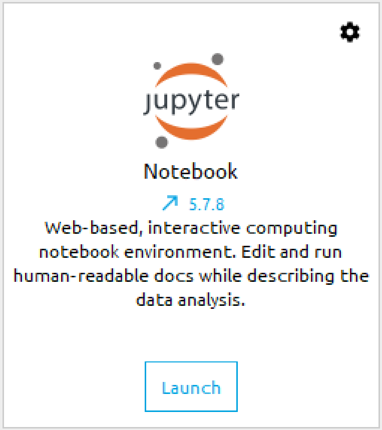
Jupyter

Foundations of AI Academy

Jupyter

A programming environment that allows you to run small code snippets at a time

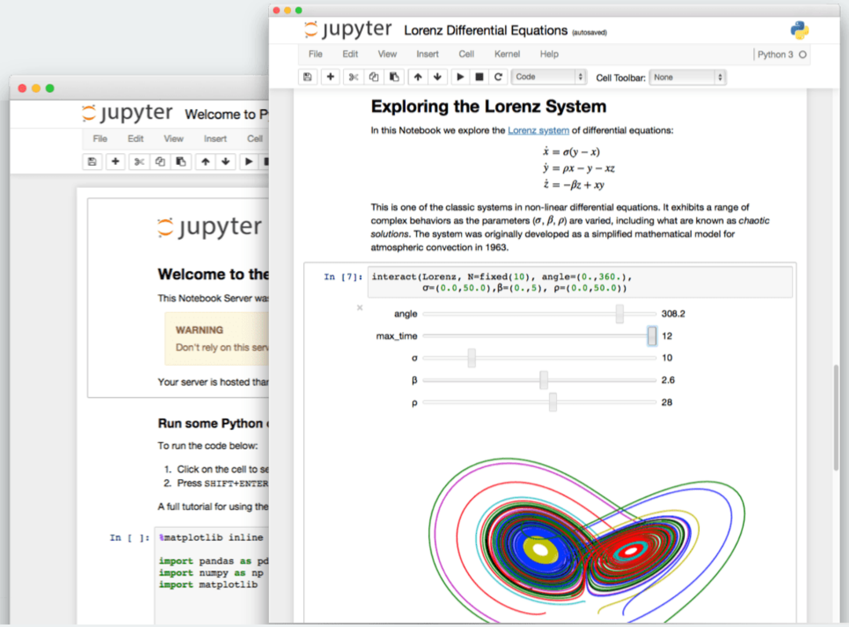


Notebook

5.7.8

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch



jupyter Welcome to P
File Edit View Insert Cell
jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None

Welcome to the

This Notebook Server wa

WARNING

Don't rely on this serv

Your server is hosted thar

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

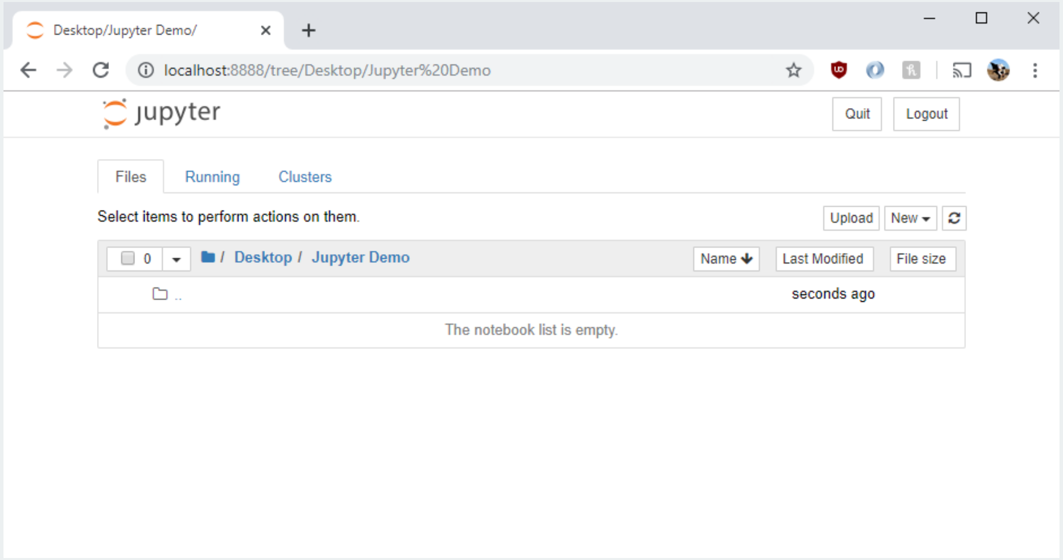
This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

angle 308.2
max_time 12
σ 10
β 2.6
ρ 28

Jupyter

Upon loading Jupyter, your browser opens a webpage that allow for **notebooks**



Select items to perform actions on them.

