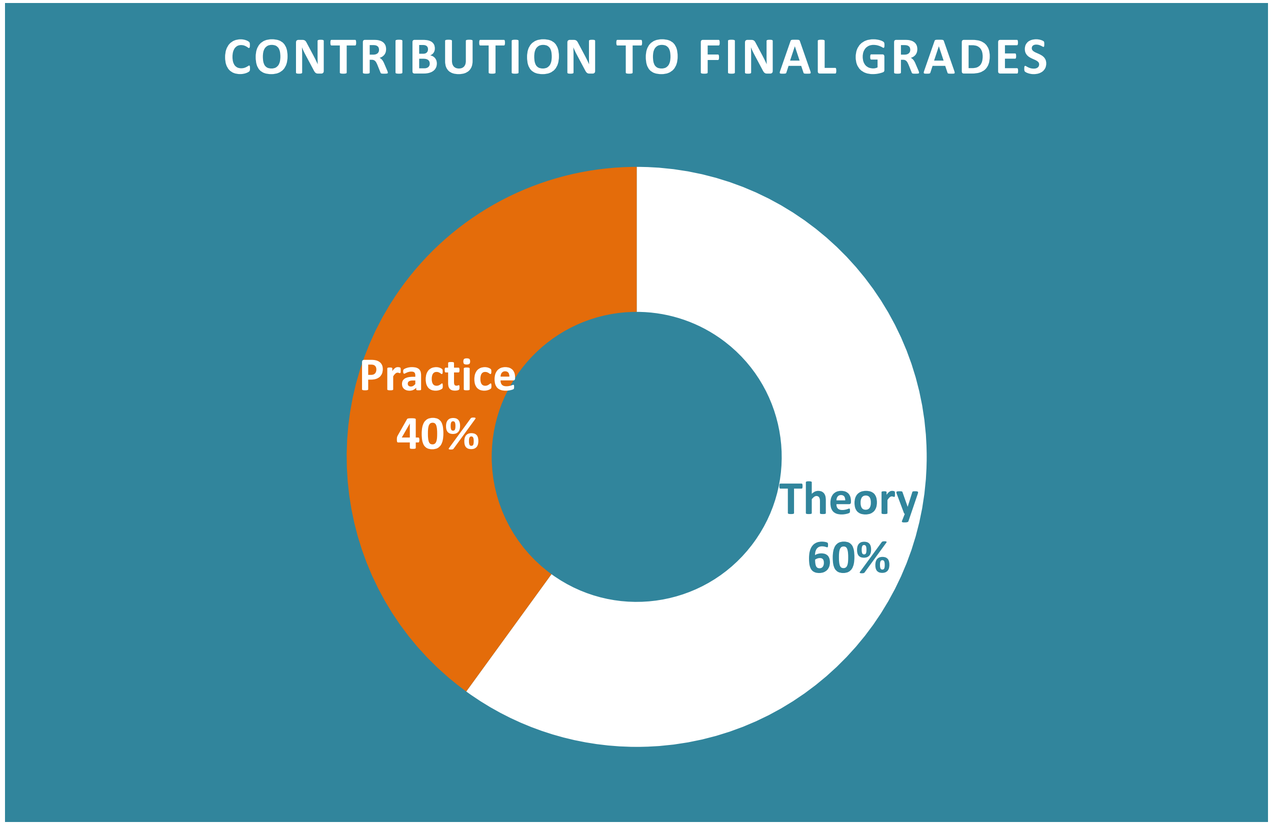
Introduction to Practical exercises workflow

José **Raúl** Ruiz Sarmiento

Content

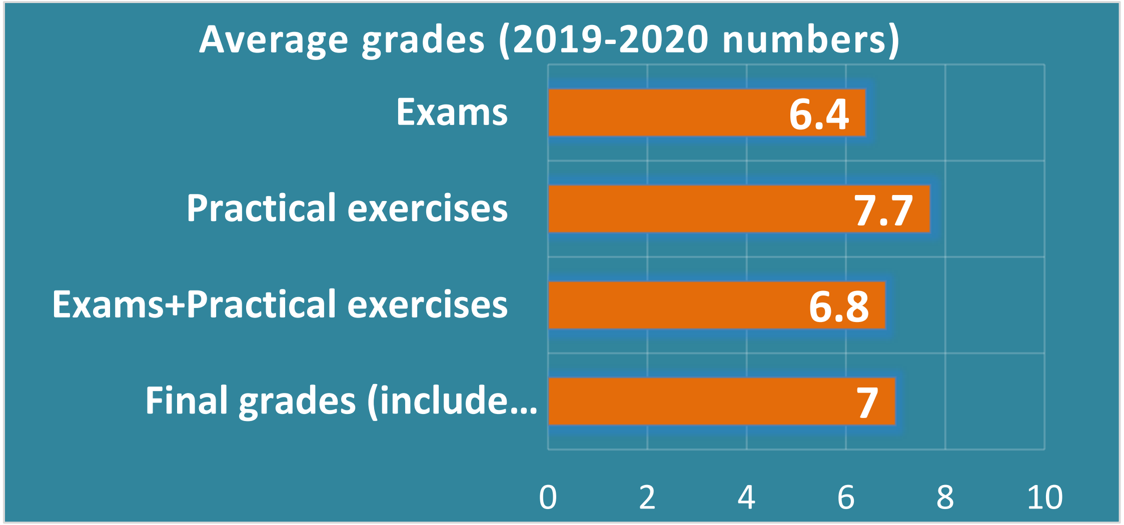
1. Practical exercises
   - Weight
   - Relevance
   - Workflow
2. The tool: Jupyter Notebook
   - Components.
   - How to use.
   - Live demo.

