Anaconda and Jupyter Notebook

Neural Networks for Health Technology Applications

Spring 2020

Sakari Lukkarinen

Helsinki Metropolia University of Applied Sciences

The Environment

Python, Anaconda and Jupyter Notebook

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform

Download

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews

https://www.anaconda.com/distribution/

Windows | macOS | Linux

Get Started with Anaconda Distribution

Documentation
Installation and user guide for Anaconda Distribution 5

Read More

Anaconda Blog
News, software releases, and developer best practices

Read More

Community Support
Solutions and knowledge from the community

Read More

Anaconda Webinars
Industry trends and tutorials from Anaconda

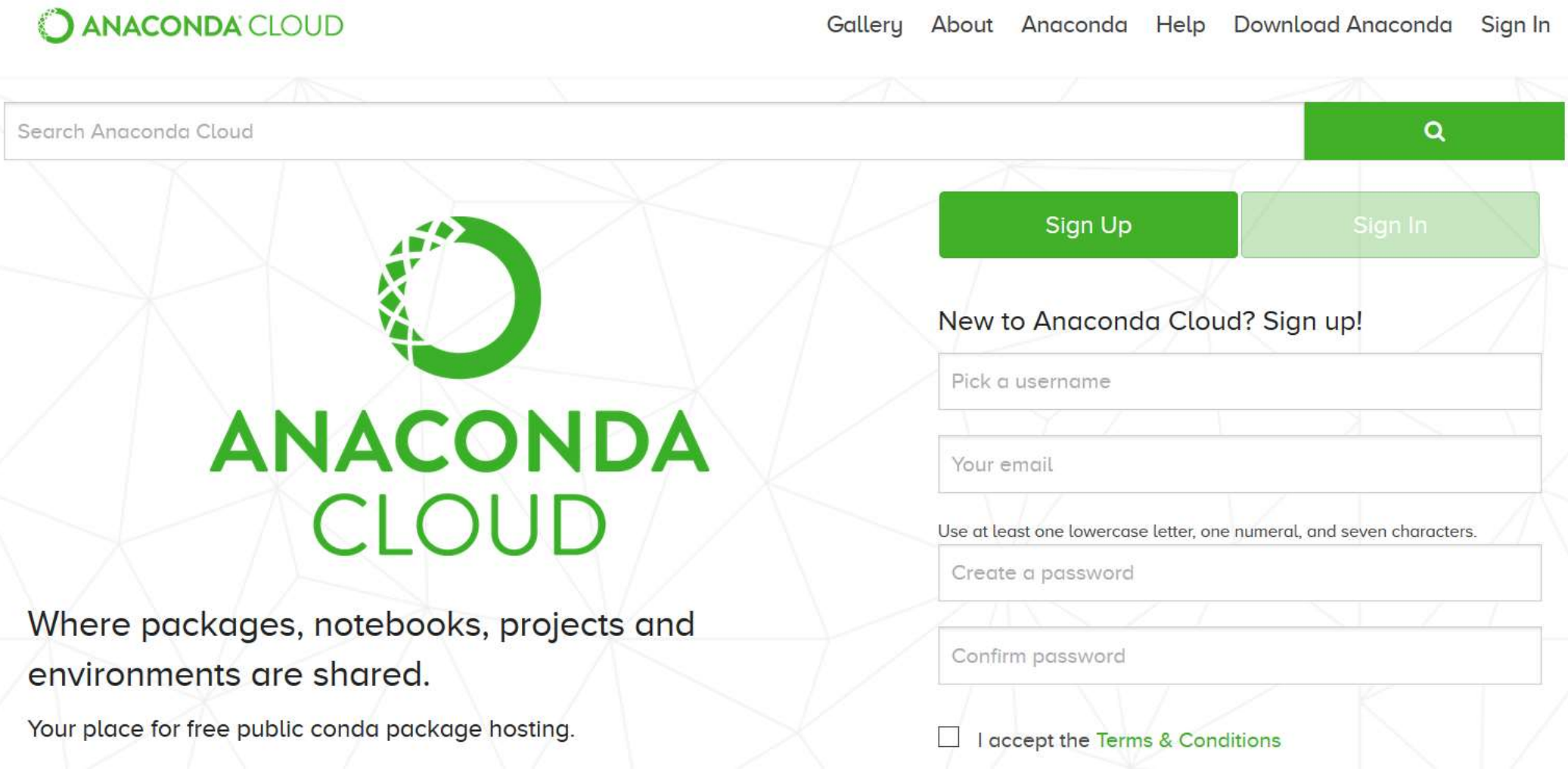
Read More

Anaconda Training
Learn Python for Data Science with DataCamp

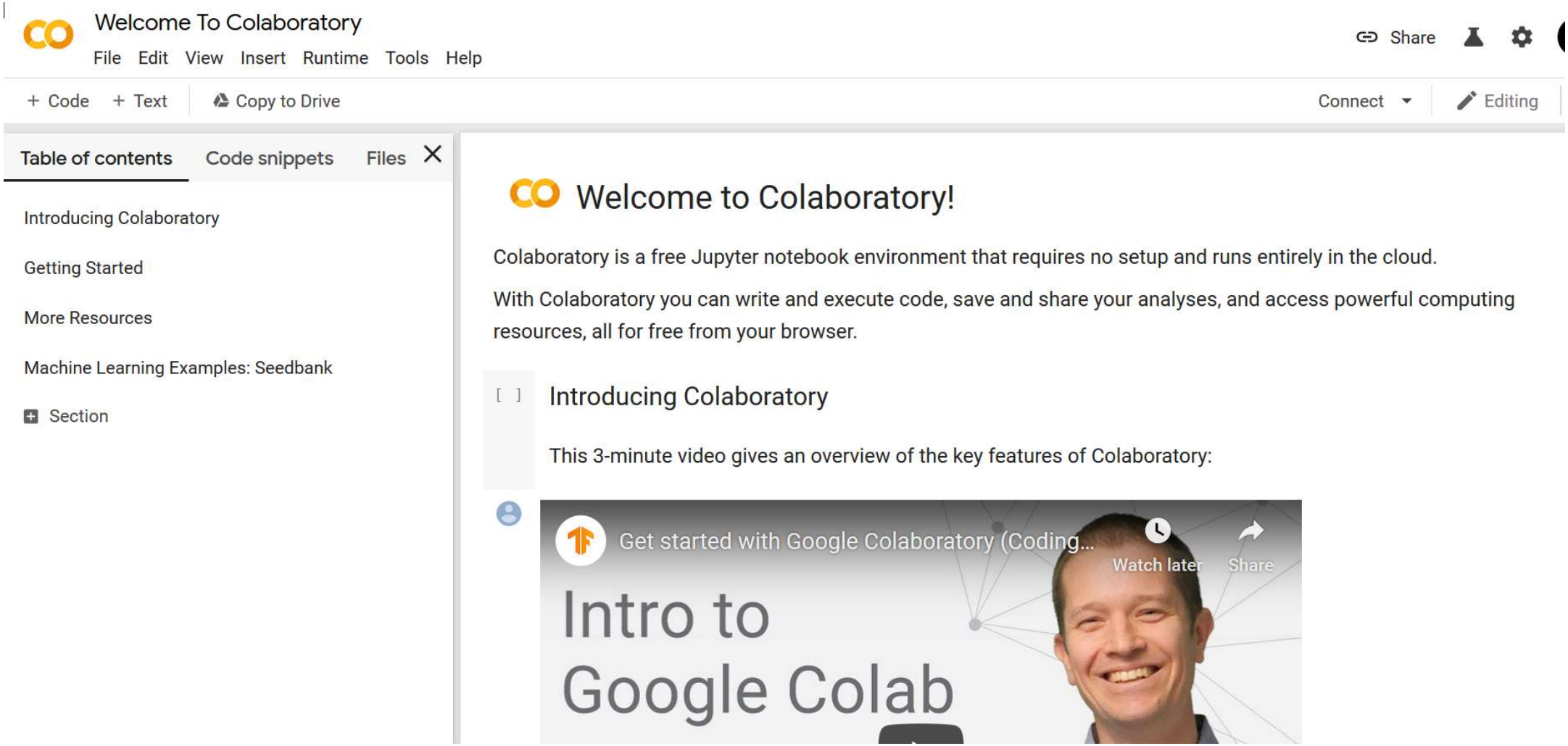
Start Learning

Jupyter Notebook cloud services

- Anaconda Cloud (Anaconda)
- Google colab (Google)
- Kaggle (Google)
- Azure (Microsoft)
- JupyterHub (Amazon)