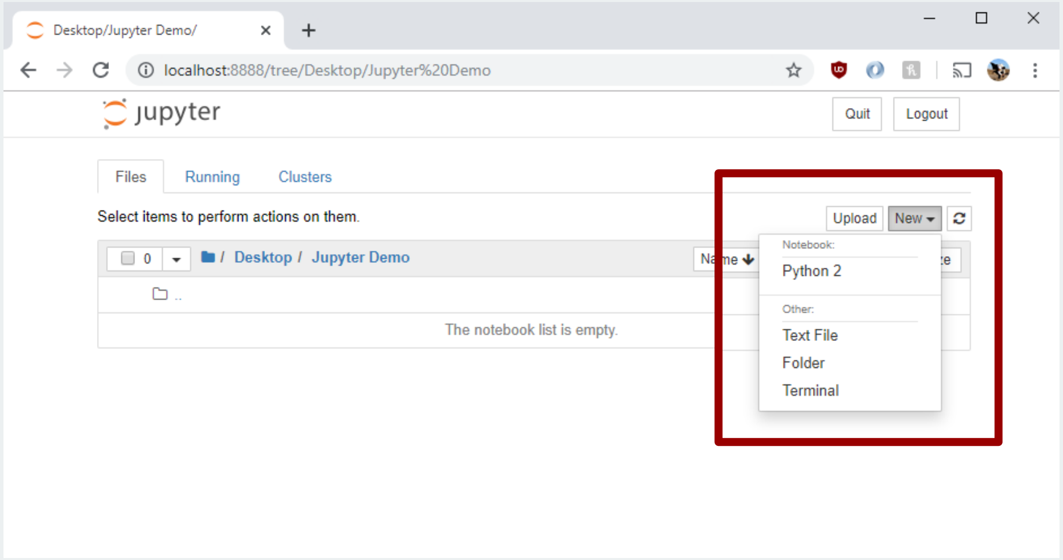
0 / Desktop / Jupyter Demo

.. seconds ago

The notebook list is empty.

