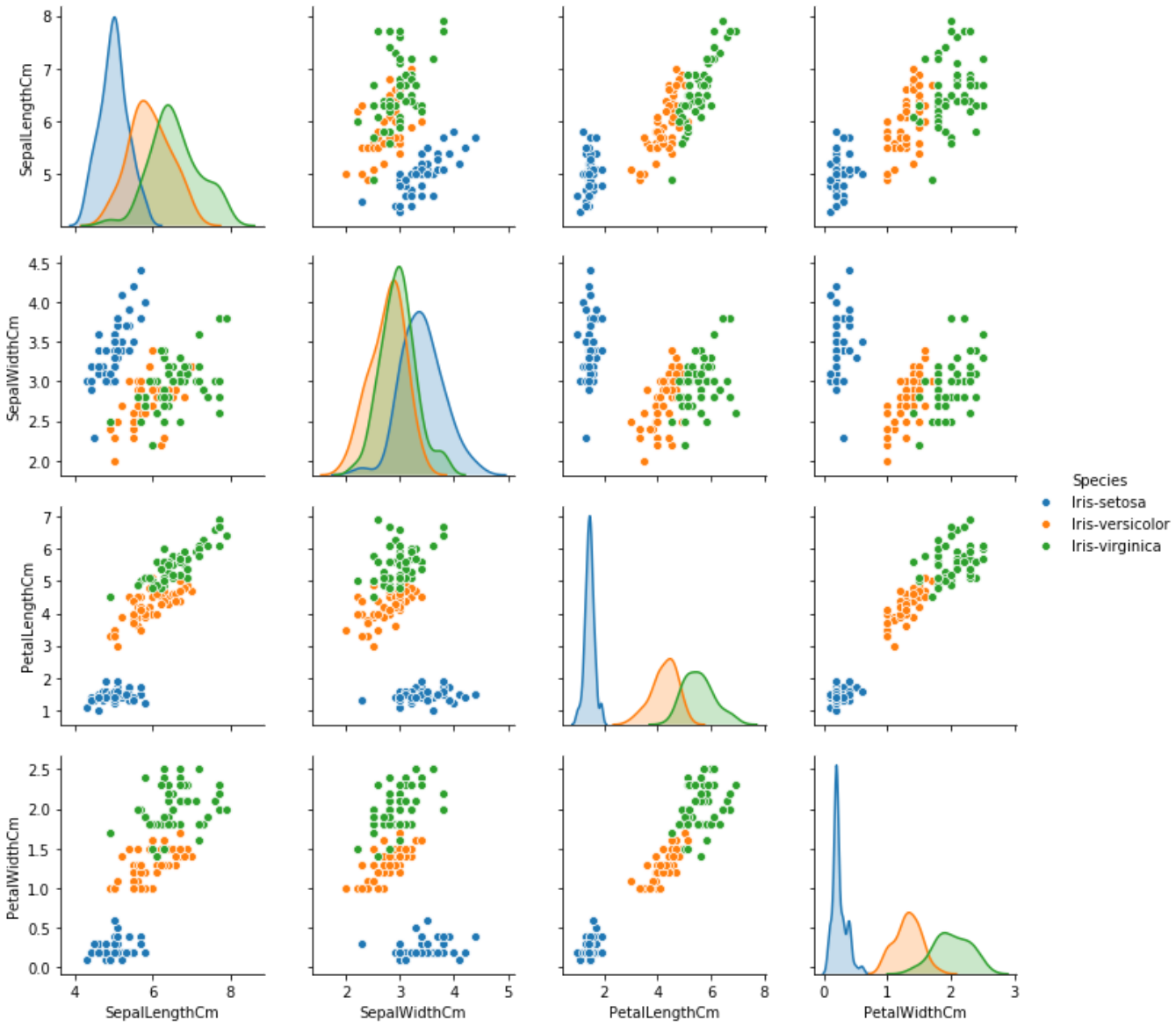


# Pair Plots

- are used if we need to visualise our features in the form of pairs, each plotted against the other in all possible combinations.

```
In [28]: sns.pairplot(df1, hue='Species')
```

```
Out[28]: <seaborn.axisgrid.PairGrid at 0x88078d96c8>
```

