In [1]:
```
#Name : Rajshri Kirandas Satpute
#Year : 3rd year
#Section : B
#Roll No : 55
#Date : 08/02/2024
```

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
import os
```

In [4]:
```
os.chdir("C:\\Users\\fatin\\OneDrive\\Desktop")
```

In [5]:
```
df = pd.read_csv("Iris.csv")
```

In [6]:
```
df.head()
```

Out[6]:

	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

In [7]:
```
df.describe()
```

Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [9]:
```
df.shape
```

Out[9]:
```
(150, 6)
```

In [10]:
```
df.drop("Id",axis=1,inplace=True)    #droping  id becuase it is no use to us , Inplace = True means changes will take effect in original dataframe
df.head()
```

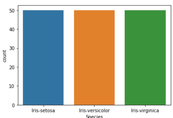
Out[10]:

	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

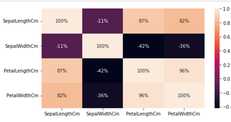
Data Visualization

In [11]:
```
print(df["Species"].value_counts())
sns.countplot(df["Species"])
```

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
C:\Users\fatin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```
Out[11]:
```
<AxesSubplot:xlabel='Species', ylabel='count'>
```

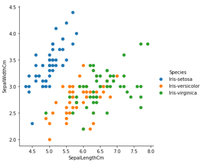


In [12]:
```
plt.figure(figsize=(8,4))
sns.heatmap(df.corr(),annot=True,fmt=".0%") #draws  heatmap with input as the correlation matrix calculted by(df.corr())
plt.show()
```



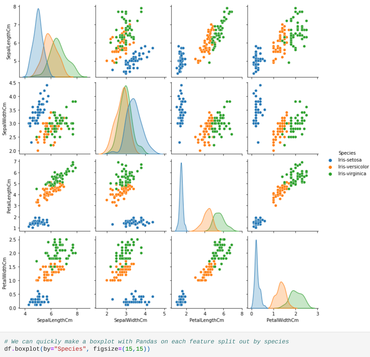
In [13]:
```
# We'll use seaborn's FacetGrid to color the scatterplot by species
sns.FacetGrid(df, hue="Species", height=5).map(plt.scatter, "SepalLengthCm", "SepalWidthCm").add_legend()
```

Out[13]:
```
<seaborn.axisgrid.FacetGrid at 0x1ee1709f760>
```



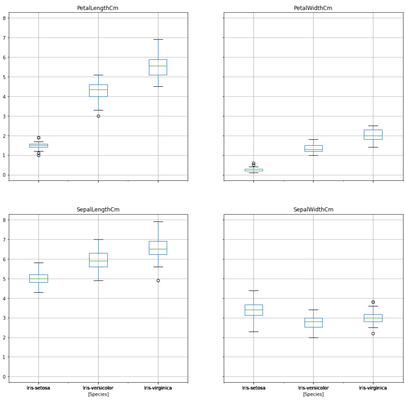
In [14]:
```
#let Create a pair plot of some columns
sns.pairplot(df.iloc[:,:],hue='Species')  # graph also  tell us about the the realationship between the two columns
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x1ee17139c10>
```



In [15]:
```
# We can quickly make a boxplot with Pandas on each feature split out by species
df.boxplot(by="Species", figsize=(15,15))
```

Out[15]:
```
array([[<AxesSubplot:title={'center':'PetalLengthCm'}, xlabel='[Species]'>,
        <AxesSubplot:title={'center':'PetalWidthCm'}, xlabel='[Species]'>],
       [<AxesSubplot:title={'center':'SepalLengthCm'}, xlabel='[Species]'>,
        <AxesSubplot:title={'center':'SepalWidthCm'}, xlabel='[Species]'>]],
      dtype=object)
```



In [16]:
```
# importing alll the necessary packages to Logistic Regression
from sklearn.linear_model import LogisticRegression  # for Logistic Regression algorithm
from sklearn.model_selection import train_test_split #to split the dataset for training and testing
from sklearn import metrics #for checking the model accuracy
```

Splitting the Data into Training and Testing Dataset

In [17]:
```
X=df.iloc[:,0:4]
Y=df["Species"]
X.head()
```

Out[17]:

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

In [18]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,random_state=0)# in this our main data is split into train and test
# the attribute test_size=0.3 splits the data into 70% and 30% ratio. train=70% and test=30%
print("Train Shape",X_train.shape)
print("Test Shape",X_test.shape)
```

```
Train Shape (112, 4)
Test Shape (38, 4)
```

Logistic Regression

In [19]:
```
log = LogisticRegression()
log.fit(X_train,Y_train)
prediction1=log.predict(X_test)
print('The accuracy of the Logistic Regression is',metrics.accuracy_score(prediction,Y_test))
```

```
The accuracy of the Logistic Regression is 0.9736842105263158
```

In [ ]: