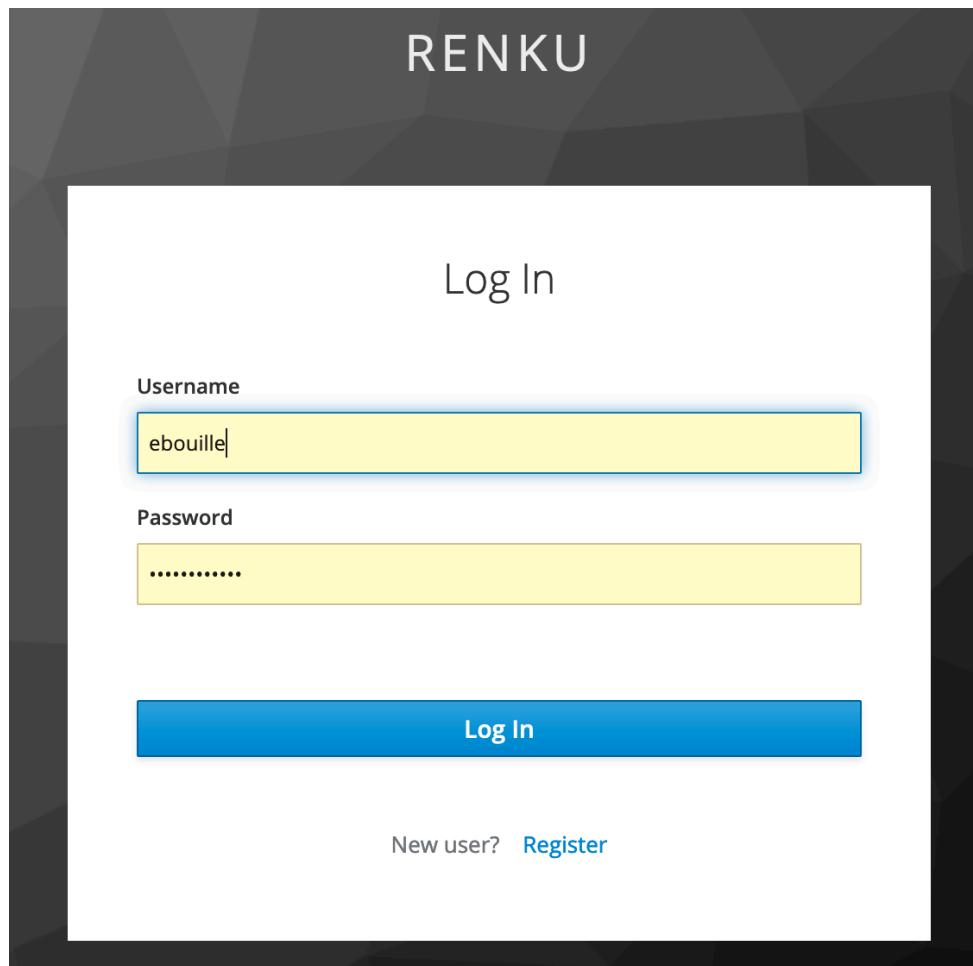

Overview Jupyter notebook

Crash course Python 3.x

Start your engines

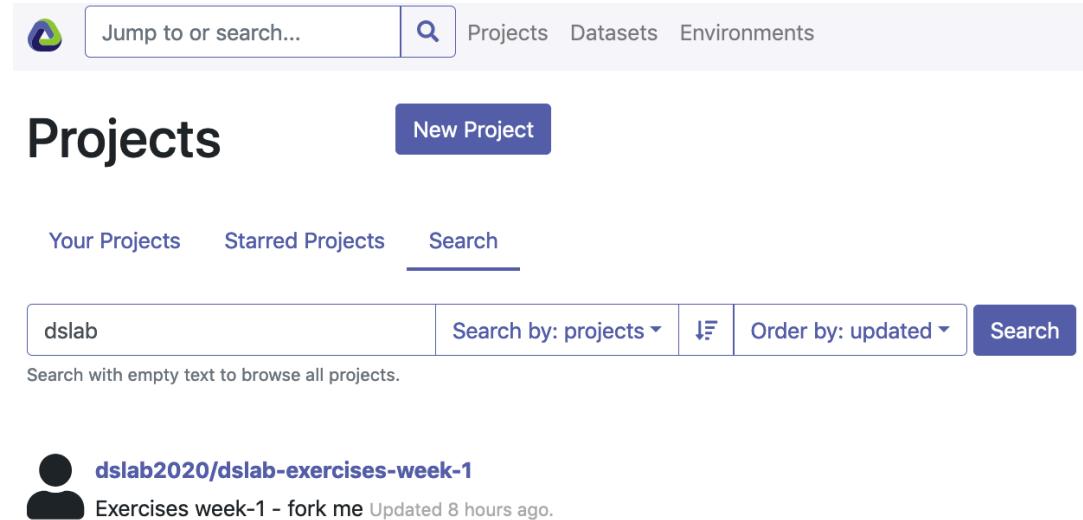
- Head over to the RENKU webpage
 - <https://renku.iccluster.epfl.ch>
 - You will need the EPFL VPN if connecting from outside the EPFL network

1. Login with your gaspar accounts



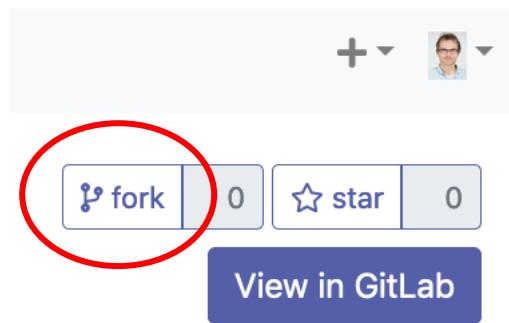
The image shows the Renku login interface. At the top, the word "RENKU" is displayed in white capital letters. Below it, the title "Log In" is centered. There are two input fields: "Username" containing "ebouille" and "Password" containing several dots. A blue "Log In" button is at the bottom. Below the button, there is a link for "New user? Register".

2. Find project **dslab2020/dslab-exercises-week1**



The image shows the Renku search results for the project "dslab2020/dslab-exercises-week1". The search bar at the top contains "dslab". The results section shows one project entry: "dslab2020/dslab-exercises-week-1" by "Exercises week-1 - fork me" (Updated 8 hours ago). The interface includes tabs for "Your Projects", "Starred Projects", and "Search" (which is currently selected), and dropdowns for "Search by: projects" and "Order by: updated".

3. Open and fork project



4. Open forked project and start environment (1 CPU max)

A screenshot of the Environments page for a forked project. The top navigation bar includes Overview, Collaboration, Files, Datasets, Environments (underlined), and Settings. The main section is titled 'Start a new interactive environment'. It shows a 'Branch (only 1 available)' dropdown set to 'master', a 'Commit' dropdown showing 'ecea987 - Eric Pierre Bouillet Bouillet - 2020-02-14 15:20:38', and a 'Docker Image' status of 'available'. Under 'Default Environment', there are buttons for '/lab' and '/rstudio'. Configuration options include 'Number of CPUs' (0.1, 0.5, 1, 2, 2 is selected), 'Amount of Memory' (1G, 2G, 4G, 8G, 8G is selected), and 'Number of GPUs' (0, 1, 2, 1 is selected). A checkbox for 'Automatically fetch LFS data' is unchecked. At the bottom is a large blue 'Start environment' button, which is circled in red.

5. Open Jupyter Notebook

The screenshot shows the 'Interactive Environments' section of a web interface. A single environment is listed:

Branch	Commit	Status	Action
master	eceba987	Running	<button>Connect</button> <input type="radio"/> Stop <input type="checkbox"/> Get logs

A red circle highlights the 'Connect' button.

6. In Jupyter notebook, exercises are in notebook folder

The screenshot shows the Jupyter Notebook interface. On the left is a file browser:

Name	Last Modified
data	4 minutes ago
notebooks	4 minutes ago
Dockerfile	4 minutes ago
Y: environment.yml	4 minutes ago
M: README.md	4 minutes ago
requirements.txt	4 minutes ago

A red arrow points from the 'notebooks' folder to the 'Notebook' icon in the 'Launcher' sidebar.

The 'Launcher' sidebar contains:

- Notebook (Python 3)
- Console (Python 3)
- Other
 - Terminal
 - Text File
 - Markdown File
 - Show Contextual Help

7. Stop the environment when you are done

The screenshot shows the 'Interactive Environments' section. The environment status has changed to 'Stopped'.

Branch	Commit	Status	Action
master	eceba987	Stopped	<input type="radio"/> Connect <input checked="" type="radio"/> Stop <input type="checkbox"/> Get logs

A red circle highlights the 'Stop' button.

8. You can also access all your active environments

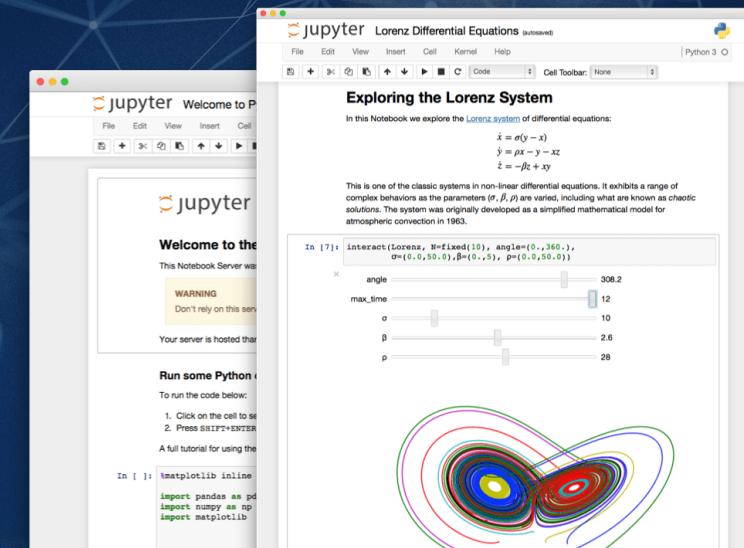
The screenshot shows the main navigation bar with the 'Environments' tab circled in red.