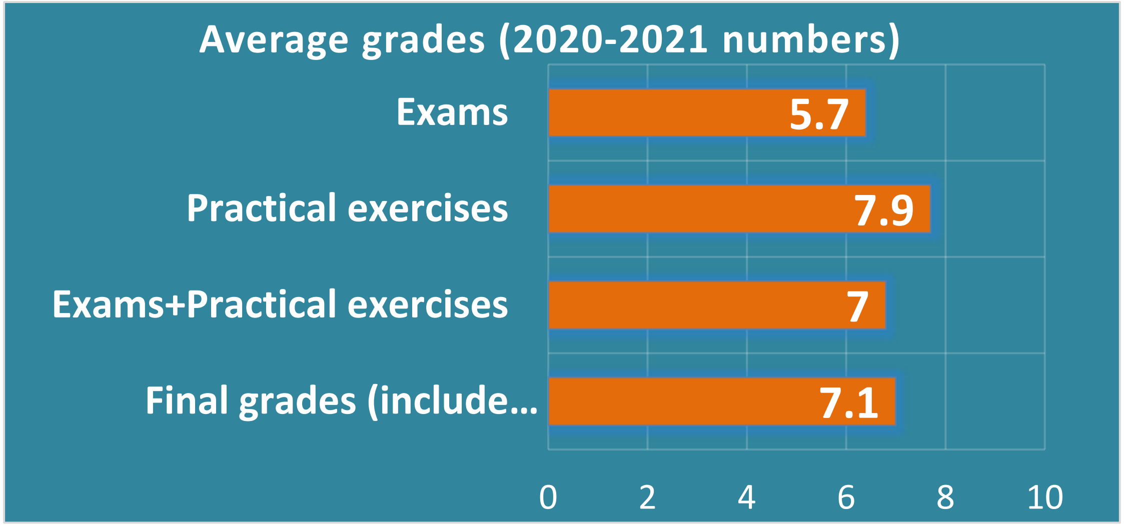
1. Practical exercises: weight



- Once the subject is passed, other factors can **increase** the grades:
  - Extra/optional exercises.
  - Participation in lessons/forums.
  - Etc.

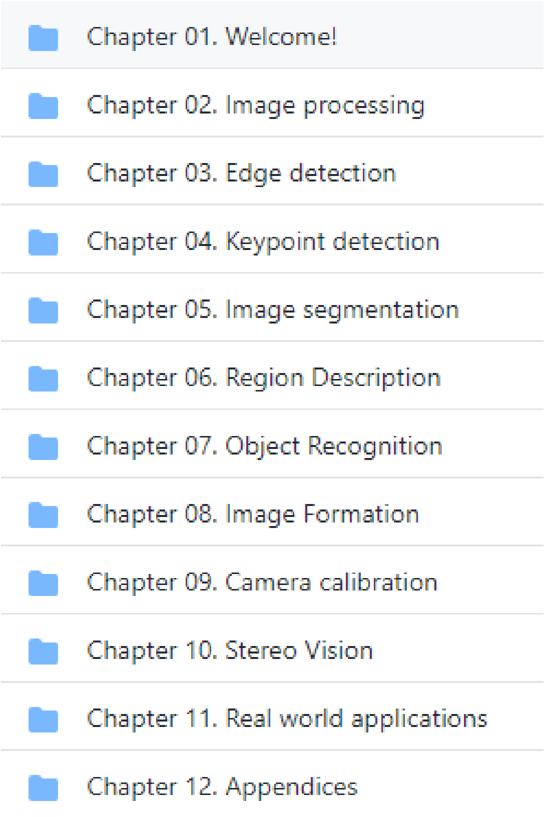
1. Practical exercises: relevance





1. Practical exercises: workflow

- Are organized in **chapters**.
- Each chapter corresponds to a theory lecture:



Chapter 01. Welcome!

Chapter 02. Image processing

Chapter 03. Edge detection

Chapter 04. Keypoint detection

Chapter 05. Image segmentation

Chapter 06. Region Description

Chapter 07. Object Recognition

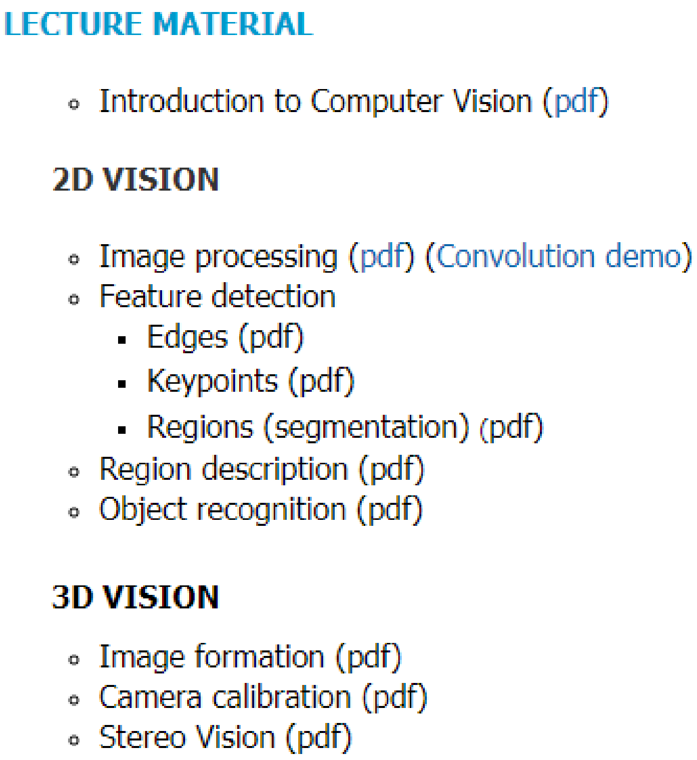
Chapter 08. Image Formation

Chapter 09. Camera calibration

Chapter 10. Stereo Vision

Chapter 11. Real world applications

Chapter 12. Appendices



LECTURE MATERIAL

- Introduction to Computer Vision (pdf)

2D VISION

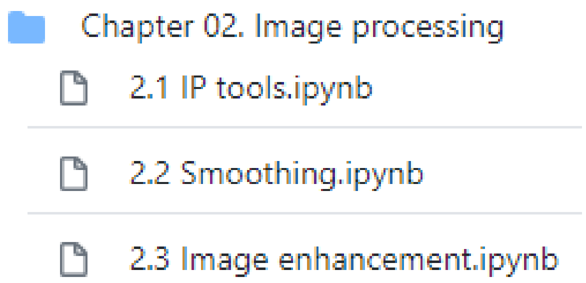
- Image processing (pdf) (Convolution demo)
- Feature detection
  - Edges (pdf)
  - Keypoints (pdf)
  - Regions (segmentation) (pdf)
- Region description (pdf)
- Object recognition (pdf)

3D VISION

- Image formation (pdf)
- Camera calibration (pdf)
- Stereo Vision (pdf)

