Excercise 2 - Pandas and Visualization

Open a Jupyter notebook and import pandas, NumPy, matplotlib, seaborn and Sklearn.

```
[6] import pandas as pd
    import numpy as np
    import matplotlib as plt
    import seaborn as sns
```

Load the inbuilt iris dataset.

```
ds = pd.read_csv('Iris.csv')
```

Peek into this data using head(), info() and glance over some statistics using describe().

```
[8] ds.head(10)
```

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

```
[9] ds.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
```

describe() helps to view the basic statistical parameters or the numeric values of count, mean, std, min , max in a dataframe.

```
ds.describe()
```

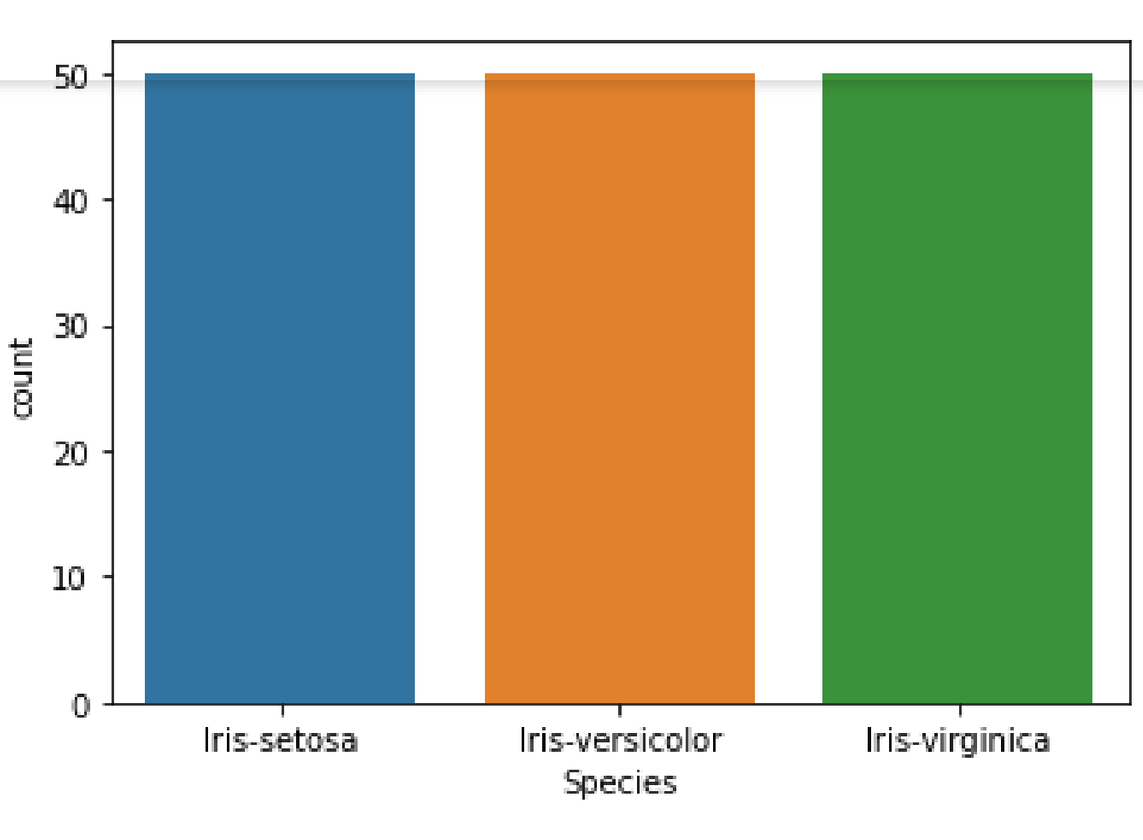
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	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

Plot the distribution of all numerical features and the categorical target using matplotlib and observe the plots.

```
numerical = [
  'Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
categorical = ['Species']

ds = ds[numerical + categorical]
ds.shape

(150, 6)
```