https://anaconda.org/



https://colab.research.google.com/

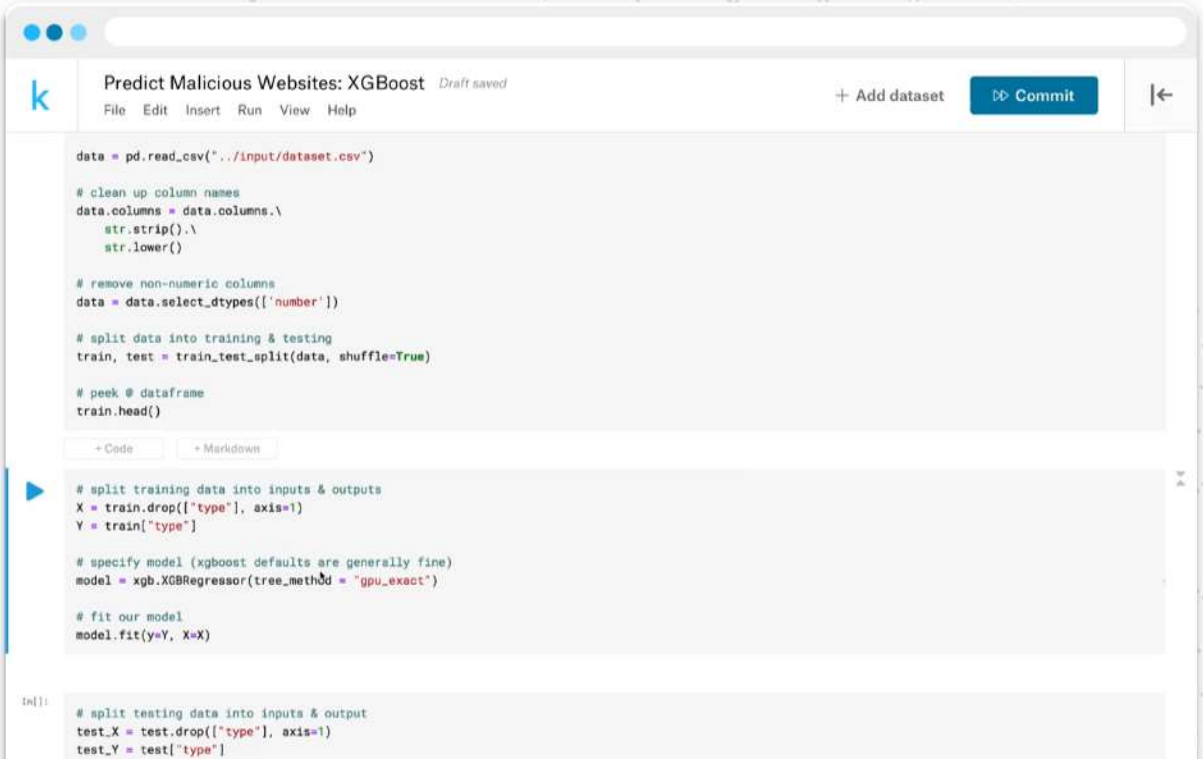
kaggle Search Competitions Datasets Notebooks Discussion Courses ••• Sign in

Start with more than a blinking cursor

Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access free GPUs and a huge repository of community published data & code.

REGISTER WITH GOOGLE

Register with Email



https://www.kaggle.com/

Develop and run code from anywhere with Jupyter notebooks on Azure.

Get started for free. Get a better experience with a free Azure Subscription.

TRY IT NOW >

https://notebooks.azure.com/

JupyterHub

PDF | Kindle | RSS

Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and narrative text. JupyterHub allows you to host multiple instances of a single-user Jupyter notebook server. When you create a cluster with JupyterHub, Amazon EMR creates a Docker container on the cluster's master node. JupyterHub, all the components required for Jupyter, and Sparkmagic run within the container.