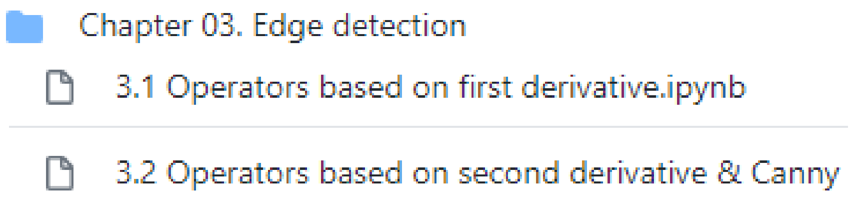
1. Practical exercises: workflow

Each chapter consists of a number of **notebooks**



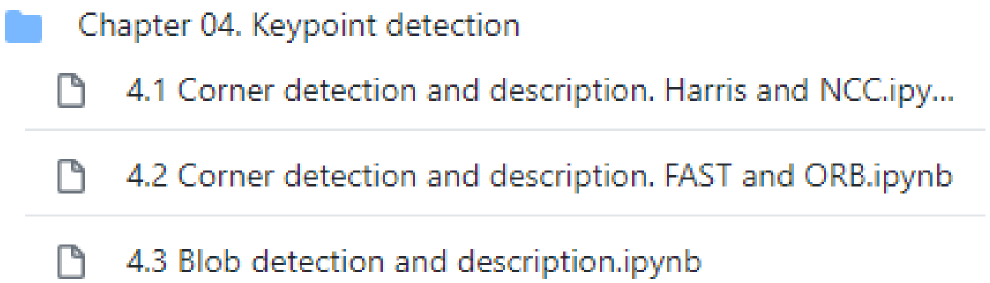
Chapter 02. Image processing

- 2.1 IP tools.ipynb
- 2.2 Smoothing.ipynb
- 2.3 Image enhancement.ipynb



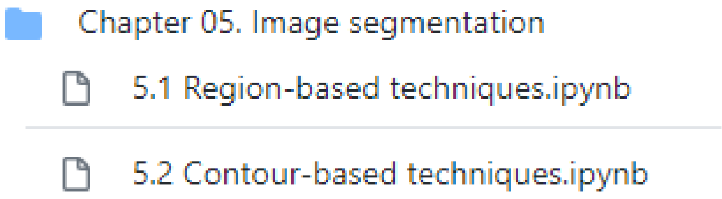
Chapter 03. Edge detection

- 3.1 Operators based on first derivative.ipynb
- 3.2 Operators based on second derivative & Canny



Chapter 04. Keypoint detection

- 4.1 Corner detection and description. Harris and NCC.ipy...
- 4.2 Corner detection and description. FAST and ORB.ipynb
- 4.3 Blob detection and description.ipynb

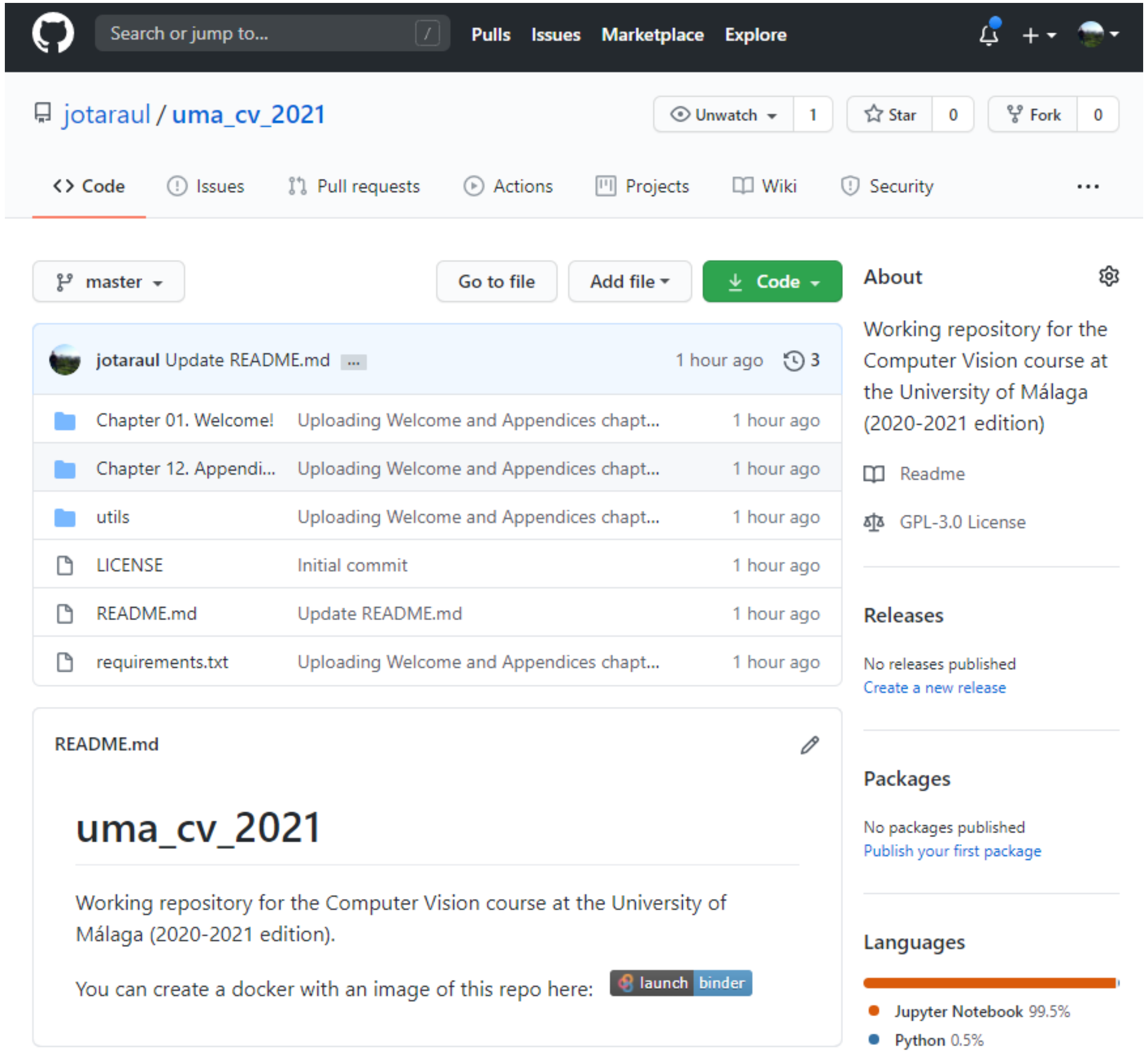


Chapter 05. Image segmentation

- 5.1 Region-based techniques.ipynb
- 5.2 Contour-based techniques.ipynb

1. Practical exercises: workflow

- Notebooks are provided to the student through a GitHub repository: **uma_cv_2022**
- URL: https://github.com/jotaraul/uma_cv_2022