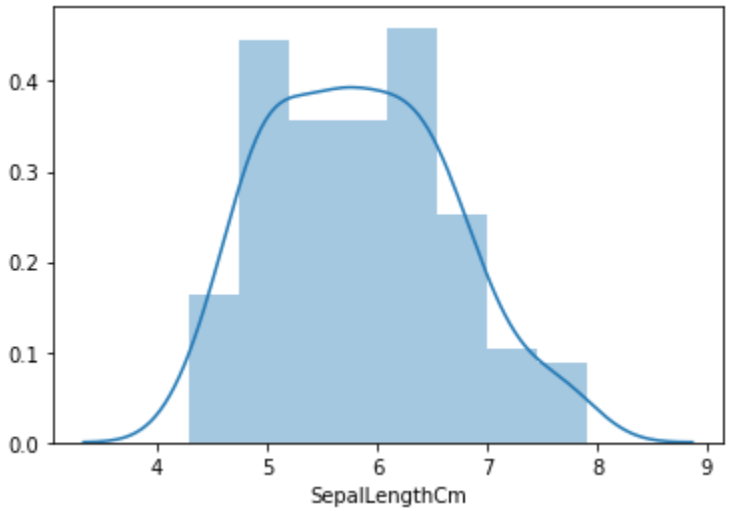



# Dist Plots

- by default shows pdf function
- if we require exact frequencies/counts we can use kde= False

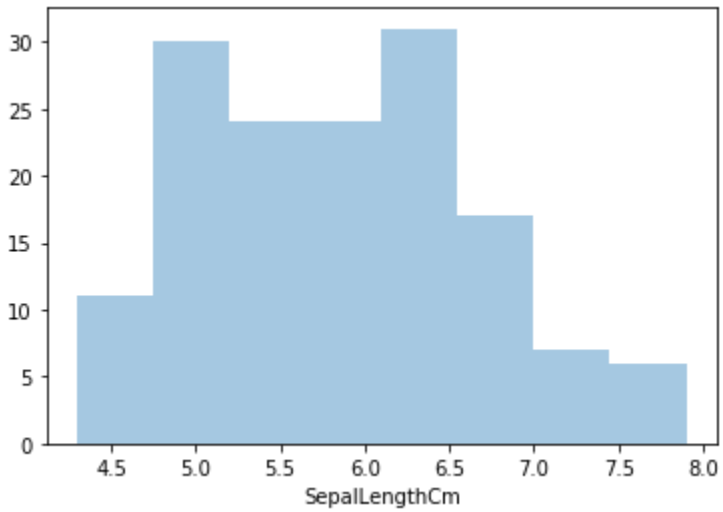
In [30]:
```
sns.distplot(df1['SepalLengthCm'])
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8808538688>
```




In [33]:
```
sns.distplot(df1.SepalLengthCm, kde=False)
```

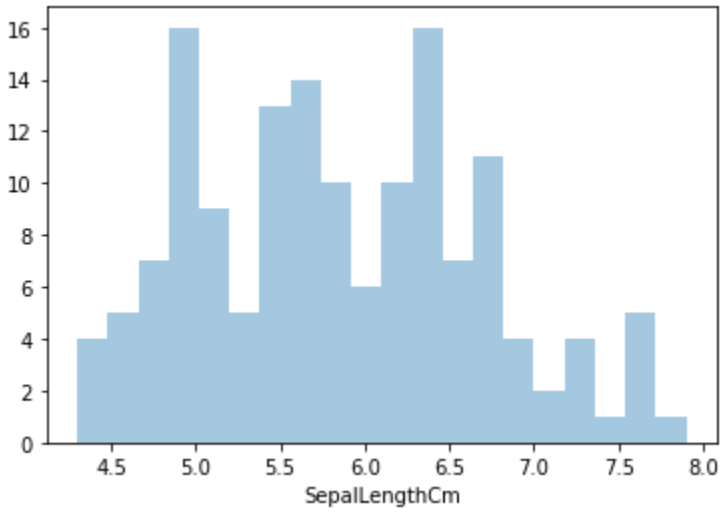
Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x88086c05c8>
```



In [35]:
```
# increasing number of bins