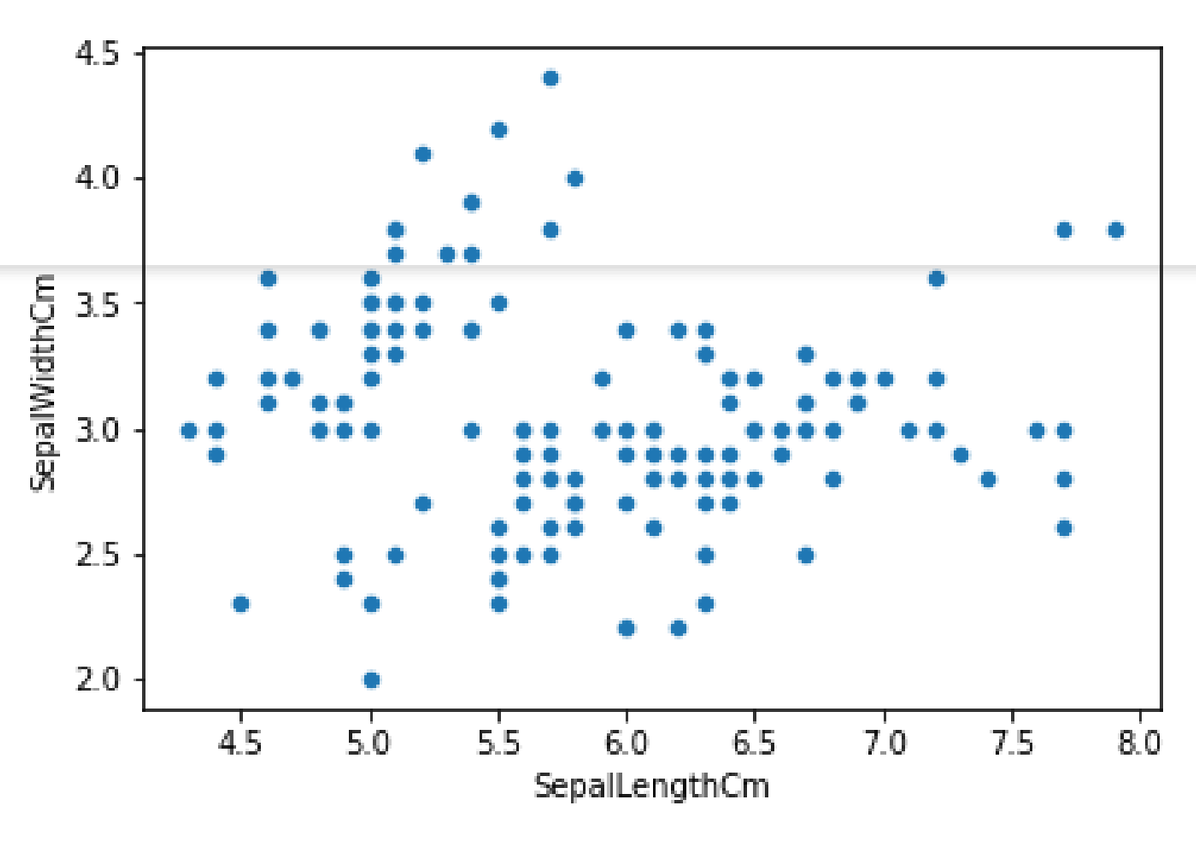
```
[12] import warnings
     warnings.filterwarnings("ignore")
```

```
[13] sns.countplot(ds['Species']);
```