Sparkmagic is a library of kernels that allows Jupyter notebooks to interact with Apache Spark running on Amazon EMR through Apache Livy, which is a REST server for Spark. Spark and Apache Livy are installed automatically when you create a cluster with JupyterHub. The default Python 3 kernel for Jupyter is available along with the PySpark 3, PySpark, SparkR, and Spark kernels that are available with Sparkmagic. You can use these kernels to run ad-hoc Spark code and interactive SQL queries using Python, R, and Scala. You can install additional kernels within the Docker container manually. For more information, see Installing Additional

https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-jupyterhub.html

Jupyter Notebook

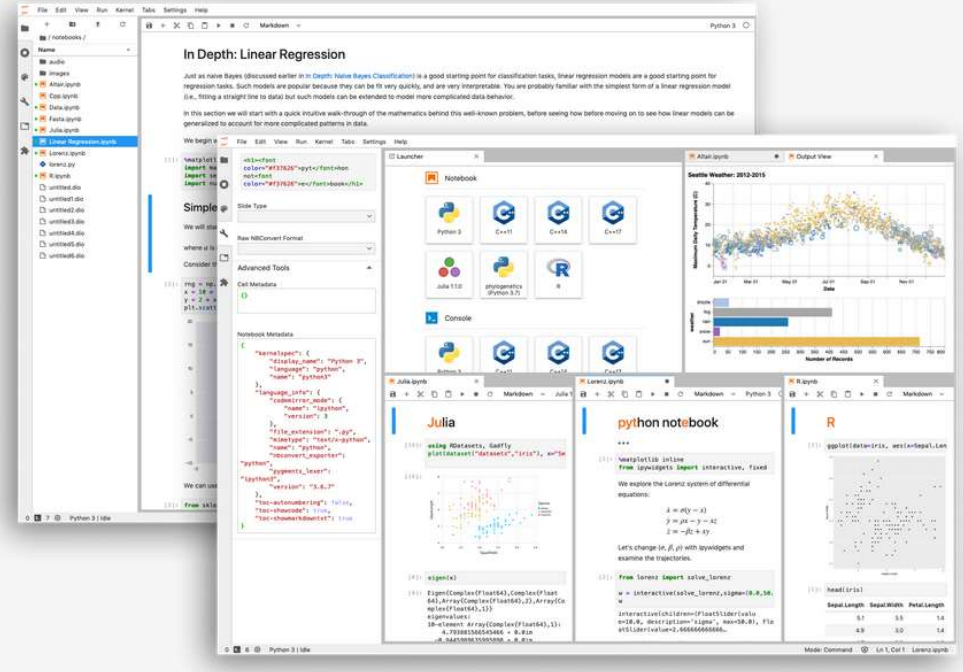
with Anaconda Distribution on own computer

jupyter

Install About Us Community Documentation NBViewer JupyterHub Widgets Blog

Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

https://jupyter.org/



JupyterLab 1.0: Jupyter's Next-Generation Notebook Interface

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

Try it in your browser

Install JupyterLab

Try Classic Notebook

A tutorial introducing basic features of Jupyter notebooks and the IPython kernel using the classic Jupyter Notebook interface.

Try JupyterLab

JupyterLab is the new interface for Jupyter notebooks and is ready for general use. Give it a try!

Try Jupyter with Julia

A basic example of using Jupyter with Julia.

Try Jupyter with R

A basic example of using Jupyter with R.