Interactive Environments

The screenshot shows the 'Interactive Environments' list:

Project	Branch	Commit	Status	Action
dslab2020/dslab-exercises-week-1	master	eceba987	Running	<button>Connect</button> <input type="radio"/> Stop <input type="checkbox"/> Get logs

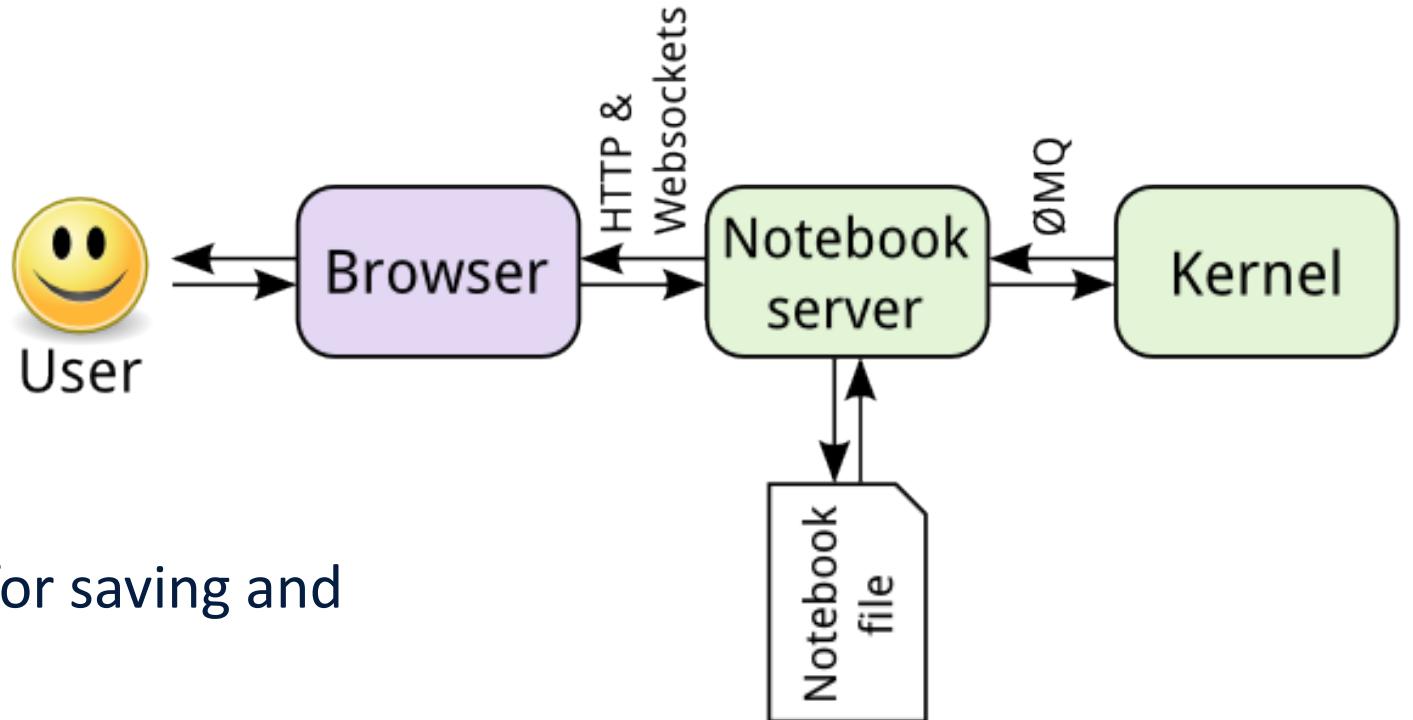
Overview Jupyter notebook



What is a Jupyter notebook?

- Open source web application
- Development started in 2014 as spin-off from IPython
- Notebooks denotes documents that contain both
 - Code, and rich text elements (figures, links, equations, etc)
- Notebook documents produced by the Jupyter Notebook App
 - Client-server application to edit and run notebooks
 - Two main components: the kernels and a dashboard
- Jupyter is a loose acronym meaning Julia, Python and R
- Nowadays, Jupyter supports many other languages

Jupyter is language-agnostic



- The **notebook server** is responsible for saving and loading notebooks
- The notebook is saved as a JSON file
- The **kernel** gets sent cells of code to execute

Crash-course in Python 3.x



Brief history of Python

- Open source programming language
- Conceived in the Netherlands by Guido van Rossum
- Implementation started in December 1989
- *Python 2.x is legacy, Python 3.x is the present & future of the language*
- Python 2.0 released on October 16, 2000 (current: 2.7.15)
- Python 3.0 released on December 3, 2008 (current: 3.7.3)
- High-level, portable, interpreted, object-oriented

Why Python?