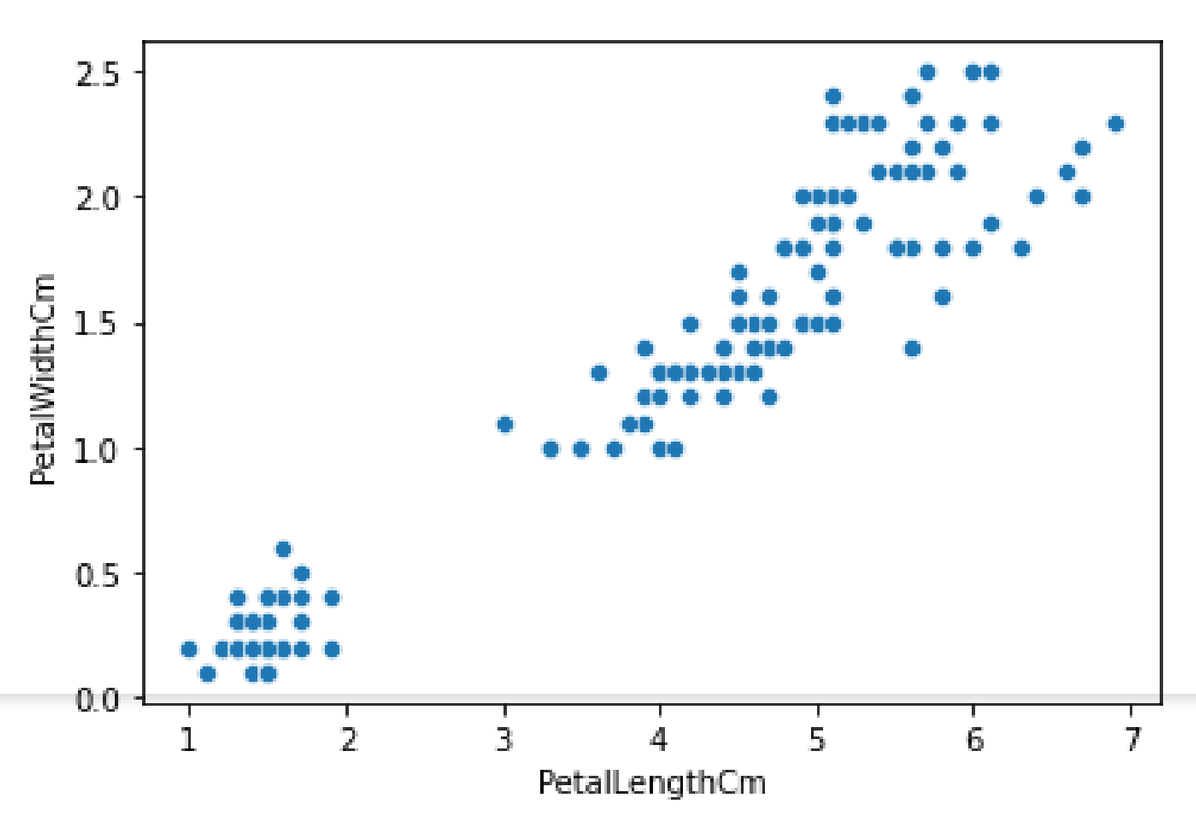


Plot a "feature pair-wise" scatter plot to see how the numerical features are correlated to each other and print out the pairwise correlation coefficients between the numerical features.