Try Jupyter with C++

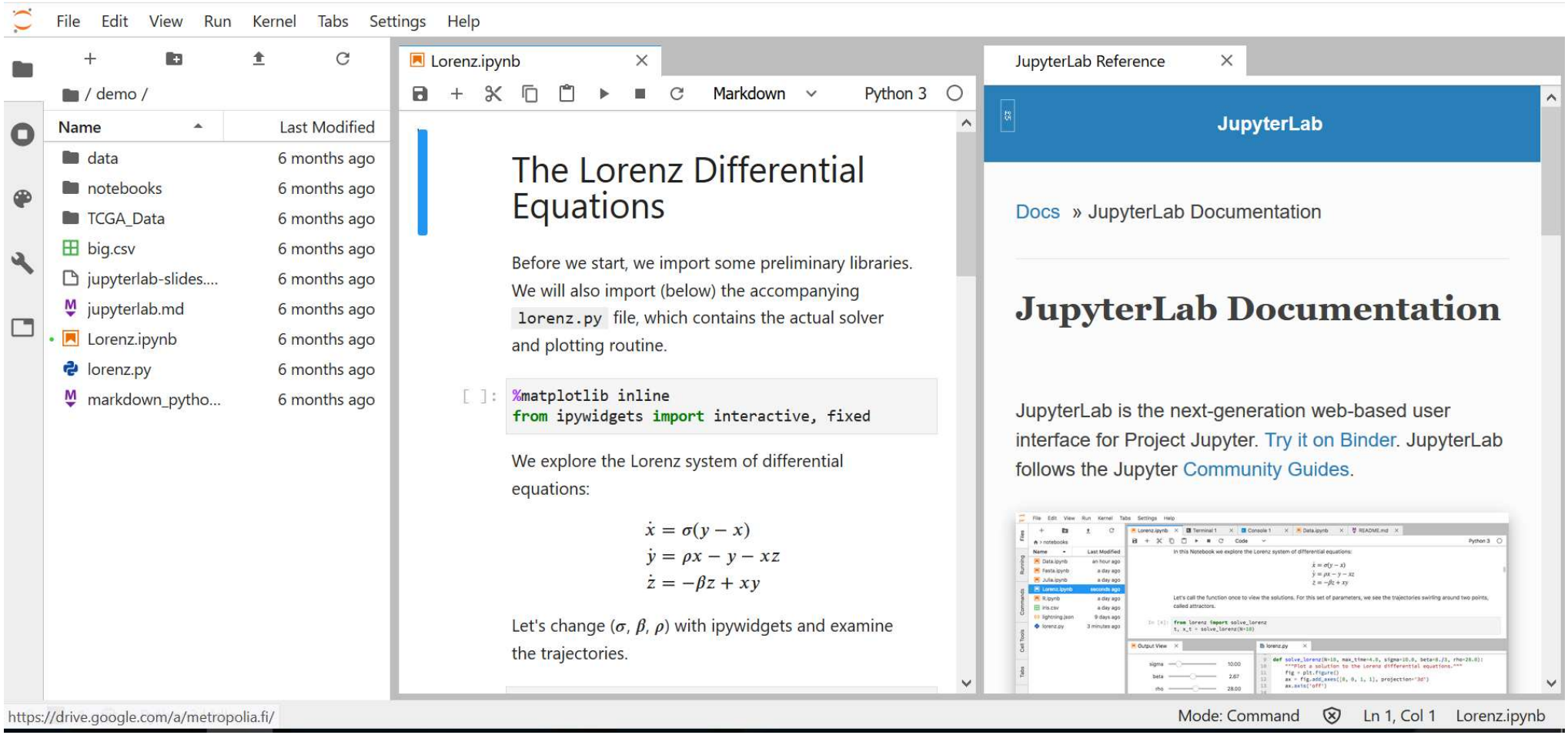
A basic example of using Jupyter with C++

Try Jupyter with Scheme

Explore the Calysto Scheme programming language, featuring integration with Python

https://jupyter.org/try

JupyterLab Interface



Classic Jupyter Browser interface

Classic Jupyter Notebook interface

jupyter Data (autosaved)

Visit repo | Copy Binder link

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

Open a CSV file using Pandas

In [4]:
```
import pandas
df = pandas.read_csv('../data/iris.csv')
df.head(20)
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	se
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
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa

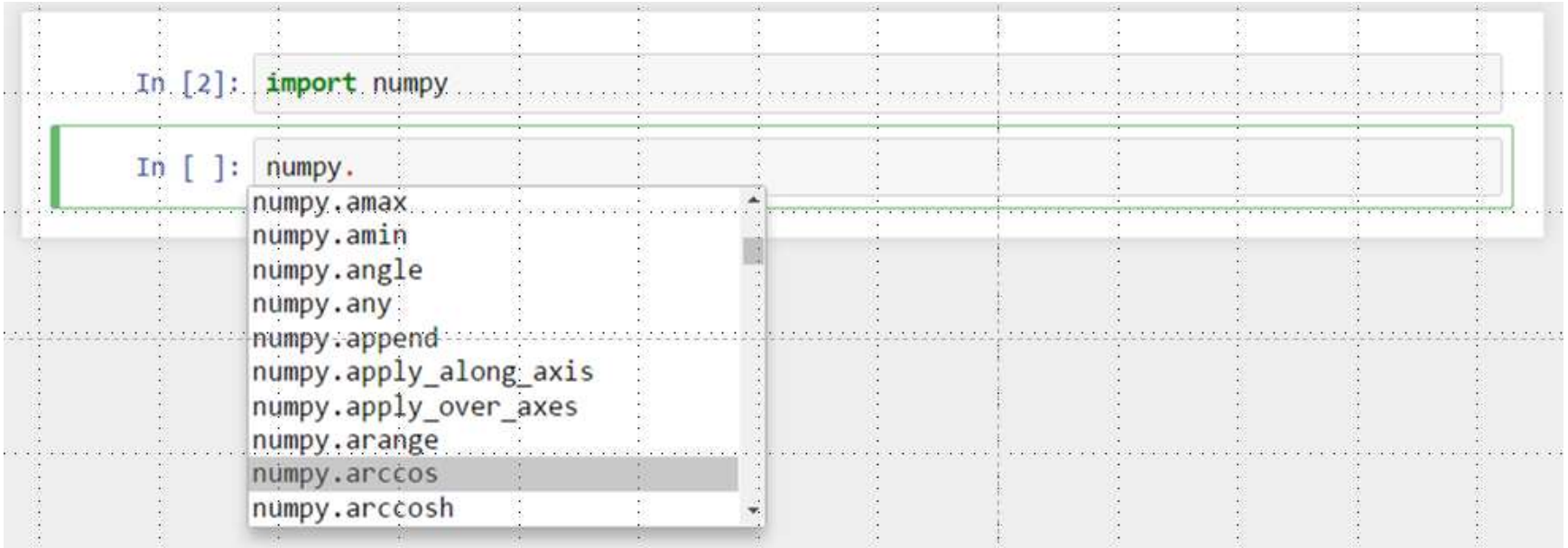
Behind the scene

Figure 1-2. *An overview of the components and layers in the scientific computing environment for Python, from a user's perspective, from top to bottom. Users typically only interact with the top three layers, but the bottom layer constitutes a very important part of the software stack. An example of specific software components from each layer in the stack is shown in the right part of the figure*

Notebook: Input and output cells