Jupyter

The **New** option allows you to create a new notebook

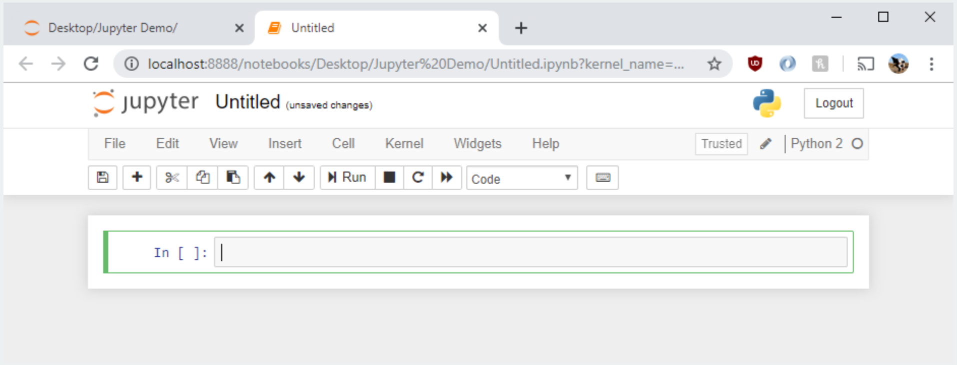


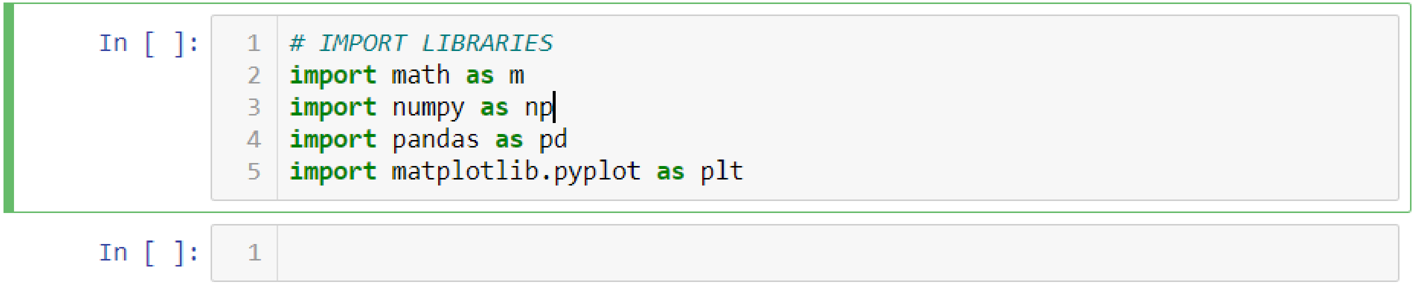
Select items to perform actions on them.

The notebook list is empty.

Jupyter Notebooks

You can think of a Jupyter Notebook like the Spyder Console, allowing you to enter in small snippets of code that are run sequentially

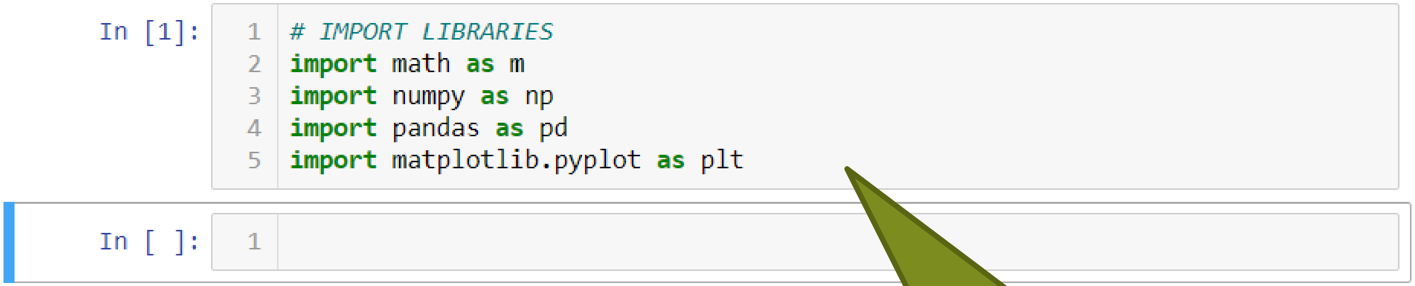




In [ ]:

```
# IMPORT LIBRARIES
import math as m
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

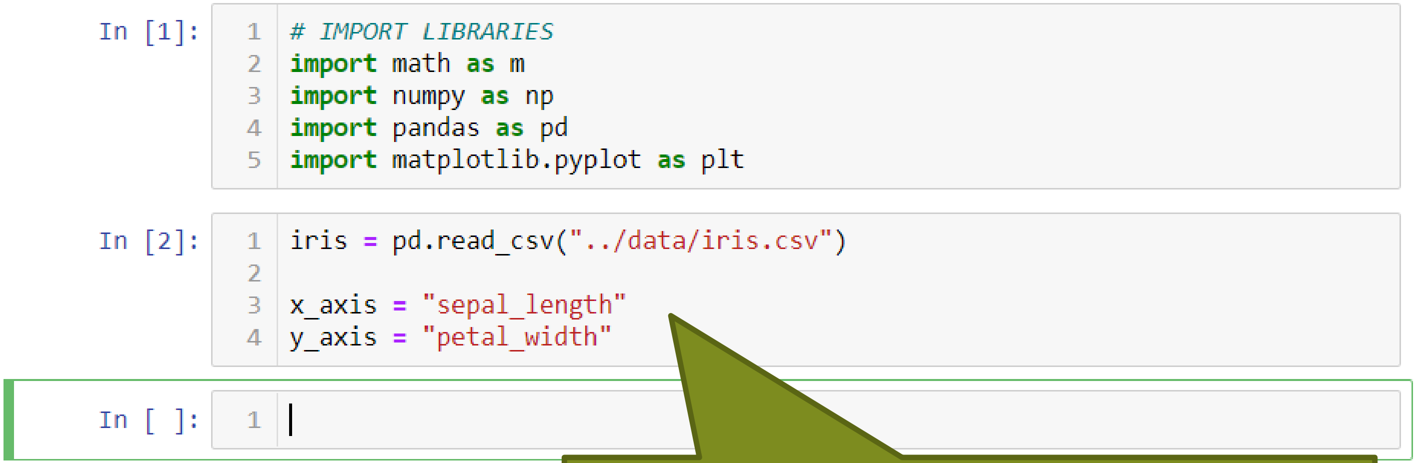
In [ ]:



```
In [1]:
# IMPORT LIBRARIES
import math as m
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
In [ ]:
```

Now, only this code block is loaded into memory



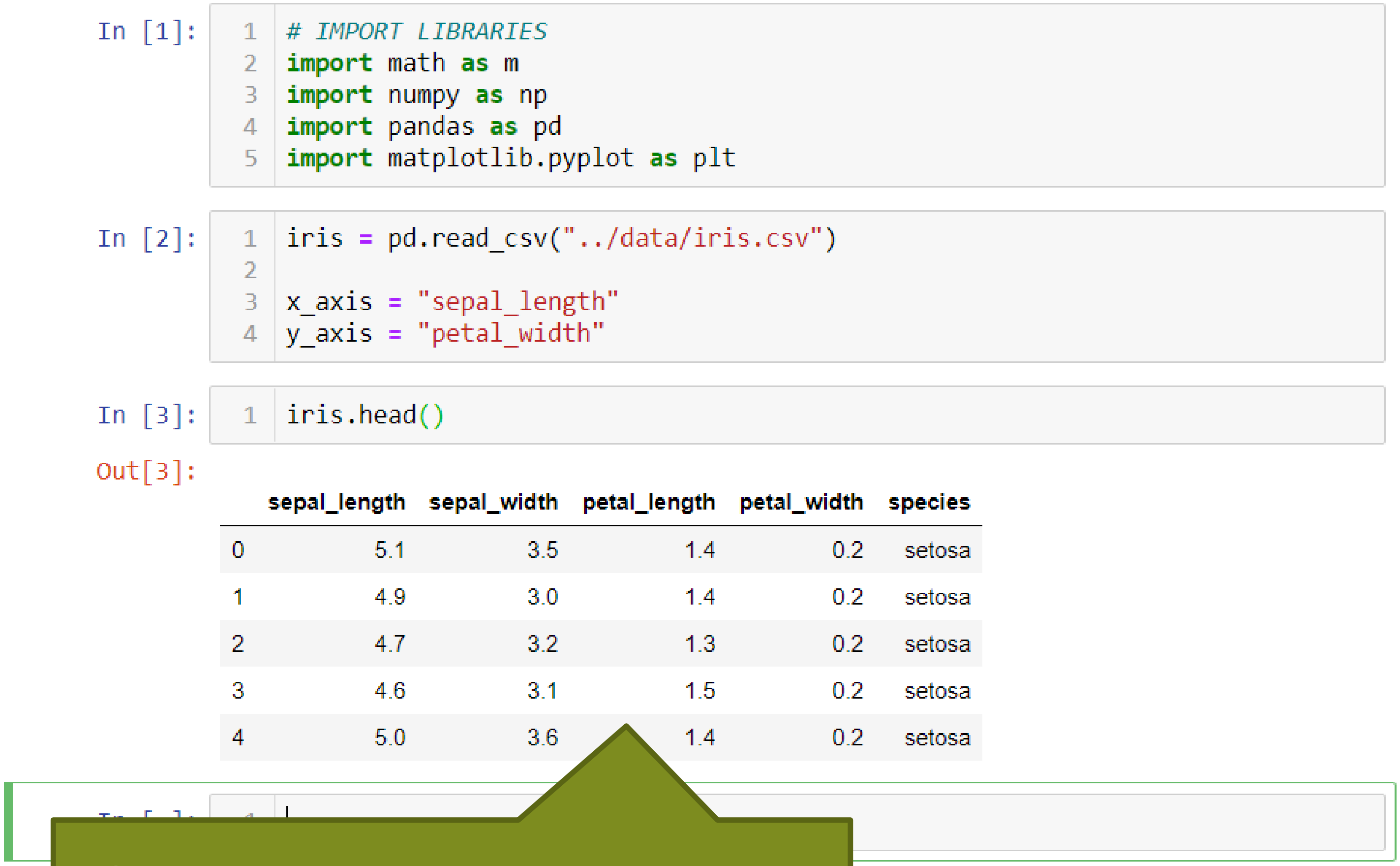
```
In [1]: # IMPORT LIBRARIES
        import math as m
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

