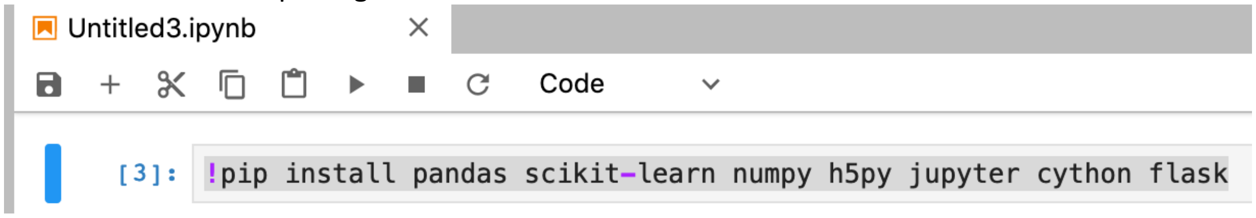
Lab: Jupyter Notebooks - Advanced Features 2

Welcome to the 3rd Jupyter Lab.
In this lab we'll explore even more features which will help you becoming more productive using Jupyter.
Let's start with an empty Notebook.
We concentrate on Python here, but you can also run Scala, R and many other programming languages from Jupyter Notebooks.

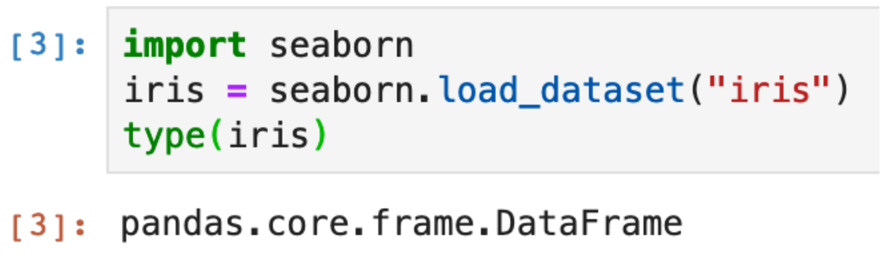
1. Sometimes it is necessary to install 3rd party libraries. This is usually done from a terminal using command line. But Jupyter lets us to run such commands from a code cell as well. You can execute any shell command using the exclamation mark "!". We now use the Python Pip package manager to install some libraries, please type and then run:
   !pip install Pandas Scikit-learn Numpy h5py Cython Flask Seaborn Scipy Numpy Matplotlib Ipython Jupyter Sympy Nose
   This will install some packages we need later



```
[3]: !pip install pandas scikit-learn numpy h5py jupyter cython flask
```

2. Once the installation was successful (please watch out for error messages) we can proceed and load a data set which comes with "Seaborn", a plotting library. We load the "Iris" data set which contains information about flowers:
   import seaborn

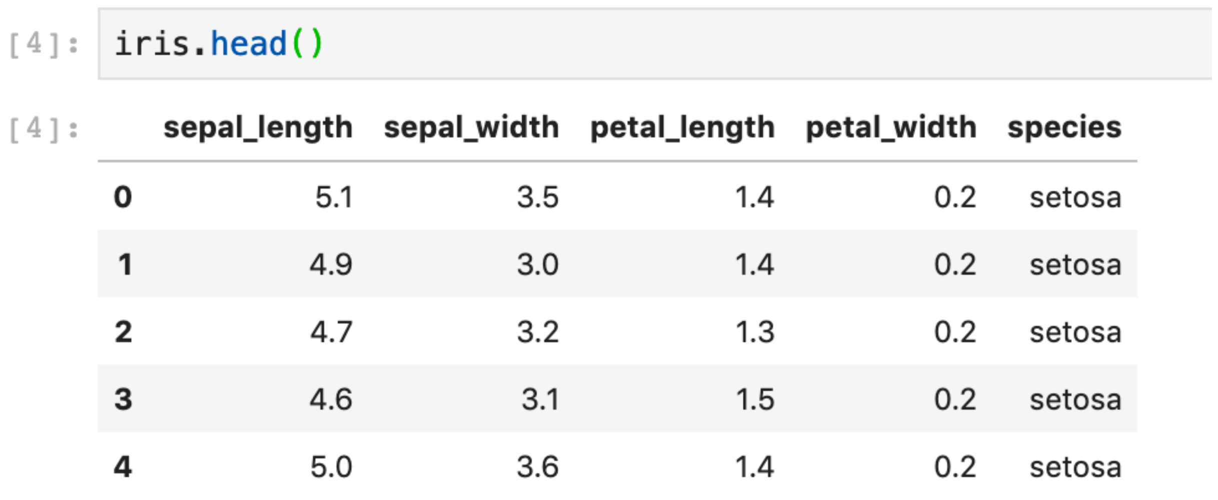
iris = seaborn.load_dataset("iris")
type(iris)



```
[3]: import seaborn
     iris = seaborn.load_dataset("iris")
     type(iris)

[3]: pandas.core.frame.DataFrame
```