- Easy, Fast & Broad
- Python Has a Healthy, Active and Supportive Community
 - Plenty of documentation, guides, tutorials and more
 - Incredibly active developer community
- Python Has Some Great Corporate Sponsors
 - Google!
- Python has Extensive Libraries

Python 3.x references

- <https://docs.python.org/3.7/tutorial/>
- Books include:
 - Learning Python by Mark Lutz
 - Python Essential Reference by David Beazley
 - Python for Data Analysis by William McKinney
 - <https://github.com/justmarkham/DAT8#python-resources>
- Python Quick Reference
 - <https://github.com/justmarkham/python-reference>

Common pitfalls in Python

1. Incorrect indentation, tabs and spaces
 - Never use tabs, only spaces (4)
2. Using a mutable value as a default value
 - ```
def foo(param=[]):
 param.append(10)
 return(param)
```
3. Scoping (local vs global variable)
  - ```
bar = 42
def foo():
    print(bar)
    bar = 0
foo()
```
4. Copying vs referencing (mutable types)
 - ```
a = [1, 2 ,3]
b = a
a[1] = 4

print(b)
```

# A few exercises in Python

---

- Open the **DSLab Week 1-1** Jupyter notebook

[./notebooks/DSLab week1 1.ipynb](#)

1. Generators
2. List comprehension
3. Lambda Operator and the functions map() and reduce()
4. Lists of characters and the functions join() and append()

