In this tutorial you'll learn all about **histograms** and **density plots**.

Set up the notebook

As always, we begin by setting up the coding environment. (*This code is hidden, but you can un-hide it by clicking on the "Code" button immediately below this text, on the right.*)

Hide Output

In [1]:

```
import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
print("Setup Complete")
```

Select a dataset

We'll work with a dataset of 150 different flowers, or 50 each from three different species of iris (*Iris setosa*, *Iris versicolor*, and *Iris virginica*).

Iris Setosa

Iris Versicolor

Iris Virginica

Load and examine the data

Each row in the dataset corresponds to a different flower. There are four measurements: the sepal length and width, along with the petal length and width. We also keep track of the corresponding species.

In [2]:

```
# Path of the file to read
iris_filepath = "../input/iris.csv"

# Read the file into a variable iris_data
iris_data = pd.read_csv(iris_filepath, index_col="Id")

# Print the first 5 rows of the data
iris_data.head()
```

Out[2]:

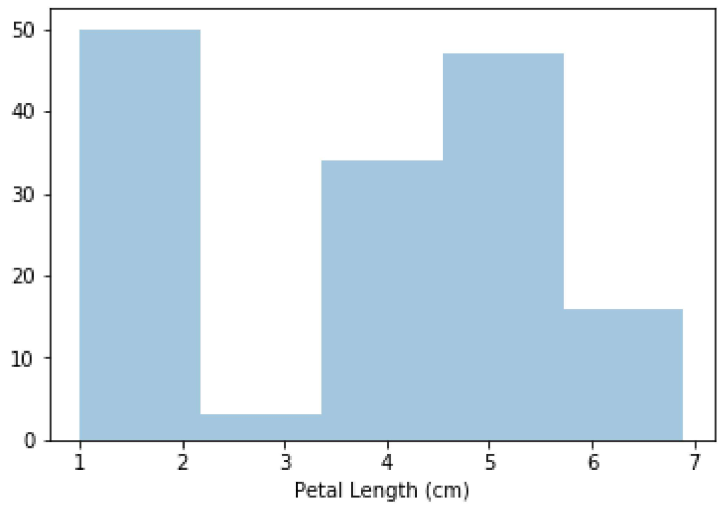
	Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)	Species
Id					
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa

Histograms

Say we would like to create a **histogram** to see how petal length varies in iris flowers. We can do this with the `sns.distplot` command.

In [3]:
```
# Histogram
sns.distplot(a=iris_data['Petal Length (cm)'], kde=False)
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f136e29e8d0>
```



We customize the behavior of the command with two additional pieces of information:

- `a=` chooses the column we'd like to plot (*in this case, we chose* `'Petal Length (cm)'`).
- `kde=False` is something we'll always provide when creating a histogram, as leaving it out will create a slightly different plot.

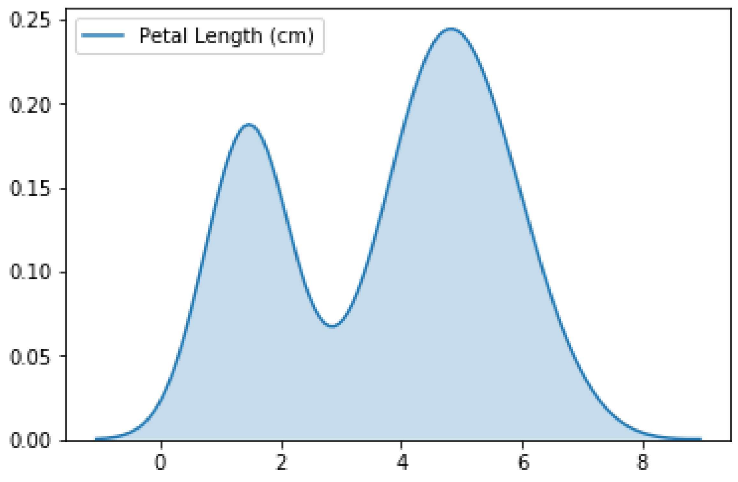
Density plots

The next type of plot is a **kernel density estimate (KDE)** plot. In case you're not familiar with KDE plots, you can think of it as a smoothed histogram.

To make a KDE plot, we use the `sns.kdeplot` command. Setting `shade=True` colors the area below the curve (*and* `data=` *has identical functionality as when we made the histogram above*).