3. As you can see, we've successfully loaded a data set. The data is contained in the "Iris" object which is a Pandas DataFrame. A data frame is some sort of a table containing data. Let's have a look inside this dataframe. Please type and run the following code:
iris.head()



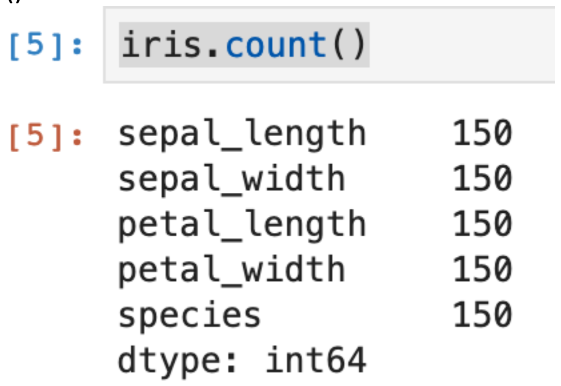
```
[4]: iris.head()
```

[4]:

	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

4. You can see the first five rows of the data set. There are four columns describing properties of the flower and the last column tells us about the species of that plant. Now let's find out how many data points we have in total by typing:

iris.count()

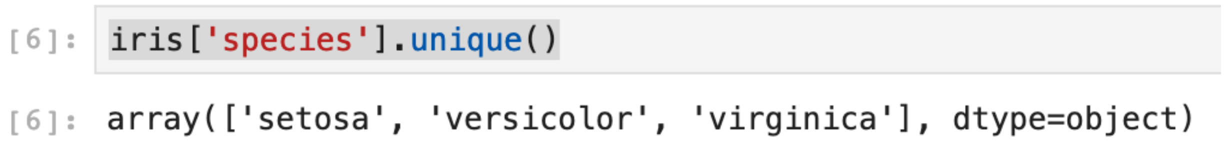


```
[5]: iris.count()

[5]: sepal_length    150
     sepal_width     150
     petal_length    150
     petal_width     150
     species         150
     dtype: int64
```

5. We see that we have data from 150 real flowers. But we still don't know how many flower types (species) we have in the data sets, so let's type:

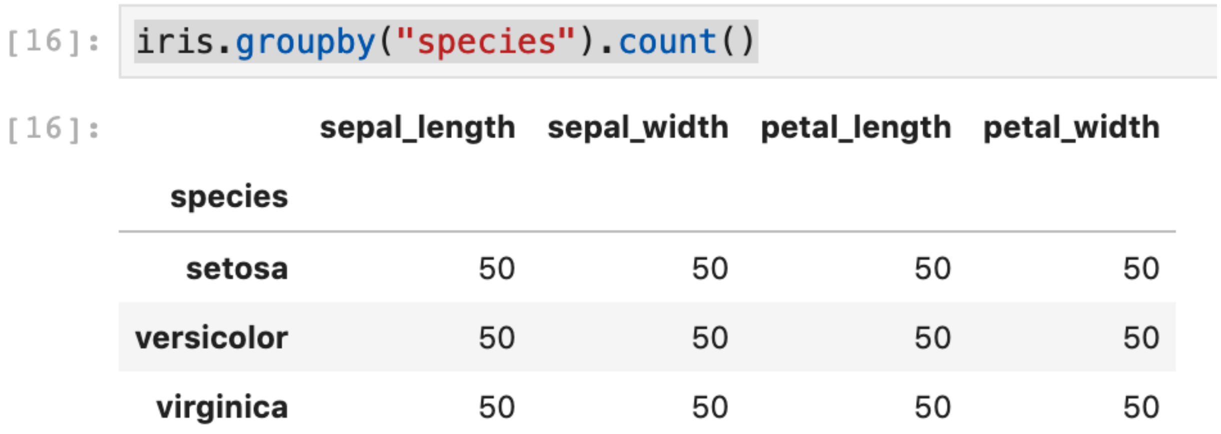
iris['species'].unique()



```
[6]: iris['species'].unique()

[6]: array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

6. Now we want to know if we have a balanced data set, this means we have roughly the same number of data points per species. Let’s type:
iris.groupby("species").count()



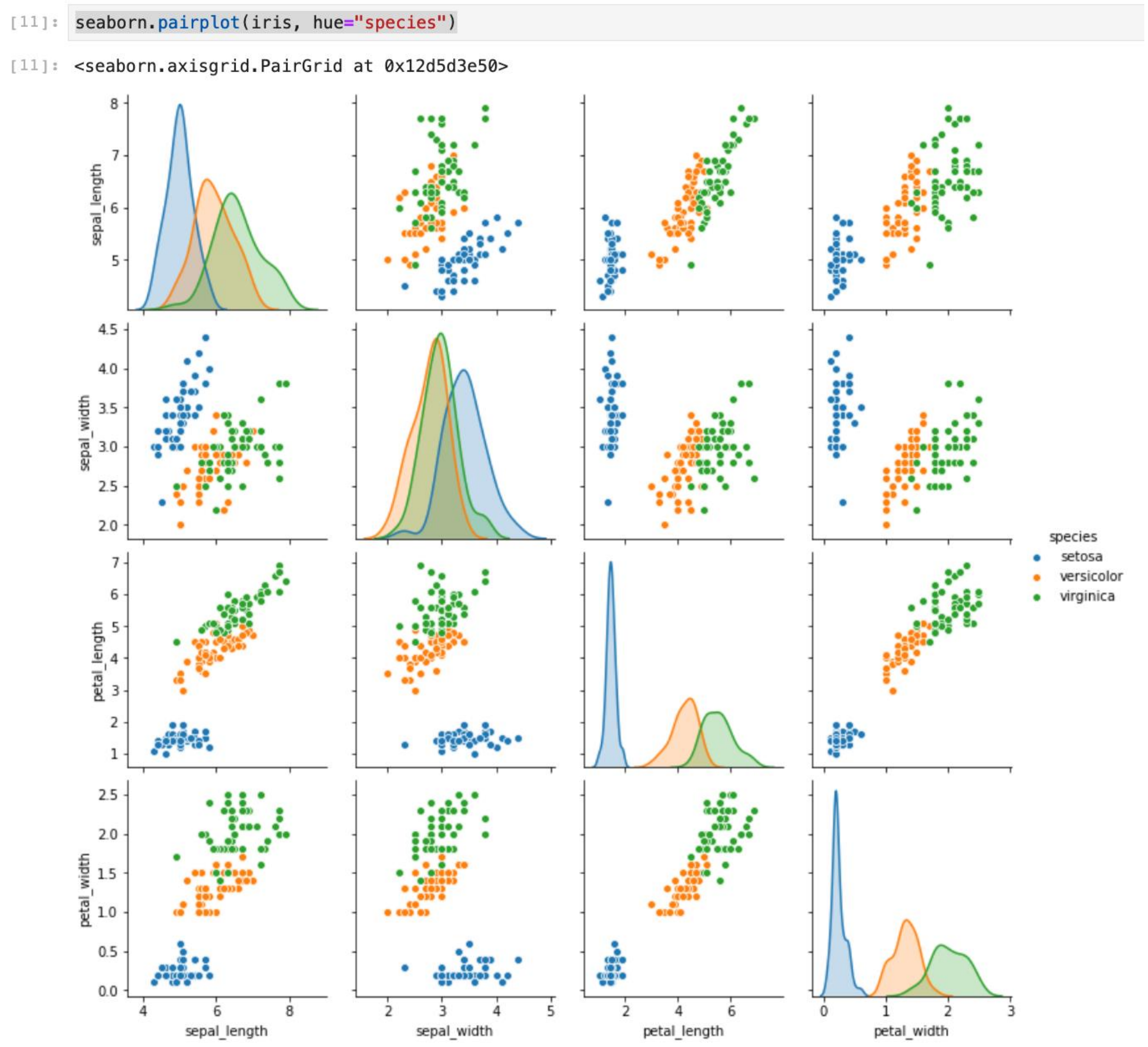
```
[16]: iris.groupby("species").count()

[16]:
```

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	50	50	50	50
versicolor	50	50	50	50
virginica	50	50	50	50

7. This is a perfectly balanced data set since every species is represented with 50 examples. Now let’s have a look at all individual data points at once to get an idea how the data distributions look like. Please type:

seaborn.pairplot(iris, hue="species")



```
[11]: seaborn.pairplot(iris, hue="species")

[11]: <seaborn.axisgrid.PairGrid at 0x12d5d3e50>
```

8. This gives us a lot of information for a single line of code. First, we see the data distributions per column and species on the diagonal. Then we see all pair-wise scatter plots on the remaining tiles, again broken down by color. It is, for example, obvious to see that a line can be drawn to separate “setosa” against “versicolor” and “virginica”. In later courses, we’ll of course teach how the overlapping species can be separated as well. This is called supervised machine learning using non-linear classifiers by the way.

This concludes this lab, I hope you’ve enjoyed it!