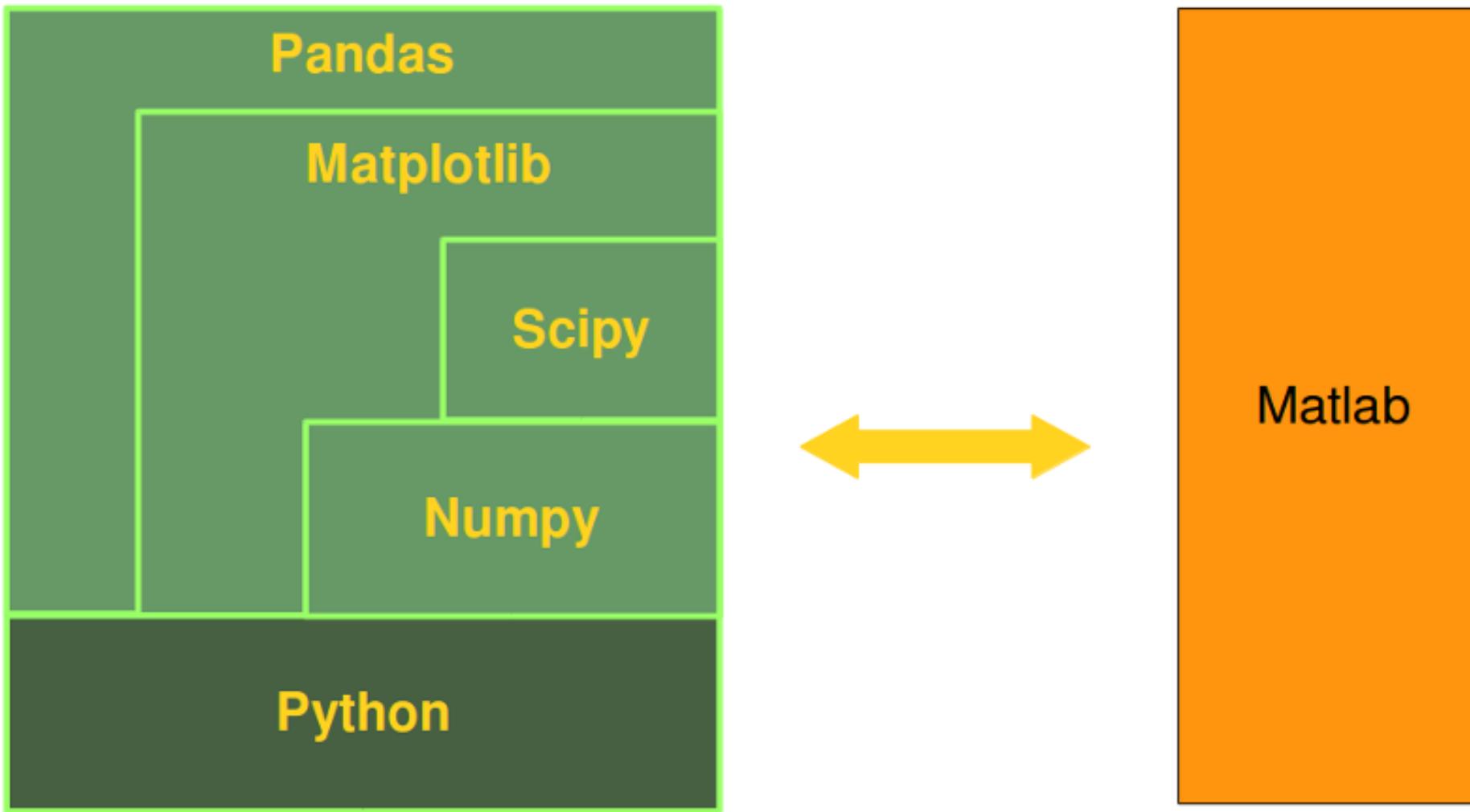
---

# NumPy, Pandas, scikit-learn & Matplotlib

---

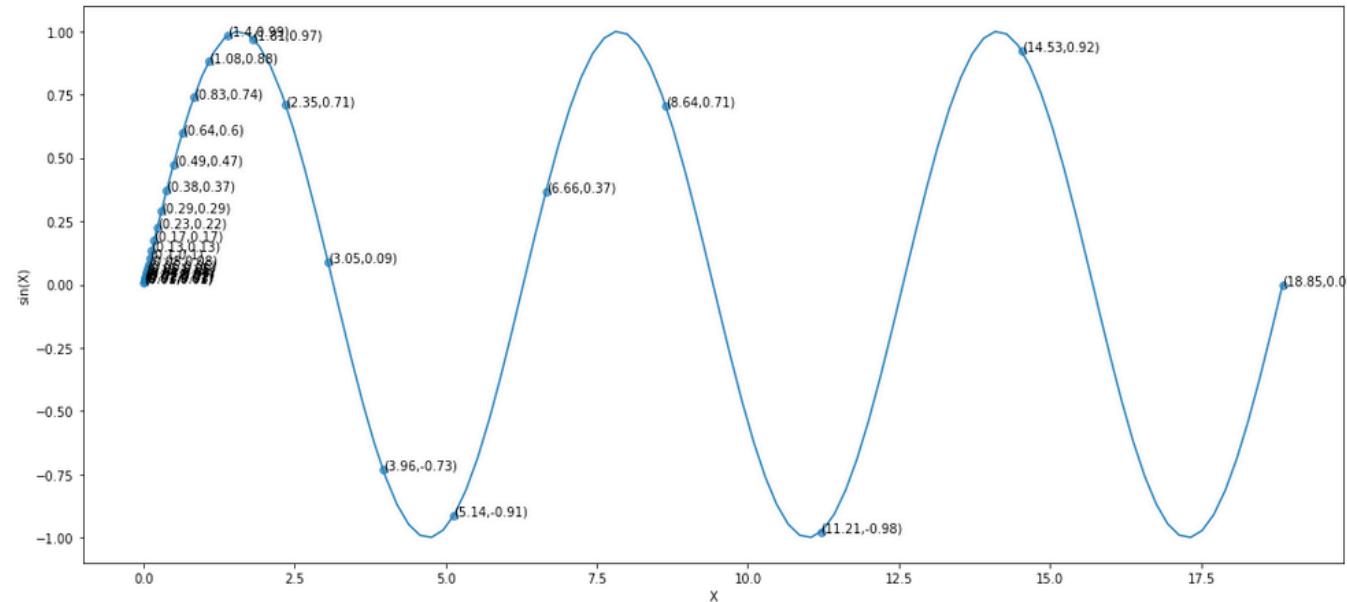
# Python alternative to Matlab

---



# Python libraries for (aspiring) data scientists

- **NumPy**: Work with large multidimensional arrays and matrices
  - **Pandas**: Do data wrangling
  - **Matplotlib**: Make line graphs, pie charts, histograms, etc
  - **Scikit-learn**: Build machine learning models



# NumPy & Data Types

---

- Core library for scientific computing in Python
- Provides a high-performance multidimensional array object, and tools for working with these arrays
- A **NumPy array** (`ndarray`) is a grid of values, all of the same type
  - indexed by a tuple of nonnegative integers
  - the *shape* of an array is the size of the array along each dimension