sns.distplot(df1.SepalLengthCm, kde=False, bins= 20)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8809a9f288>
```



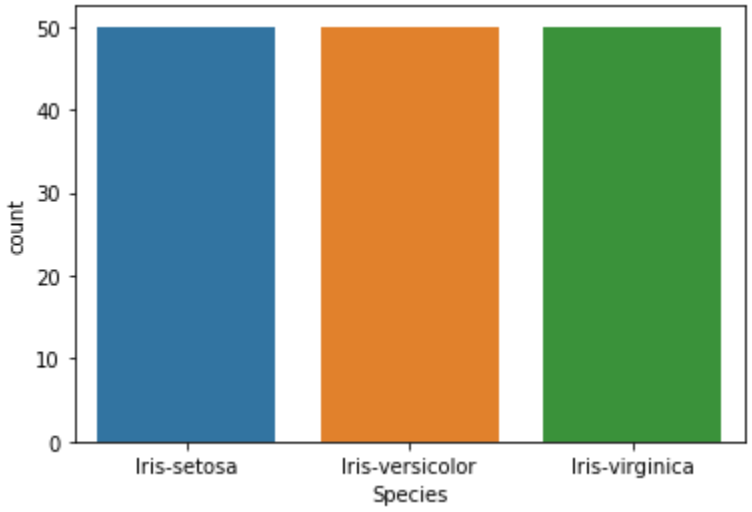

# SNS plots for categorical feature manipulation

## CountPlots

- need to specify only one feature whose values frequency has to be counted.
- will always have count value plotted on thee other axis

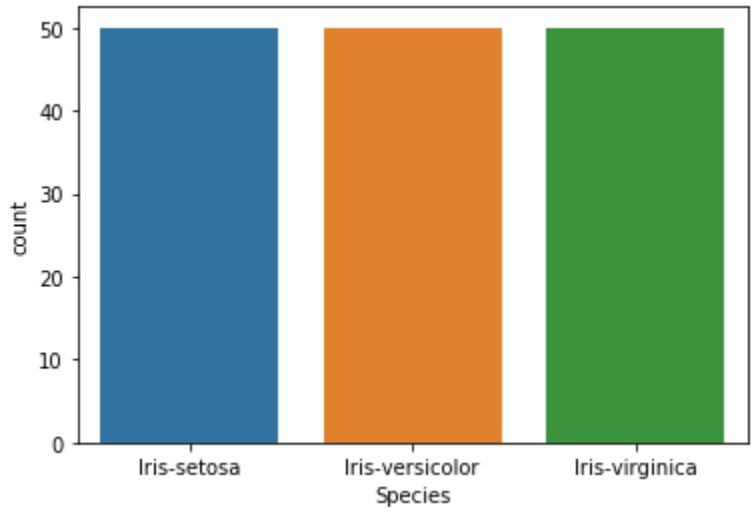
In [36]:
```
sns.countplot(df.Species)
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8808567a08>
```



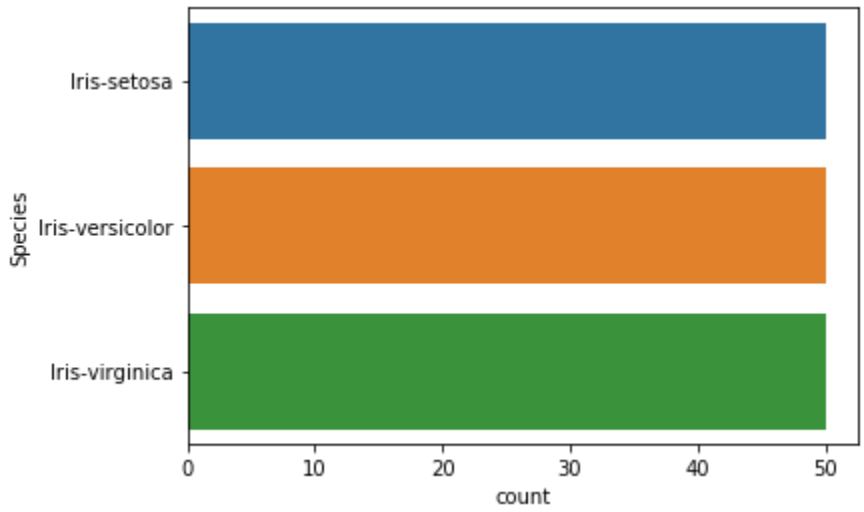
In [37]:
```
sns.countplot('Species', data=df)
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8803ee8d08>
```




In [40]:
```
sns.countplot(y=df.Species)
```

Out[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8809bfba88>
```