In [4]:
```
# KDE plot
sns.kdeplot(data=iris_data['Petal Length (cm)'], shade=True)
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f136e1e3da0>
```



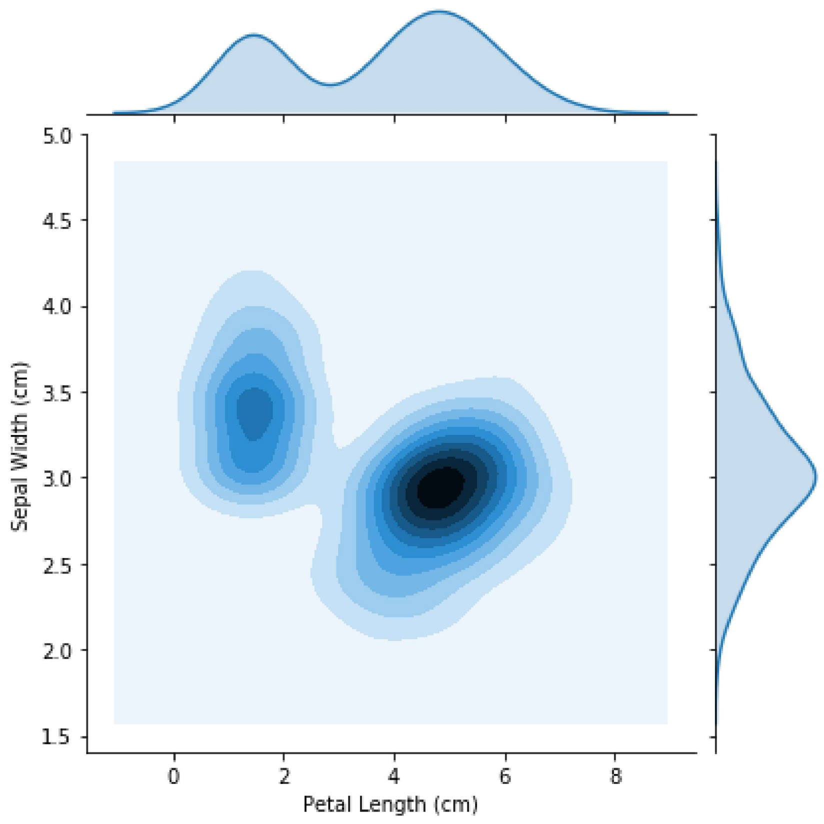
2D KDE plots

We're not restricted to a single column when creating a KDE plot. We can create a **two-dimensional (2D) KDE plot** with the `sns.jointplot` command.

In the plot below, the color-coding shows us how likely we are to see different combinations of sepal width and petal length, where darker parts of the figure are more likely.

In [5]:
```
# 2D KDE plot
sns.jointplot(x=iris_data['Petal Length (cm)'], y=iris_data['Sepal Width (cm)'], kind="kde")
```

Out[5]:
```
<seaborn.axisgrid.JointGrid at 0x7f136e138ba8>
```



Note that in addition to the 2D KDE plot in the center,

- the curve at the top of the figure is a KDE plot for the data on the x-axis (in this case, `iris_data['Petal Length (cm)']`), and
- the curve on the right of the figure is a KDE plot for the data on the y-axis (in this case, `iris_data['Sepal Width (cm)']`).

Color-coded plots

For the next part of the tutorial, we'll create plots to understand differences between the species. To accomplish this, we begin by breaking the dataset into three separate files, with one for each species.

In [6]:

```
# Paths of the files to read
iris_set_filepath = "../input/iris_setosa.csv"
iris_ver_filepath = "../input/iris_versicolor.csv"
iris_vir_filepath = "../input/iris_virginica.csv"

# Read the files into variables
iris_set_data = pd.read_csv(iris_set_filepath, index_col="Id")
iris_ver_data = pd.read_csv(iris_ver_filepath, index_col="Id")
iris_vir_data = pd.read_csv(iris_vir_filepath, index_col="Id")

# Print the first 5 rows of the Iris versicolor data
iris_ver_data.head()
```

Out[6]:

	Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)	Species
Id					
51	7.0	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4.0	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor

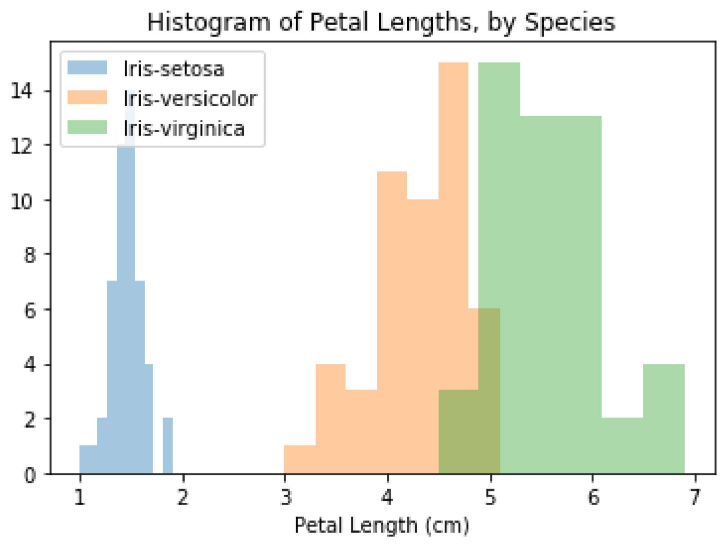
In the code cell below, we create a different histogram for each species by using the `sns.distplot` command (*as above*) three times. We use `label=` to set how each histogram will appear in the legend.

In [7]:
```python
# Histograms for each species
sns.distplot(a=iris_set_data['Petal Length (cm)'], label="Iris-setosa", kde=False
)
sns.distplot(a=iris_ver_data['Petal Length (cm)'], label="Iris-versicolor", kde=F
alse)
sns.distplot(a=iris_vir_data['Petal Length (cm)'], label="Iris-virginica", kde=Fa
lse)

# Add title
plt.title("Histogram of Petal Lengths, by Species")

# Force legend to appear
plt.legend()
```

Out[7]:
```
<matplotlib.legend.Legend at 0x7f136dfc8400>
```



In this case, the legend does not automatically appear on the plot. To force it to show (for any plot type), we can always use `plt.legend()`.

We can also create a KDE plot for each species by using `sns.kdeplot` (*as above*). Again, `label=` is used to set the values in the legend.

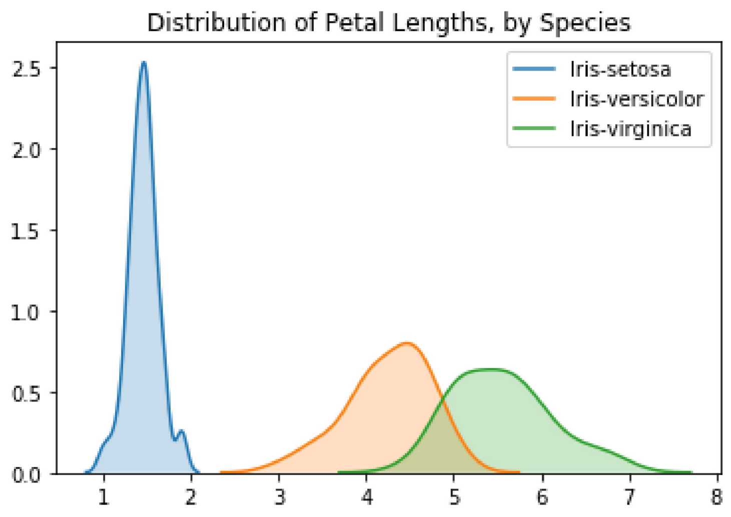
In [8]:

```
# KDE plots for each species
sns.kdeplot(data=iris_set_data['Petal Length (cm)'], label="Iris-setosa", shade=True)
sns.kdeplot(data=iris_ver_data['Petal Length (cm)'], label="Iris-versicolor", shade=True)
sns.kdeplot(data=iris_vir_data['Petal Length (cm)'], label="Iris-virginica", shade=True)

# Add title
plt.title("Distribution of Petal Lengths, by Species")
```

Out[8]:

```
Text(0.5, 1.0, 'Distribution of Petal Lengths, by Species')
```



One interesting pattern that can be seen in plots is that the plants seem to belong to one of two groups, where *Iris versicolor* and *Iris virginica* seem to have similar values for petal length, while *Iris setosa* belongs in a category all by itself.

In fact, according to this dataset, we might even be able to classify any iris plant as *Iris setosa* (as opposed to *Iris versicolor* or *Iris virginica*) just by looking at the petal length: if the petal length of an iris flower is less than 2 cm, it's most likely to be *Iris setosa*!

What's next?

Put your new skills to work in a **coding exercise**!