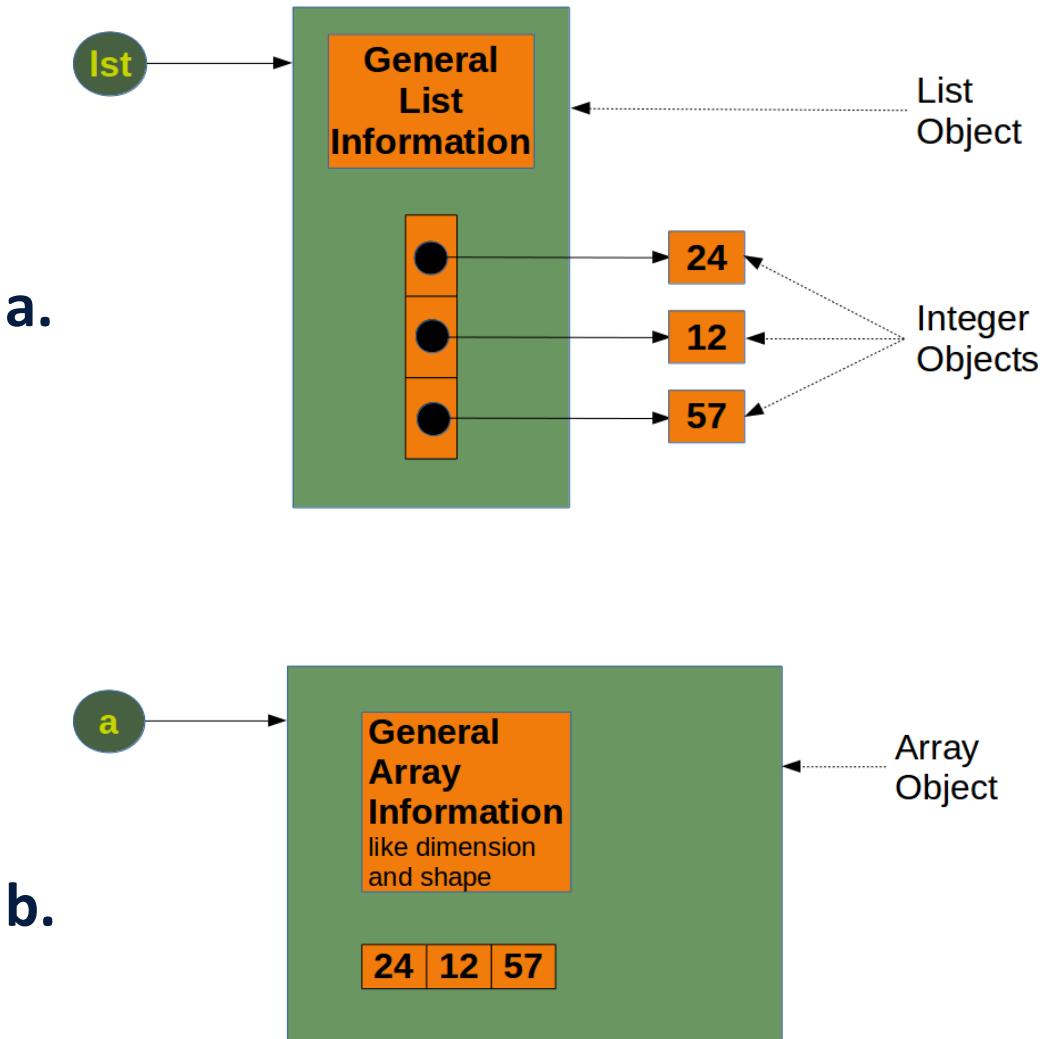
```
1 import numpy as np
2 a = np.array([[1,2,3], [4,5,6]])
3 print("Shape of matrix a: ", a.shape)
4 print("Value of a[0,2]:", a[0,2])
```

Shape of matrix a: (2, 3)

Value of a[0,2]: 3

# Python lists vs NumPy arrays

- 1. Functionality:** SciPy and NumPy have optimized functions such as linear algebra operations built in
- 2. Performance:** need for speed and are faster than lists
- 3. Size:** Numpy data structures take up less space
  - list:  $64 + n \cdot 8 + n \cdot 28$  bytes
  - ndarray:  $96 + n \cdot 8$  bytes



# Relevant NumPy functions

---

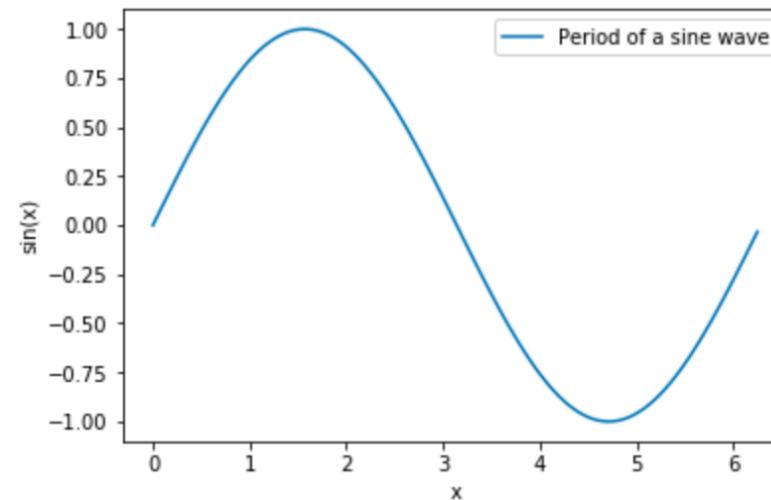
- **arange**([start,] stop[, step], [, dtype=None] )
  - **linspace**(start, stop, num=50, endpoint=True, retstep=False)
    - Checkout logspace & geomspace
  - **reshape**(array, newshape, order='C' )
  - **copy**(obj, order='K' )
- + Indexing and slicing arrays

# What is Matplotlib?

---

- Plotting library

```
1 import numpy as np
2 from matplotlib import pyplot as plt
3
4 %matplotlib inline
5
6 x = np.arange(0, 2*np.pi, 0.05, np.float32)
7 y = np.sin(x)
8
9 plt.plot(x,y)
10
11 plt.xlabel("x")
12 plt.ylabel("sin(x)")
13 plt.legend(['Period of a sine wave'])
14 plt.show()
15
```