```
In [2]: import numpy

In [3]: 3*3
Out[3]: 9

In [4]: In[3]
Out[4]: '3*3'

In [5]: Out[3]-2
Out[5]: 7

In [ ]:
```

Autocompletion



Documentation

```
In [6]: numpy.cos?
```

```
Type:            ufunc
String form:     <ufunc 'cos'>
File:            c:\anaconda3\lib\site-packages\numpy\__init__.py
Docstring:
cos(x[, out])

Cosine element-wise.

Parameters
----------
x : array_like
    Input array in radians.
out : ndarray, optional
    Output array of same shape as `x`.
```

Tip: Fastest way to find information is to use Google with proper keywords, try for example: numpy cosine

numpy cosine

Web Kuvat Videot Uutiset Kartat Asetukset

Suomi Turvahaku: Kohtuullinen Milloin tahansa

numpy.cos — NumPy v1.17 Manual - SciPy.org — SciPy.org

https://docs.scipy.org/doc/numpy/reference/generated/numpy.cos.html

Notes. If out is provided, the function writes the result into it, and returns a reference to out. (See Examples) References. M. Abramowitz and I. A. Stegun, Handbook ...

numpy.cos — NumPy v1.13 Manual - SciPy.org — SciPy.org

https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.cos.html

A location into which the result is stored. If provided, it must have a shape that the inputs broadcast to. If not provided or None, a freshly-allocated array is ...

numpy.cos — NumPy v1.19.dev0 Manual

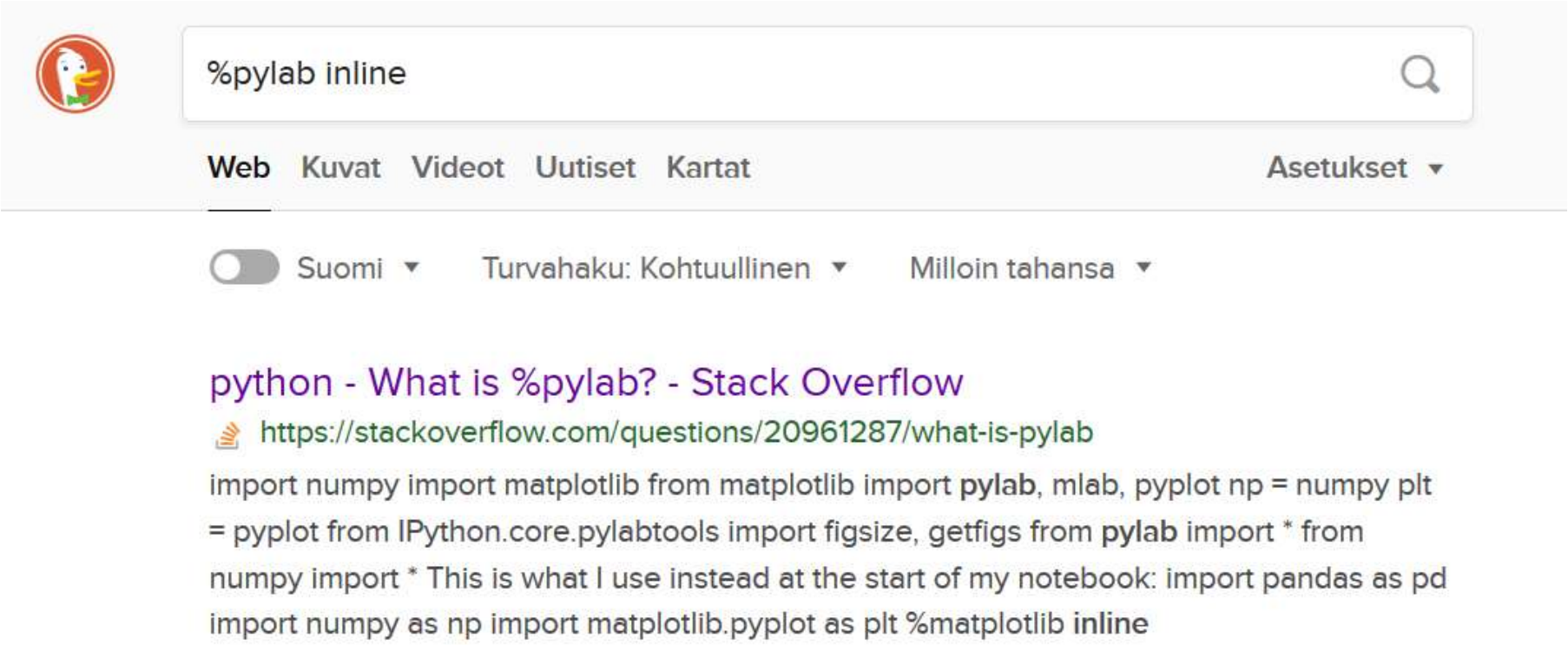
https://numpy.org/devdocs/reference/generated/numpy.cos.html

The corresponding **cosine** values. This is a scalar if x is a scalar. Notes. If out is provided, the function writes the result into it, and returns a reference to out. (See Examples) References. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions. New York, NY: Dover, 1972. Examples

Magic commands