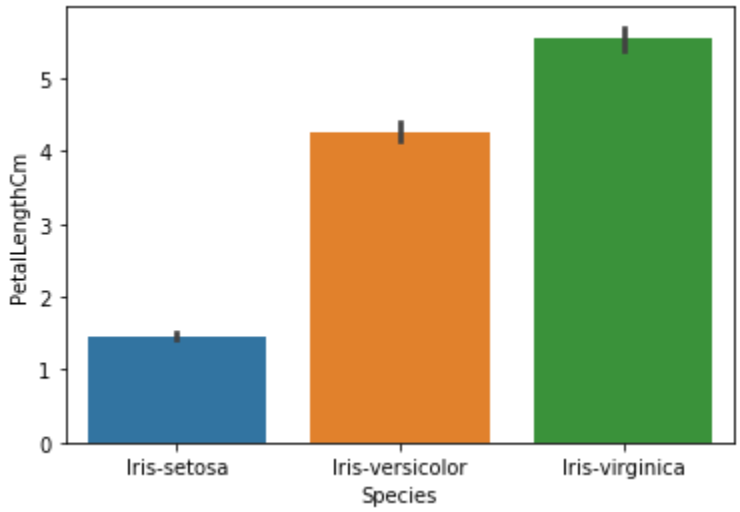

# Barplot

- need to declare which column will be plotted on which axis
- we always specify a categorical column on one axis & a numerical column on the other.

In [45]:
```
#count plot using multiple features

sns.barplot(y='PetalLengthCm',x='Species', data=df)
```

Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8809ef4e48>
```

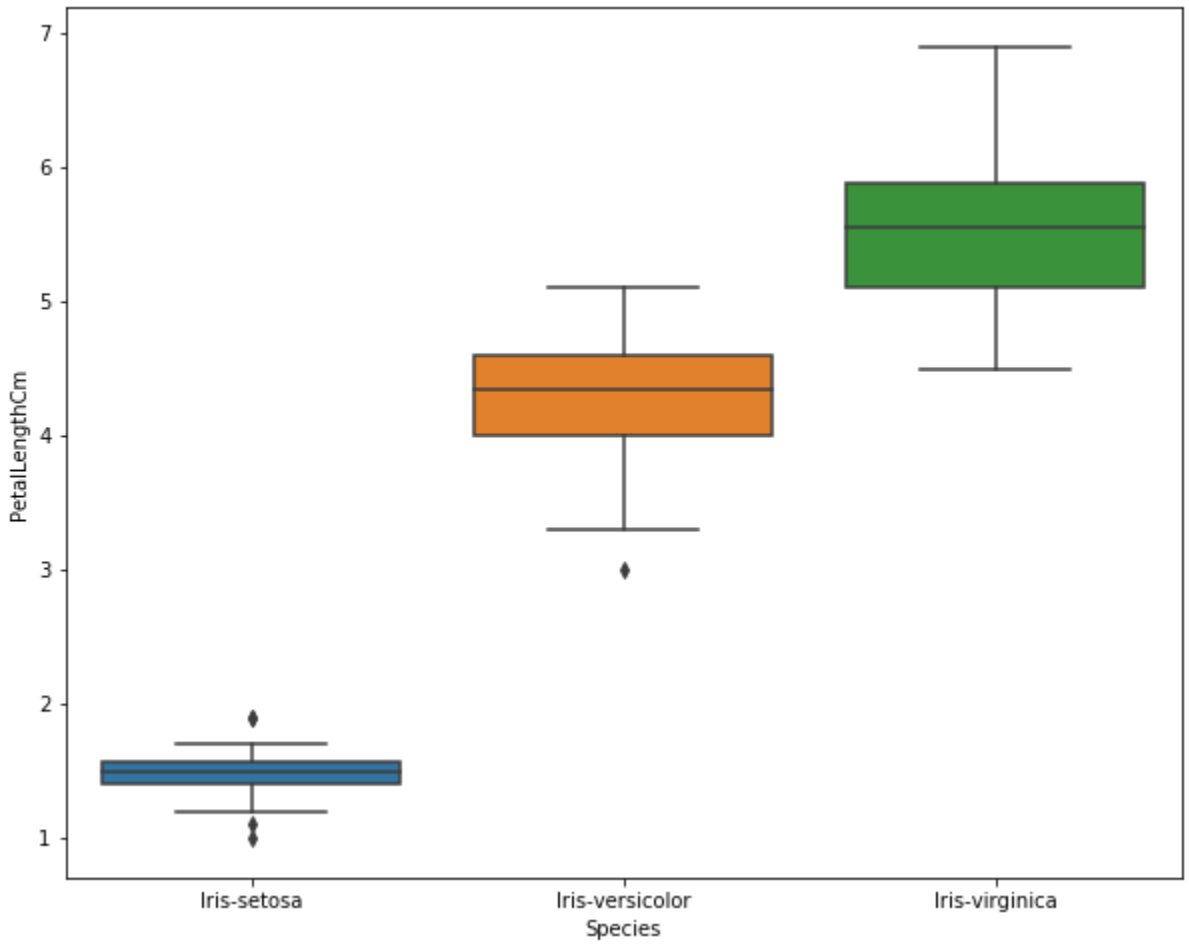



# Box & Whisker Plots

- for summarising percentiles
- finding out meadian
- visualising outliers, if any

In [50]:
```
plt.figure(figsize=(10,8)     )
sns.boxplot(df.Species, df.PetalLengthCm)
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x8809f501c8>