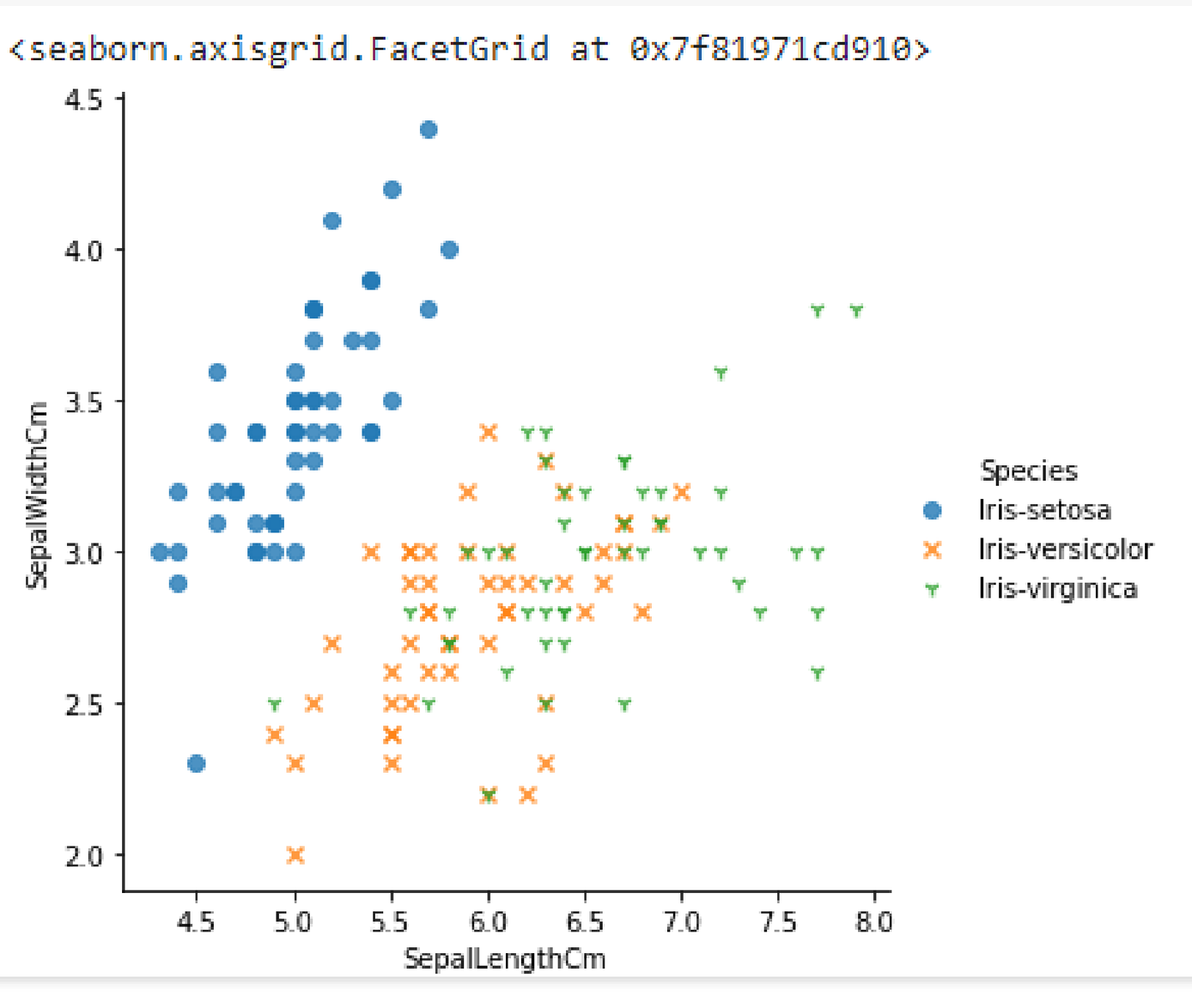
```
[14] sns.scatterplot(x=ds['SepalLengthCm'], y=ds['SepalWidthCm']);
```




```
sns.scatterplot(x=ds['PetalLengthCm'], y=ds['PetalWidthCm']);
```




```
sns.lmplot( x="SepalLengthCm", y="SepalWidthCm", data=ds, fit_reg=False, hue='Species', legend=True, markers=["o", "x", "1"])

<seaborn.axisgrid.FacetGrid at 0x7f81971cd910>
```