jotaraul Update README.md ... 1 hour ago 3

Chapter 01. Welcome!	Uploading Welcome and Appendices chapt...	1 hour ago
Chapter 12. Appendi...	Uploading Welcome and Appendices chapt...	1 hour ago
utils	Uploading Welcome and Appendices chapt...	1 hour ago
LICENSE	Initial commit	1 hour ago
README.md	Update README.md	1 hour ago
requirements.txt	Uploading Welcome and Appendices chapt...	1 hour ago

README.md

uma_cv_2021

Working repository for the Computer Vision course at the University of Málaga (2020-2021 edition).

You can create a docker with an image of this repo here: launch binder

About

Working repository for the Computer Vision course at the University of Málaga (2020-2021 edition)

Readme

GPL-3.0 License

Releases

No releases published
Create a new release

Packages

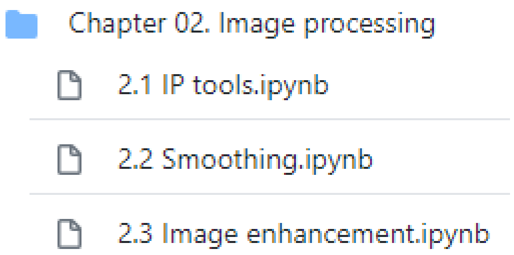
No packages published
Publish your first package

Languages

Jupyter Notebook 99.5%
Python 0.5%

1. Practical exercises: workflow

- After completing the notebooks corresponding to a chapter, the student **must submit a *.pdf* file** containing **the result of executing those notebooks**.



Chapter 02. Image processing

2.1 IP tools.ipynb

2.2 Smoothing.ipynb

2.3 Image enhancement.ipynb

- A **workshop** will be enabled for that in the *Campus Virtual*.

Workshop Chapter 6

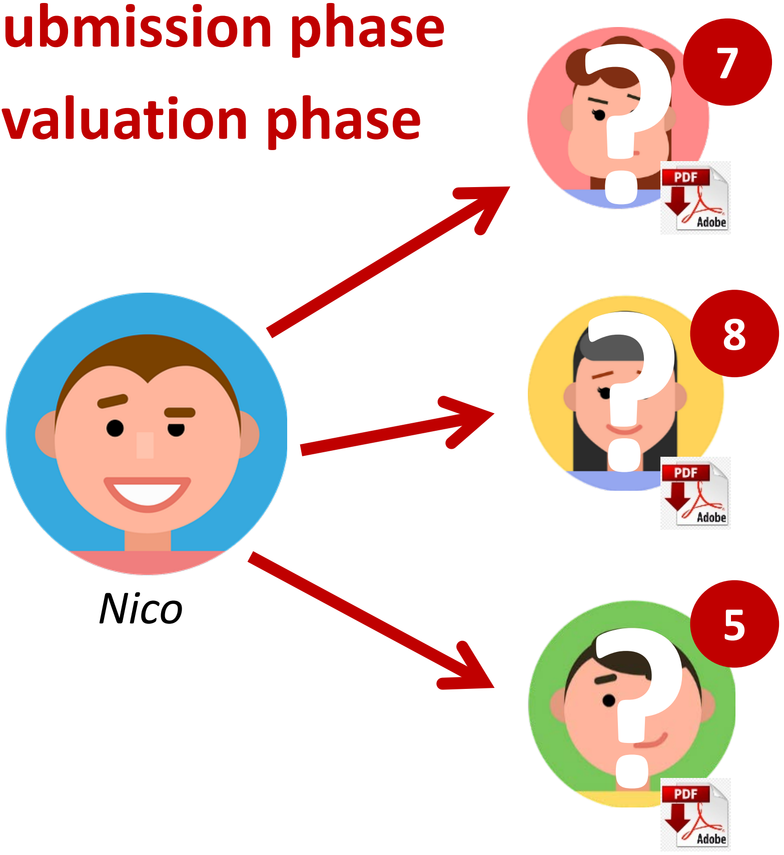
- Such submissions will have a **hard deadline**.
  - Submissions beyond this deadline will not be accepted.

1. Practical exercises: workflow

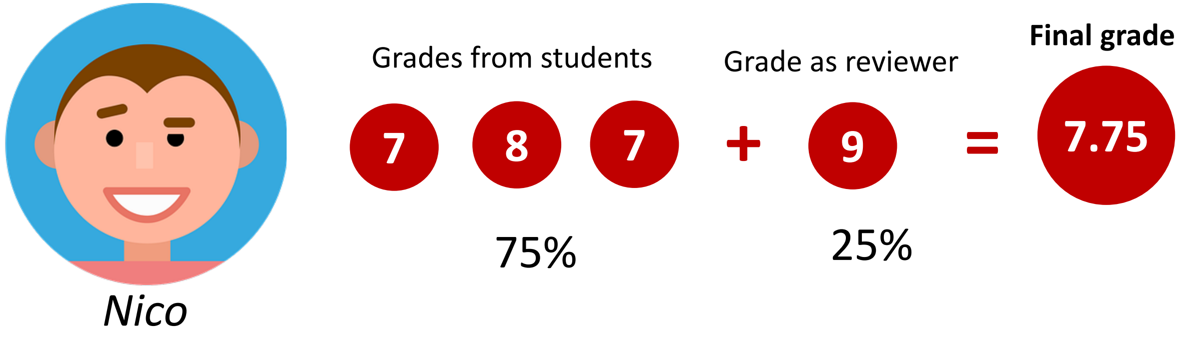
- **Evaluation** will be done through the **Workshop (taller)** activity of the *Campus Virtual*. This implements a **peer review** process.