```
In [7]: %pylab inline

        Populating the interactive namespace from numpy and matplotlib

In [8]: %lsmagic

Out[8]: Available line magics:
        %alias  %alias_magic  %autocall  %automagic  %autosave  %bookmark  %cd  %clear
        %cls  %colors  %config  %connect_info  %copy  %ddir  %debug  %dhist  %dirs  %do
        ctest_mode  %echo  %ed  %edit  %env  %gui  %hist  %history  %install_default_co
        nfig  %install_ext  %install_profiles  %killbgscripts  %ldir  %less  %load  %lo
        ad_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %logstop  %ls  %lsmagic
          %macro  %magic  %matplotlib  %mkdir  %more  %notebook  %page  %pastebin  %pdb
          %pdef  %pdoc  %pfile  %pinfo  %pinfo2  %popd  %pprint  %precision  %profile
         %prun  %psearch  %psource  %pushd  %pwd  %pycat  %pylab  %qtconsole  %quickref
          %recall  %rehashx  %reload_ext  %ren  %rep  %rerun  %reset  %reset_selective
        %rmdir  %run  %save  %sc  %set_env  %store  %sx  %system  %tb  %time  %timeit
         %unalias  %unload_ext  %who  %who_ls  %whos  %xdel  %xmode

        Available cell magics:
        %%!  %%HTML  %%SVG  %%bash  %%capture  %%cmd  %%debug  %%file  %%html  %%javasc
        ript  %%latex  %%perl  %%prun  %%pypy  %%python  %%python2  %%python3  %%ruby
         %%script  %%sh  %%svg  %%sx  %%system  %%time  %%timeit  %%writefile