# Relevant Matplotlib functions

- **Functions:** plot, subplot, scatter and annotate
- %matplotlib {inline, notebook}
- matplotlib.get\_backend()

# Other visualization libraries

---

- 1. Seaborn:** Python's Statistical Data Visualization Library
  - high-level interface to draw statistical graphics
  - <https://seaborn.pydata.org/>
  
- 2. Bokeh:** Python interactive visualization library
  - interactive plots, dashboards, and data applications
  - <https://bokeh.pydata.org/en/latest/>

# Pandas & Pandas Data Frames

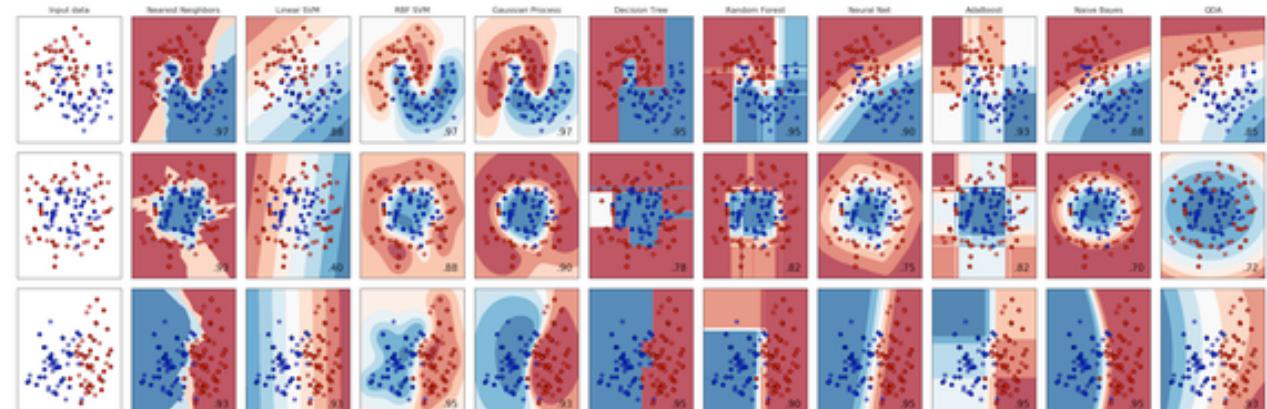
---

- Powerful & flexible data munging library
- Pandas was built on NumPy
  - NumPy stores your data in arrays
  - Pandas takes the NumPy Array...  
... and gives you a labeled index to it
- Pandas DataFrame is a 2-D labeled data structure with columns of potentially different types
  - Basically, dictionary-based NumPy arrays
- Recommended reading (before doing the exercises)
  - [https://chrisalbon.com/python/data\\_wrangling/pandas\\_apply\\_operations\\_to\\_groups/](https://chrisalbon.com/python/data_wrangling/pandas_apply_operations_to_groups/)

# Scikit-learn: Machine Learning in Python

---

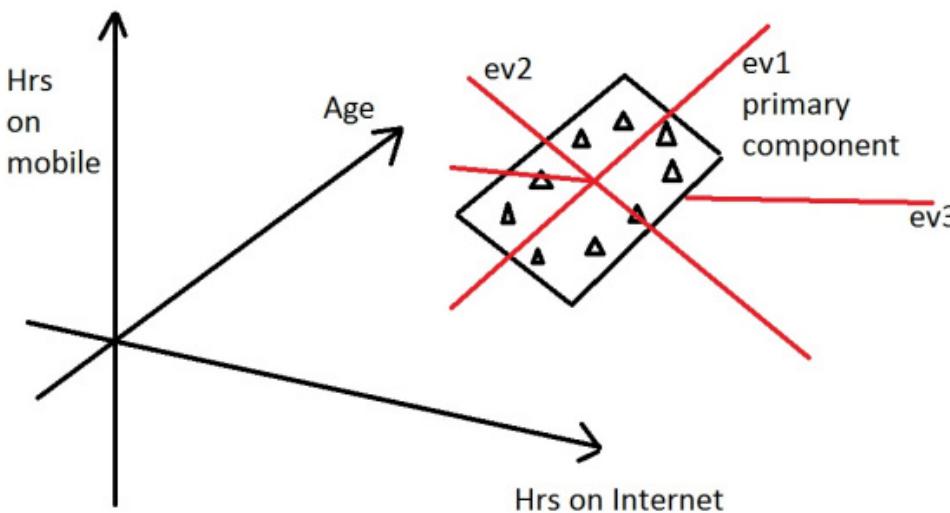
- <http://scikit-learn.org/stable/index.html>
  - Open source, built on NumPy, SciPy, and matplotlib
1. Classification
  2. Regression (Logistic Regression)
  3. Clustering (KMeans)
  4. Dimensionality reduction (PCA)
  5. Model selection
  6. Preprocessing