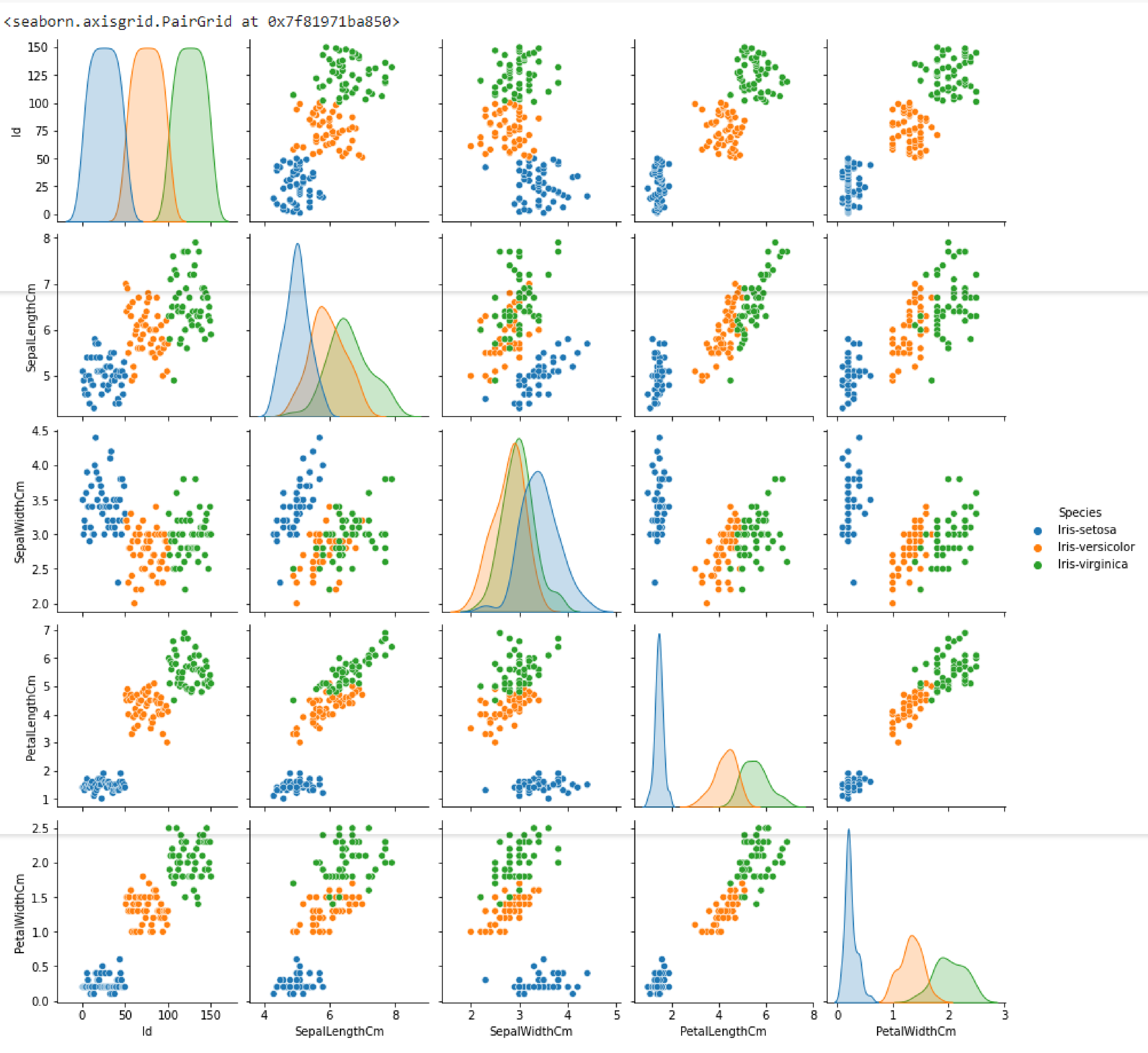
Plot a "feature pair-wise" scatter plot to see how the numerical features are correlated to each other.

Observation 1: Pairplot captures multiple relationship between two variables, the SepalLength and SepalWidth in a dataframe.

The hue parameter categorizes the data, Species.

The dataset 'Iris' and the column 'species' for hue is used for the pairplot() function.

```
sns.pairplot(ds, hue="Species")

<seaborn.axisgrid.PairGrid at 0x7f81971ba850>
```



Print out the pairwise correlation coefficients between the numerical features

Observation 2: Correlation coefficient formulas are used to find how strong a relationship is between data. The formulas return a value between -1 and 1, where:

1-> positive relationship. -1 -> indicates a strong negative relationship. zero -> no relationship.

```
[18] import pandas as pd
     import scipy.stats as stats
```