        Automagic is ON, % prefix IS NOT needed for line magics.
```

Most important magic command

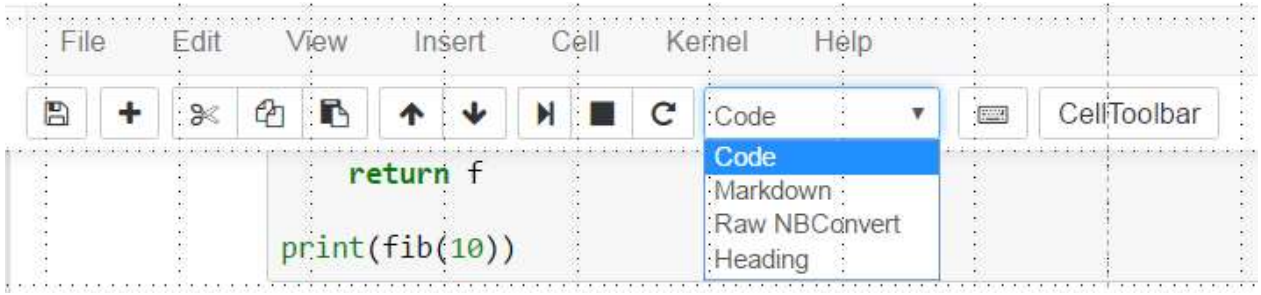


https://stackoverflow.com/questions/20961287/what-is-pylab

Defining functions (Python)

```
In [10]: def fib(n):
             """
             Return a list of Fibonacci numbers
             """
             f0, f1 = 0, 1
             f = [1] * n

             # Loop from 1 to n-1
             for i in range(1, n):
                 f[i] = f0 + f1
                 f0, f1 = f1, f[i]

             return f

         print(fib(10))

         [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Cell types



Code

Any Python code. Press Shift+Enter to send the code to the kernel. The results are sent back to the browser

Markdown

Contains marked-up plain text, which is interpreted using Markdown Language and HTML (and Latex)

Raw

A raw text cell, displayed without any interpretation

Headings

Heading cell, from level 1 to 6 (#, ##, ###, ….)

Markdown cells

Function	Syntax by example
Italics	`*text*`
Bold	`**text**`
Strike-through	`~~text~~`
Fixed-width font	`` `text` ``
URL	`[URL text](http://www.example.com)`
New paragraph	Separate the text of two paragraphs with an empty line.

Verbatim	Lines that start with four blank spaces are displayed as-is, without any further processing, using a fixed-width font. This is useful for code-like text segments.
	`␣␣␣␣def func(x):`
	`␣␣␣␣ return x ** 2`

Table	`| A | B | C |`
	`|---|---|---|`
	`| 1 | 2 | 3 |`
	`| 4 | 5 | 6 |`

Horizontal line	A line containing three dashes is rendered as a horizontal line separator: `---`

Heading	`# Level 1 heading`
	`## Level 2 heading`
	`### Level 3 heading`
	`...`

Block quote	`Lines that start with a '>' are rendered as a block quote.`
	`> Text here is indented and offset`
	`> from the main text body.`

Unordered list	`* Item one`
	`* Item two`
	`* Item three`

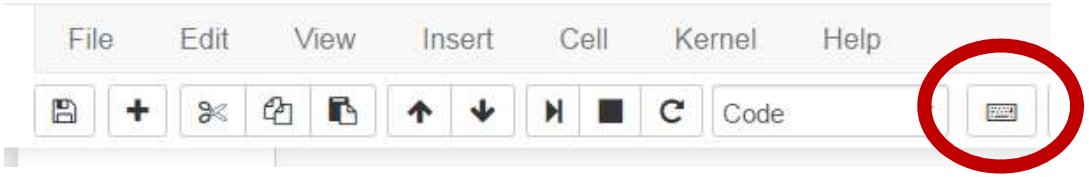
Markdown cells (continued)

Function	Syntax by example

Ordered list	`1. Item one`
	`2. Item two`
	`3. Item three`

Image	`![Alternative text](image-file.png)`[9] or `![Alternative text](http://www.example.com/image.png)`
Inline LaTeX equation	`$\LaTeX$`
Displayed LaTeX equation (centered, and on a new line)	`$$\LaTeX$$` or `\begin{env}...\end{env}` where `env` can be a LaTeX environment such as `equation`, `eqnarray`, `align`, etc.

Keyboard shortcuts



a – Create new cell above

b – Create new cell below

c – Copy cell

x – Cut cell

v – Paste cell

m – Convert to markdown

y – Convert to code

h – Help

s – Save notebook

i – i - Interrupt kernel

0 – 0 – Restart kernel

ENTER – Enter edit mode

Esc – Exit edit mode

UP – Previous cell

DOWN – Next cell

Exercises

See OMA:

- Class Exercise 1. Getting familiar with Jupyter Notebook