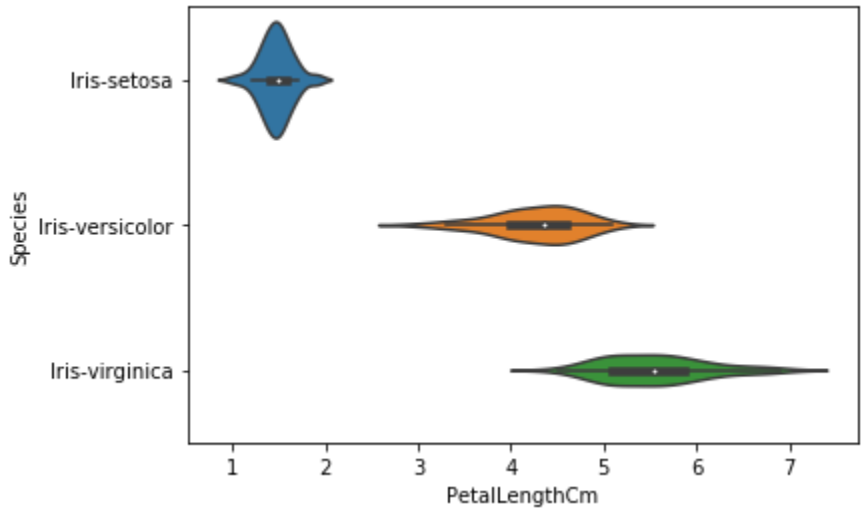



# Violin Plot

- shows the data distribution
- shows the pdf function
- shows the box plot along with the pdf

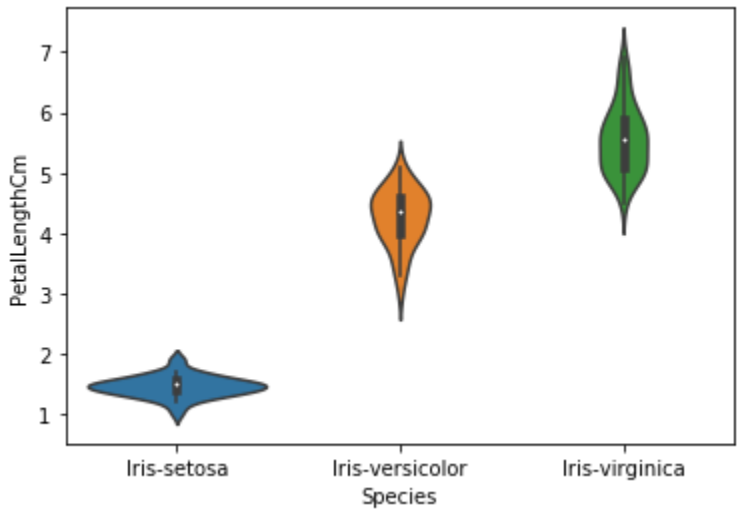
In [51]:
```
sns.violinplot(x='PetalLengthCm', y='Species', data=df)
```

Out[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x880a2a6fc8>
```




In [54]:
```
sns.violinplot(y='PetalLengthCm', x='Species', data=df)
```

Out[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x880a168dc8>
```




/