1. Submission phase

2. Evaluation phase



3. Grading phase



1. Practical exercises: workflow

- **Advantages** of workshops:
  - Provides students with insight into the evaluation criteria.
  - Clarify the requirements for producing work of a particular standard.
  - Provides students with a degree of ownership of the assessment process.
  - Encourages them to reflect on the quality of their work.
  - Discourages poor practices that may be more apparent to a marker than the original writer.
  - Fosters the development of generic skills such as:
    - Critical appraisal,
    - An ability to provide colleagues with objective feedback on their work.

Harris, Judy R. "Peer assessment in large undergraduate classes: an evaluation of a procedure for marking laboratory reports and a review of related practices." Advances in physiology education 35.2 (2011): 178-187.

Al-Khalifa, Amal, K., and Marie Devlin. "Evaluating a Peer Assessment Approach in Introductory Programming Courses." United Kingdom & Ireland Computing Education Research conference. 2020.

Dolezal, Dominik, et al. "Person-centered learning using peer review method–an evaluation and a concept for student-centered classrooms." (2018): 127-147.

1. Practical exercises: workflow summary

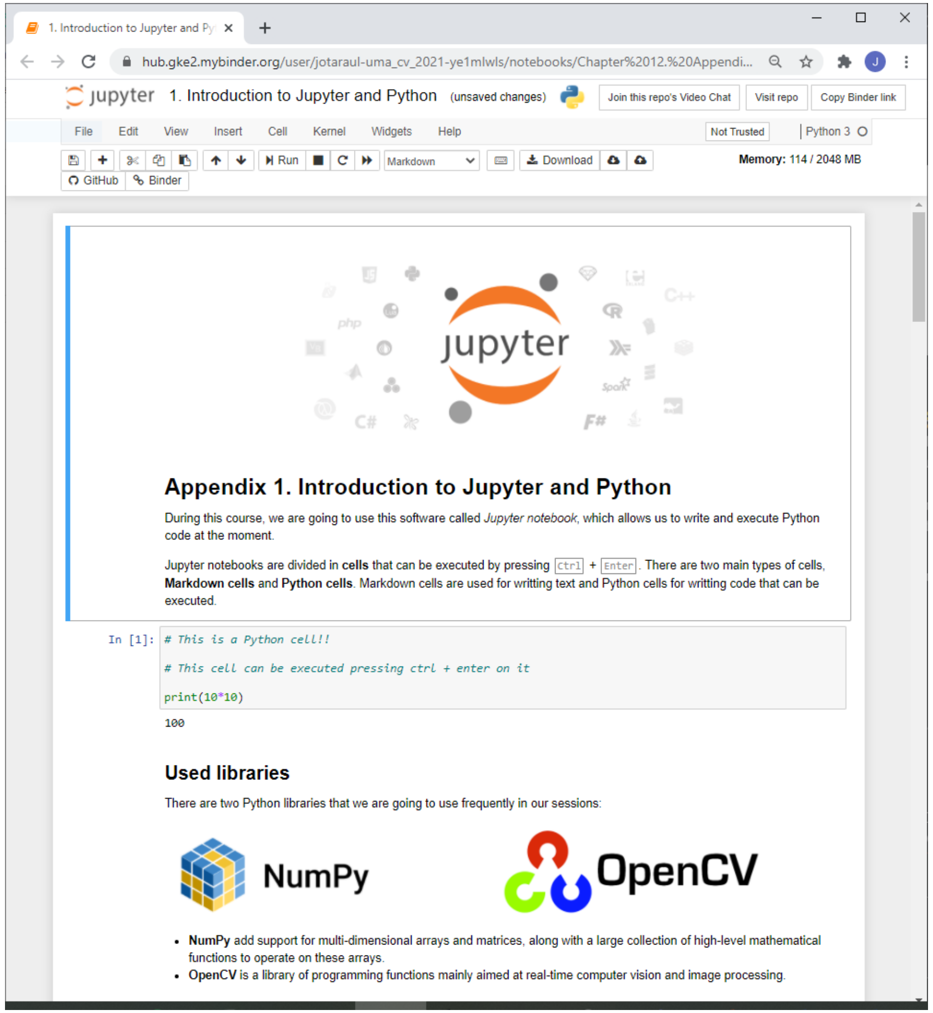
1. Complete notebooks of each chapter.
2. Submit there before deadline (as a unique *.pdf* file).
3. Review classmates’ work (workshop activity).
4. Get grades.

Final grade of practical exercises:

Average of grades achieved at each chapter.

2. The tool: Jupyter Notebook

- An **open-source web application** that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
- Brief history:
  - Created as an evolution of IPython by the Colombian Fernando Pérez in **2014**, in the umbrella of **Project Jupyter**.
  - In 2015, GitHub and Project Jupyter designed the *.ipynb* format.



1. Introduction to Jupyter and Py
hub.gke2.mybinder.org/user/jotaraul-uma_cv_2021-ye1mlwls/notebooks/Chapter%2012.%20Appendi...
jupyter
1. Introduction to Jupyter and Python
(unsaved changes)
Join this repo's Video Chat
Visit repo
Copy Binder link
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Markdown
Download
Memory: 114 / 2048 MB
GitHub
Binder
jupyter
NumPy
OpenCV

Appendix 1. Introduction to Jupyter and Python

During this course, we are going to use this software called *Jupyter notebook*, which allows us to write and execute Python code at the moment.

Jupyter notebooks are divided in **cells** that can be executed by pressing `Ctrl` + `Enter`. There are two main types of cells, **Markdown cells** and **Python cells**. Markdown cells are used for writting text and Python cells for writting code that can be executed.

```
In [1]: # This is a Python cell!!

        # This cell can be executed pressing ctrl + enter on it

        print(10*10)

        100
```