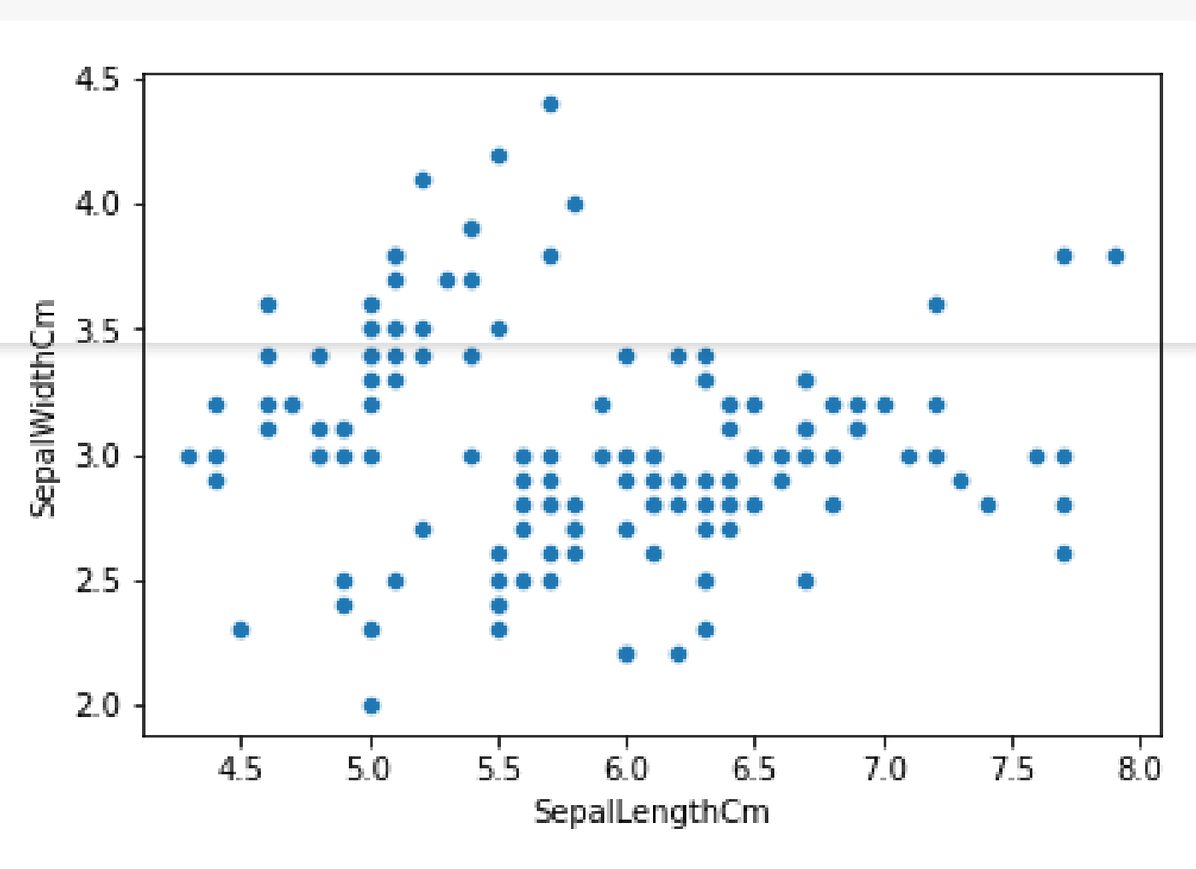
```
[19] ds = pd.read_csv('Iris.csv')
```

```
[28] corr = np.corrcoef(ds['SepalLengthCm'], ds['SepalWidthCm'])
     print("Confusion Matrix : ", corr)

Correlation Matrix :  [[ 1.         -0.10936925]
 [-0.10936925  1.        ]]
```

```
[29] corr = np.corrcoef(ds['PetalLengthCm'], ds['PetalLengthCm'])
     print("Confusion  Matrix :" , corr)

Confusion  Matrix : [[1. 1.]
 [1. 1.]]
```

Observation 3: This scatter plot indicates a correlation coefficient of -1 for every positive increase in one variable, such that there will be a negative decrease of a fixed proportion with the other.