In [2]: iris = pd.read_csv("../data/iris.csv")

        x_axis = "sepal_length"
        y_axis = "petal_width"

In [ ]: 
```

I can load datasets similar to using Spyder, but now can view information line-by-line


```
In [1]:
# IMPORT LIBRARIES
import math as m
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
In [2]:
iris = pd.read_csv("../data/iris.csv")

x_axis = "sepal_length"
y_axis = "petal_width"
```

```
In [3]:
iris.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Outputs like print() and .head() will be displayed inline with code

```
In [3]: iris.head()
```

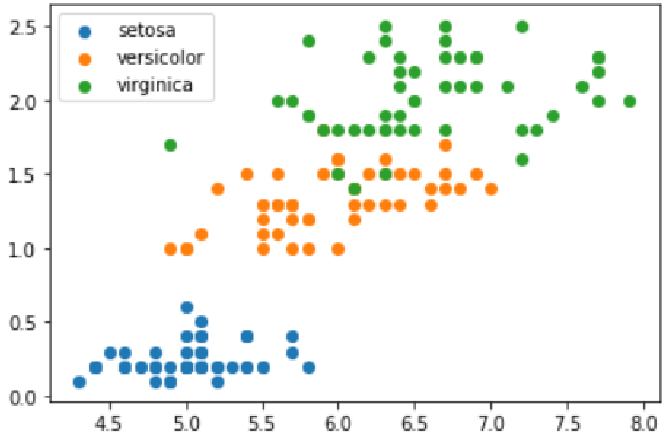
Out[3]:

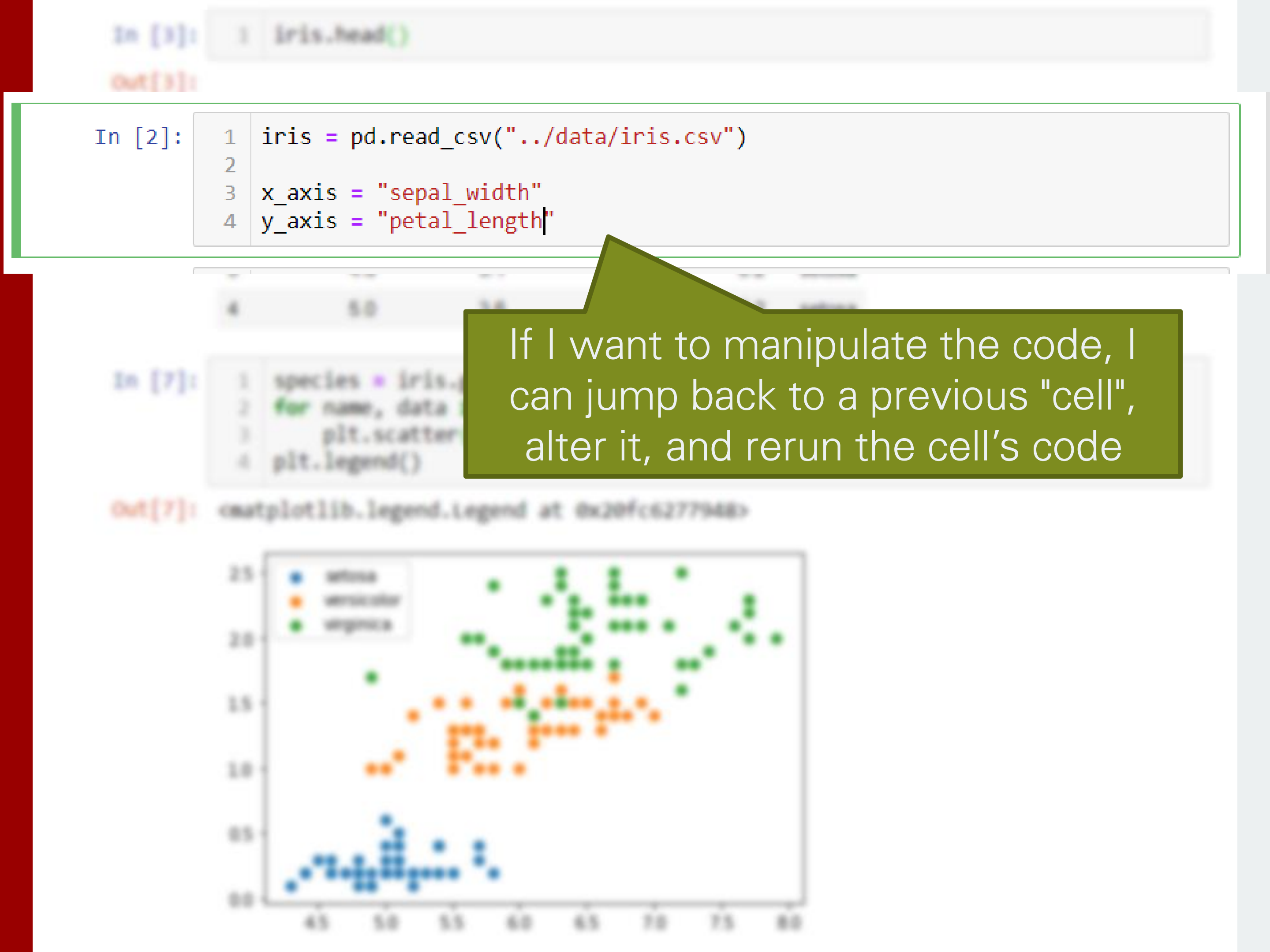
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	[illegible]	[illegible]	[illegible]
2	4.7	3.2	[illegible]	[illegible]	[illegible]
3	4.6	3.1	[illegible]	[illegible]	[illegible]
4	5.0	3.6	[illegible]	[illegible]	[illegible]