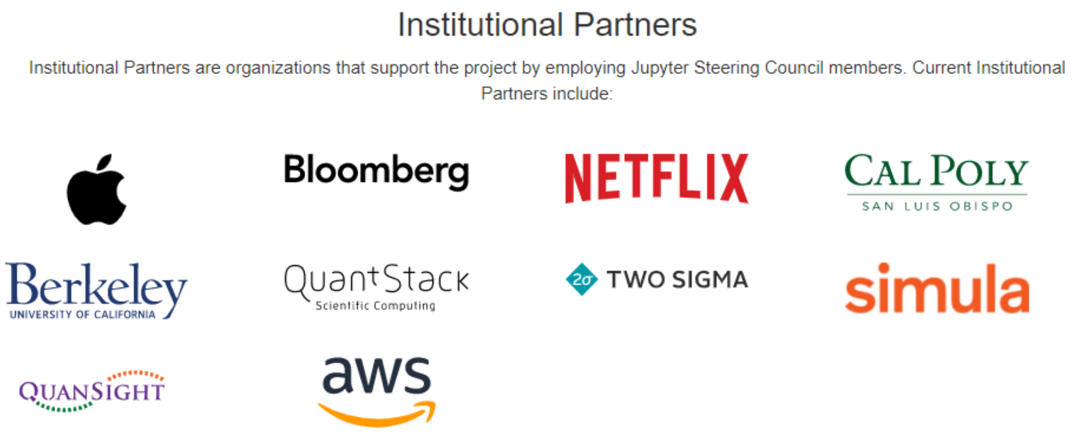
Used libraries

There are two Python libraries that we are going to use frequently in our sessions:

- **NumPy** add support for multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **OpenCV** is a library of programming functions mainly aimed at real-time computer vision and image processing.

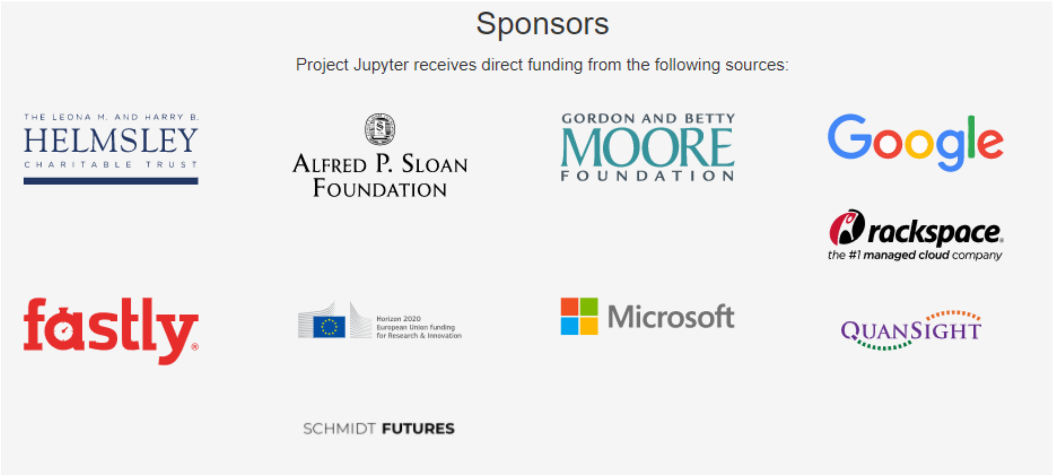
2. The tool: Jupyter Notebook

- GitHub public Jupyter notebooks:
  - 2015: 200K
  - 2018: 2,5M
- Powerful combinations:
- Supported by companies/institutions.



Institutional Partners

Institutional Partners are organizations that support the project by employing Jupyter Steering Council members. Current Institutional Partners include:



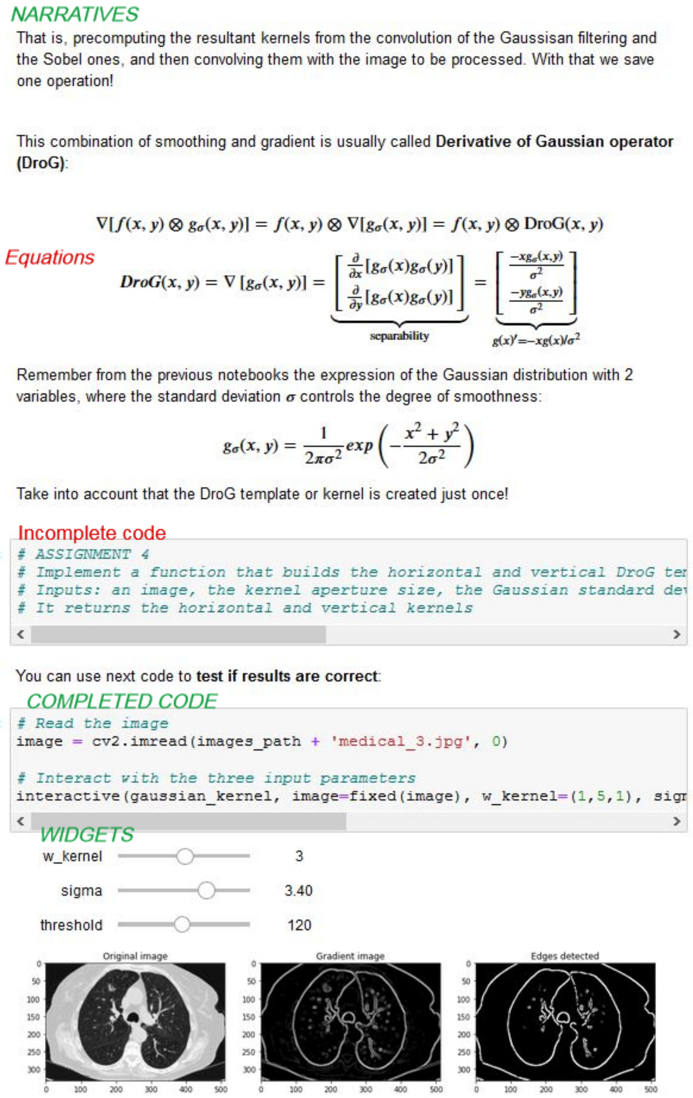
Sponsors

Project Jupyter receives direct funding from the following sources:

THE LEONA M. AND HARRY B. HELMSLEY CHARITABLE TRUST
ALFRED P. SLOAN FOUNDATION
GORDON AND BETTY MOORE FOUNDATION
Google
rackspace
the #1 managed cloud company
fastly
Horizon 2020 European Union Funding for Research & Innovation
Microsoft
QUANSIGHT
SCHMIDT FUTURES

2. Jupyter Notebook: components

- **Notebook**
  - Text format used to store the interactive documents (*json*).
  - It is composed of **cells**.
  - **Text cells** (markdown):
    - Theoretical concepts.
    - Equations.
    - Images.
    - Videos.
    - HTML components.
  - **Code cell:** executable cell that produces some computation, typically returning and printing results.