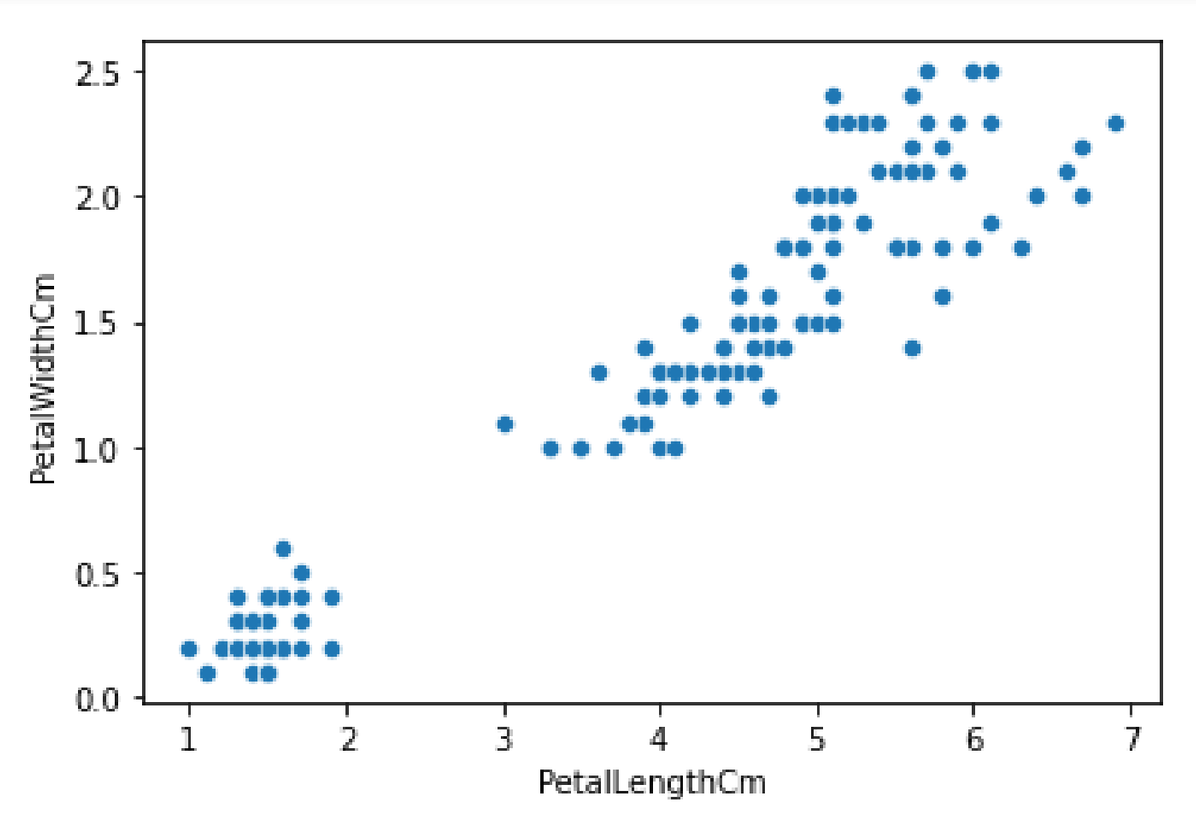
```
[26] sns.scatterplot(x=ds['SepalLengthCm'], y=ds['SepalWidthCm']);
```




```
r = stats.pearsonr(ds['SepalLengthCm'], ds['SepalWidthCm'])
print(r)
print("Coefficient Correlation =", r[0])

(-0.10936924995064937, 0.1827652152713699)
Coefficient Correlation = -0.10936924995064937
```

Observation 4: This scatter plot shows a positive correlation which indicates a 1. Here, for every positive increase in one variable, there is a positive increase of a fixed proportion in the other also. When the Petal Length increases there will be a perfect correlation with Petal Width also.

```
sns.scatterplot(x=ds['PetalLengthCm'], y=ds['PetalWidthCm']);
```




```
r = stats.pearsonr(ds['PetalLengthCm'], ds['PetalLengthCm'])
print(r)
print("Correlation Coefficient =", r[0])

[25]
(1.0, 0.0)
Correlation Coefficient = 1.0
```