Likewise, Matplotlib plots appear in line with the rest of the code.

```
In [7]: species = iris.groupby('species')
        for name, data in species:
            plt.scatter(data[x_axis], data[y_axis], label=name)
        plt.legend()
```

```
Out[7]: <matplotlib.legend.Legend at 0x20fc6277948>
```





In [2]:
```python
iris = pd.read_csv("../data/iris.csv")

x_axis = "sepal_width"
y_axis = "petal_length"
```

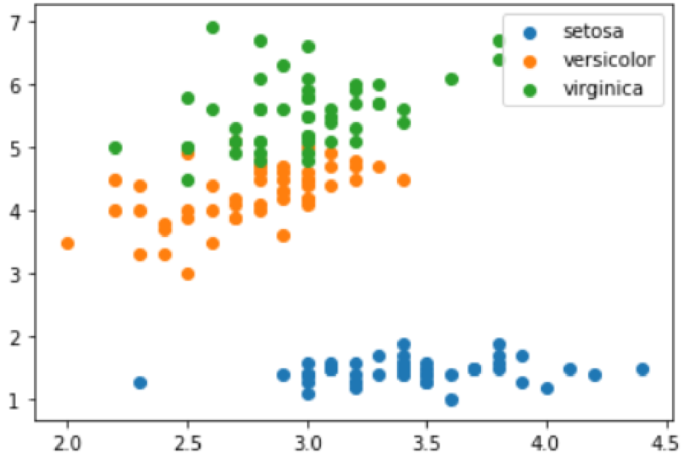
If I want to manipulate the code, I can jump back to a previous "cell", alter it, and rerun the cell’s code

In [3]:
```
iris.head()
```
Out[3]:

In [2]:
```
iris = pd.read_csv("../data/iris.csv")

x_axis = "sepal_width"
y_axis = "petal_length"
```

In [10]:
```
species = iris.groupby('species')
for name, data in species:
    plt.scatter(data[x_axis], data[y_axis], label=name)
plt.legend()
```
Out[10]:
```
<matplotlib.legend.Legend at 0x20fc62bbb88>
```



Pandas Data Manipulation

Sometimes the data you need doesn’t exist, you so you want to calculate it

```
>>> df = pd.DataFrame({'temp_c': [17.0, 25.0]},
         index=['Portland', 'Berkeley'])
>>> df
          temp_c
Portland    17.0
Berkeley    25.0
```

Pandas Data Manipulation

You can use the **assign** function to assign new values based on other values

```
>>> df.assign(temp_f=df['temp_c'] * 9 / 5 + 32)
          temp_c  temp_f
Portland    17.0    62.6
Berkeley    25.0    77.0
```

Pandas Data Manipulation

You can use the **assign** function to assign new values based on other values

```
>>> df.assign(temp_f=df['temp_c'] * 9 / 5 + 32)
          temp_c  temp_f
Portland    17.0    62
Berkeley    25.0    7
```

DataFrames will process the new column’s data on a row by row basis for each row’s `temp_c`