# Principal Component Analysis (PCA)

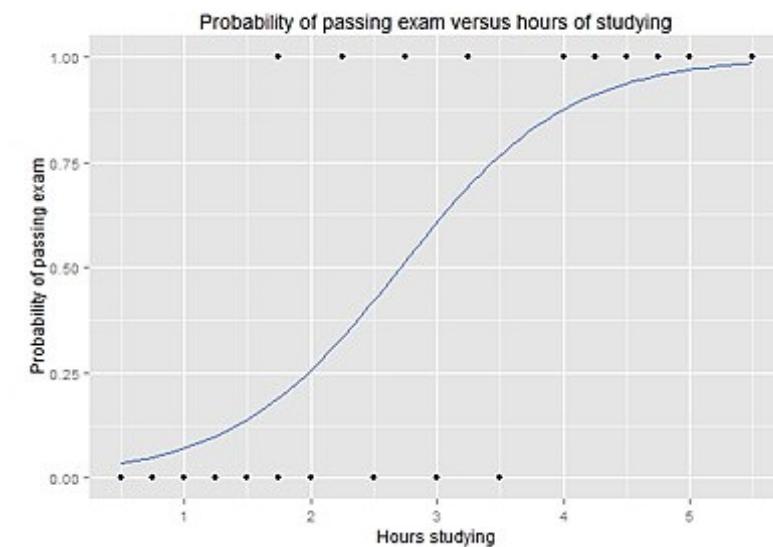
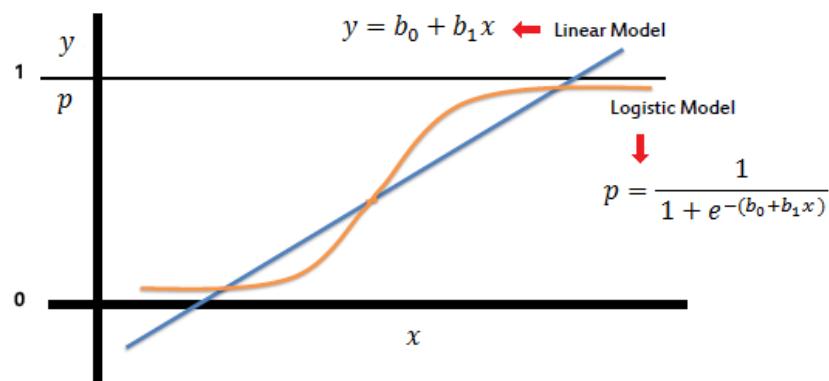
- sklearn.decomposition.PCA()
- (Linear) dimensionality reduction using SVD of the data to project it to a lower dimensional space
- Maximize the variance in the low-dimensional representation

- Take e.g. the
  - Example



# Logistic Regression

- sklearn.linear\_model.LogisticRegression()
- Model used for prediction of the probability of occurrence of an event by fitting data to a logistic curve
- Standard industry workhorse!

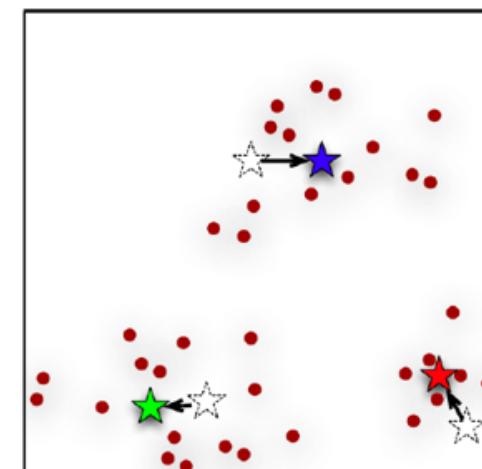
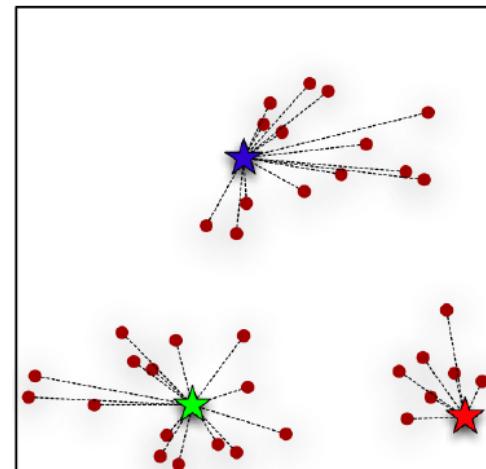


- Chaining a PCA and a logistic regression

# Kmeans – Clustering algorithm

---

- sklearn.cluster.KMeans()
- Find similarity groups in a data, called clusters
- Randomize K points, then two-step iterative algorithm:
  - 1. Cluster assignment step
  - 2. Move centroid step.



# **Exercises ... a bit harder/longer**

---

**Open DSLab Week 1 Jupyter notebooks in Renku**

1. `./notebooks/DSLab_week1-2.ipynb`
  - a. NumPy arrays & Matplotlib
2. `./notebooks/DSLab_week1-3.ipynb`
  - a. Getting familiar with Pandas
  - b. Simple Machine Learning problems

# Thank you!