NARRATIVES

That is, precomputing the resultant kernels from the convolution of the Gaussisan filtering and the Sobel ones, and then convolving them with the image to be processed. With that we save one operation!

This combination of smoothing and gradient is usually called **Derivative of Gaussian operator (DroG)**:

$$\nabla[f(x,y) \otimes g_\sigma(x,y)] = f(x,y) \otimes \nabla[g_\sigma(x,y)] = f(x,y) \otimes \mathrm{DroG}(x,y)$$

Equations

$$DroG(x,y) = \nabla\,[g_\sigma(x,y)] = \underbrace{\begin{bmatrix} \frac{\partial}{\partial x}[g_\sigma(x)g_\sigma(y)] \\ \frac{\partial}{\partial y}[g_\sigma(x)g_\sigma(y)] \end{bmatrix}}_{\text{separability}} = \underbrace{\begin{bmatrix} \frac{-xg_\sigma(x,y)}{\sigma^2} \\ \frac{-yg_\sigma(x,y)}{\sigma^2} \end{bmatrix}}_{g(x)'=-xg(x)/\sigma^2}$$

Remember from the previous notebooks the expression of the Gaussian distribution with 2 variables, where the standard deviation σ controls the degree of smoothness:

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Take into account that the DroG template or kernel is created just once!

Incomplete code

```
# ASSIGNMENT 4
# Implement a function that builds the horizontal and vertical DroG ter
# Inputs: an image, the kernel aperture size, the Gaussian standard dev
# It returns the horizontal and vertical kernels
```

You can use next code to **test if results are correct**:

COMPLETED CODE

```
# Read the image
image = cv2.imread(images_path + 'medical_3.jpg', 0)

# Interact with the three input parameters
interactive(gaussian_kernel, image=fixed(image), w_kernel=(1,5,1), sigr
```

WIDGETS

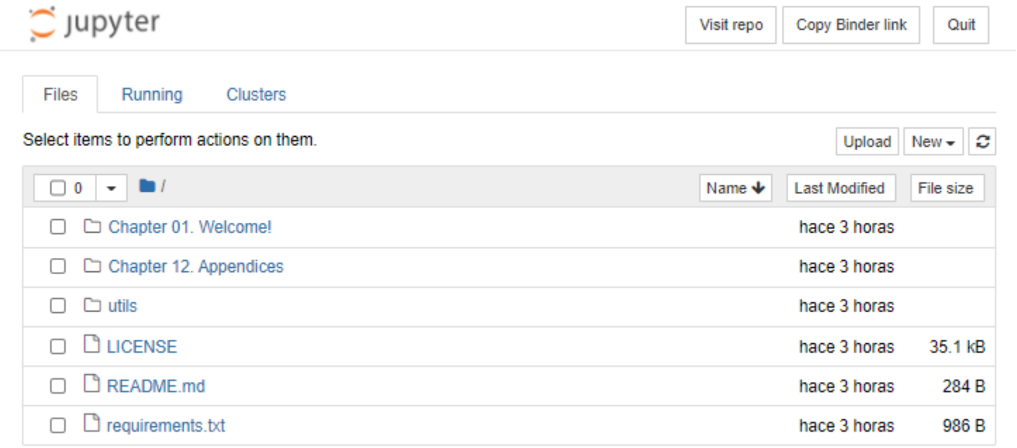
w_kernel 3

sigma 3.40

threshold 120

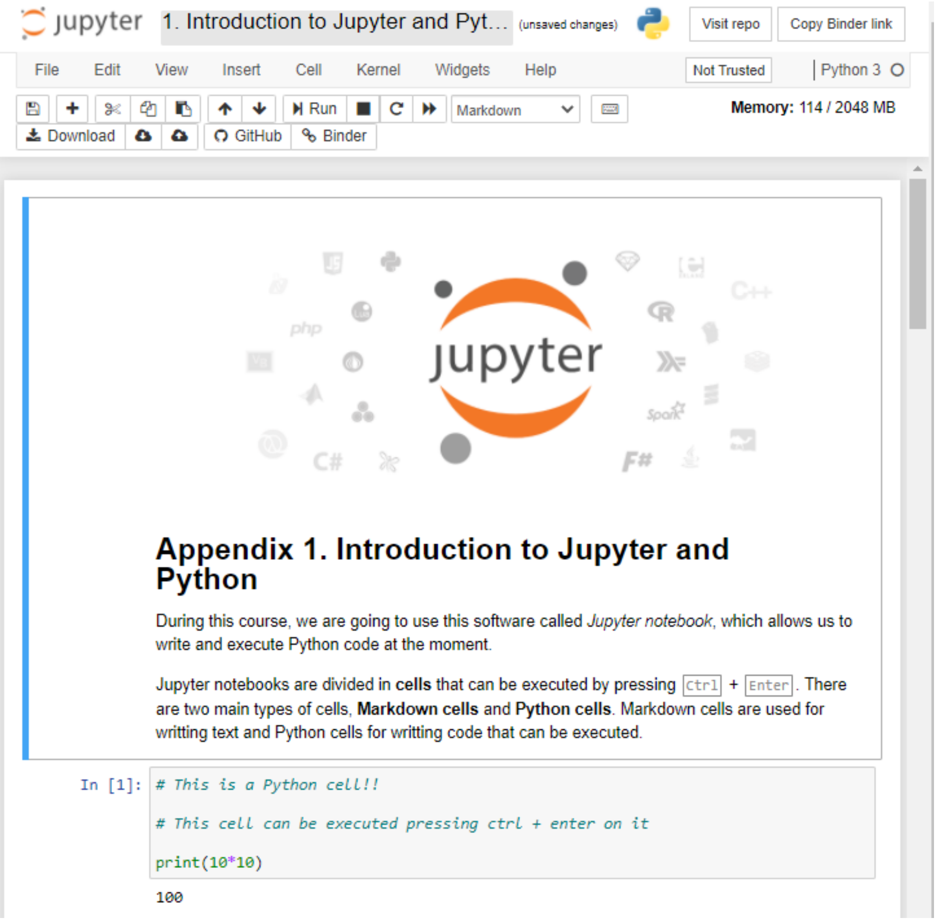
2. Jupyter Notebook: components

- **Web application**
  - Permits us to create, edit and run notebooks.
  - Requirements:
    - An installation of jupyter (local or remote).
    - A web browser.



Select items to perform actions on them.

Name	Last Modified	File size
Chapter 01. Welcome!	hace 3 horas	
Chapter 12. Appendices	hace 3 horas	
utils	hace 3 horas	
LICENSE	hace 3 horas	35.1 kB
README.md	hace 3 horas	284 B
requirements.txt	hace 3 horas	986 B



Memory: 114 / 2048 MB

Appendix 1. Introduction to Jupyter and Python

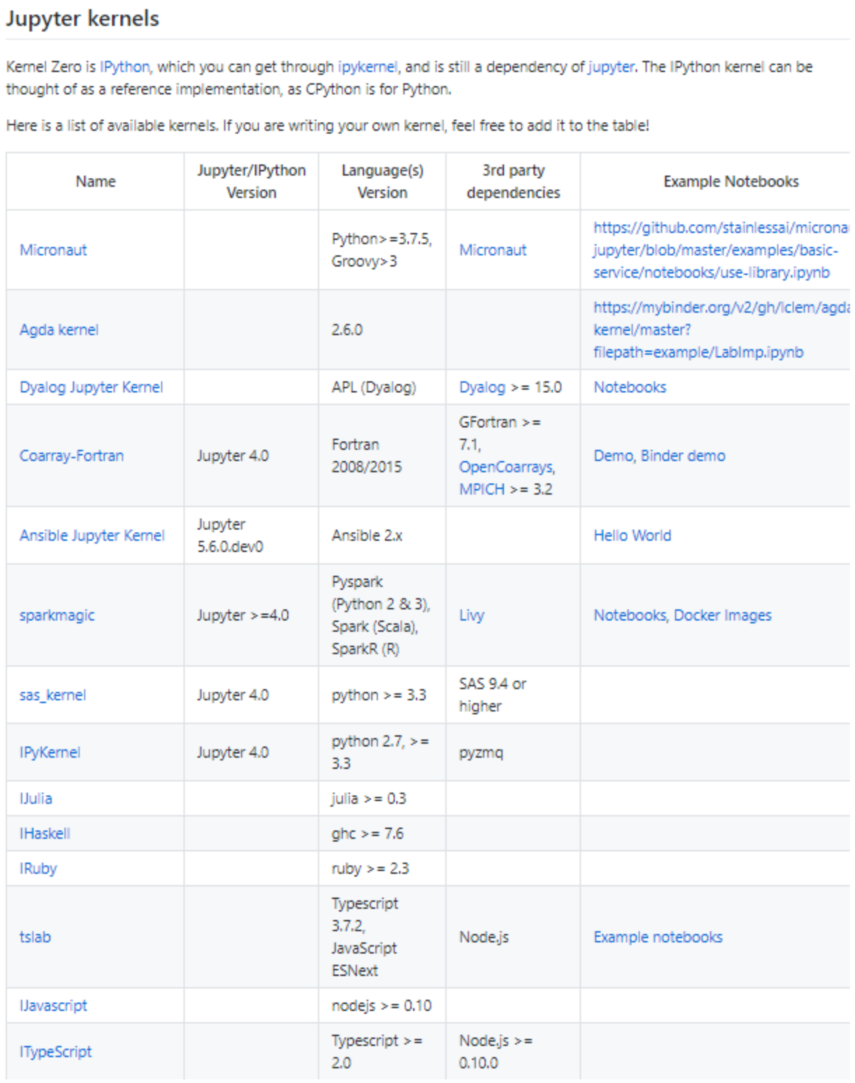
During this course, we are going to use this software called *Jupyter notebook*, which allows us to write and execute Python code at the moment.

Jupyter notebooks are divided in **cells** that can be executed by pressing `ctrl` + `Enter`. There are two main types of cells, **Markdown cells** and **Python cells**. Markdown cells are used for writting text and Python cells for writting code that can be executed.

```
In [1]: # This is a Python cell!!

        # This cell can be executed pressing ctrl + enter on it

        print(10*10)

        100
```

2. Jupyter Notebook: components

- **Kernel**
  - Backed application in charge of running the code contained in each cell.
  - Initially only the Python kernel was developed.
  - Now there is kernel support for more than 50 programming languages.



Jupyter kernels

Kernel Zero is IPython, which you can get through ipykernel, and is still a dependency of jupyter. The IPython kernel can be thought of as a reference implementation, as CPython is for Python.

Here is a list of available kernels. If you are writing your own kernel, feel free to add it to the table!

Name	Jupyter/IPython Version	Language(s) Version	3rd party dependencies	Example Notebooks
Micronaut		Python>=3.7.5, Groovy>3	Micronaut	https://github.com/stainlessai/microna jupyter/blob/master/examples/basic-service/notebooks/use-library.ipynb
Agda kernel		2.6.0		https://mybinder.org/v2/gh/lclem/agda kernel/master?filepath=example/LabImp.ipynb
Dyalog Jupyter Kernel		APL (Dyalog)	Dyalog >= 15.0	Notebooks
Coarray-Fortran	Jupyter 4.0	Fortran 2008/2015	GFortran >= 7.1, OpenCoarrays, MPICH >= 3.2	Demo, Binder demo
Ansible Jupyter Kernel	Jupyter 5.6.0.dev0	Ansible 2.x		Hello World
sparkmagic	Jupyter >=4.0	Pyspark (Python 2 & 3), Spark (Scala), SparkR (R)	Livy	Notebooks, Docker Images
sas_kernel	Jupyter 4.0	python >= 3.3	SAS 9.4 or higher	
IPyKernel	Jupyter 4.0	python 2.7, >= 3.3	pyzmq	
IJulia		julia >= 0.3		
IHaskell		ghc >= 7.6		
IRuby		ruby >= 2.3		
tslab		Typescript 3.7.2, JavaScript ESNext	Node.js	Example notebooks
IJavascript		nodejs >= 0.10		
ITypeScript		Typescript >= 2.0	Node.js >= 0.10.0	

2. Jupyter Notebook: how to use

- Alternatives to work with Jupyter notebooks:
  - Online services:
    - Mybinder
    - Google Colab
  - Local installations:
    - Traditional: install Python and all the needed packages.
    - Install a distribution like Anaconda.
    - Utilize a package for managing environments.

- More information at *Campus Virtual*.
  - Tools for working with Jupyter notebooks

2. Jupyter Notebook: live demo

Let’s see Jupyter notebooks in action!