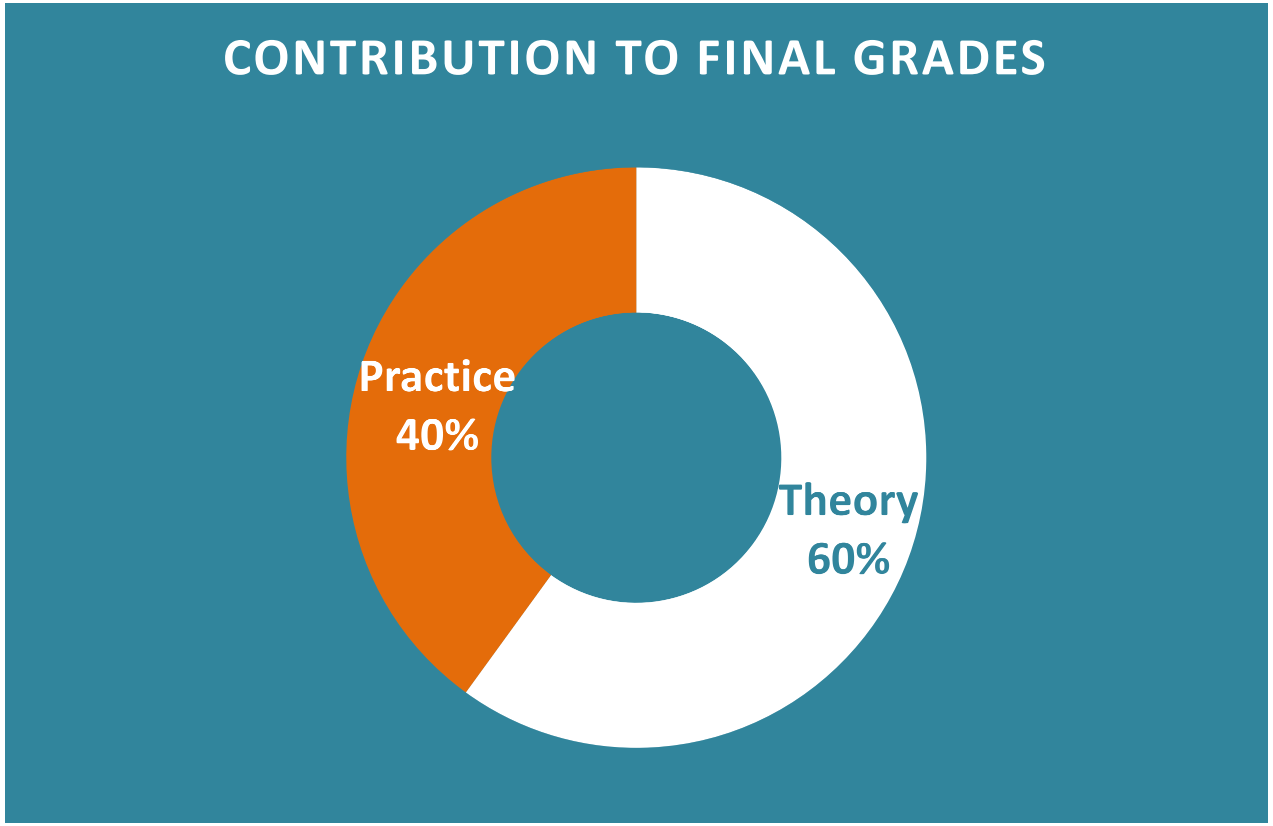
Introduction to Practical exercises workflow

José **Raúl** Ruiz Sarmiento

Content

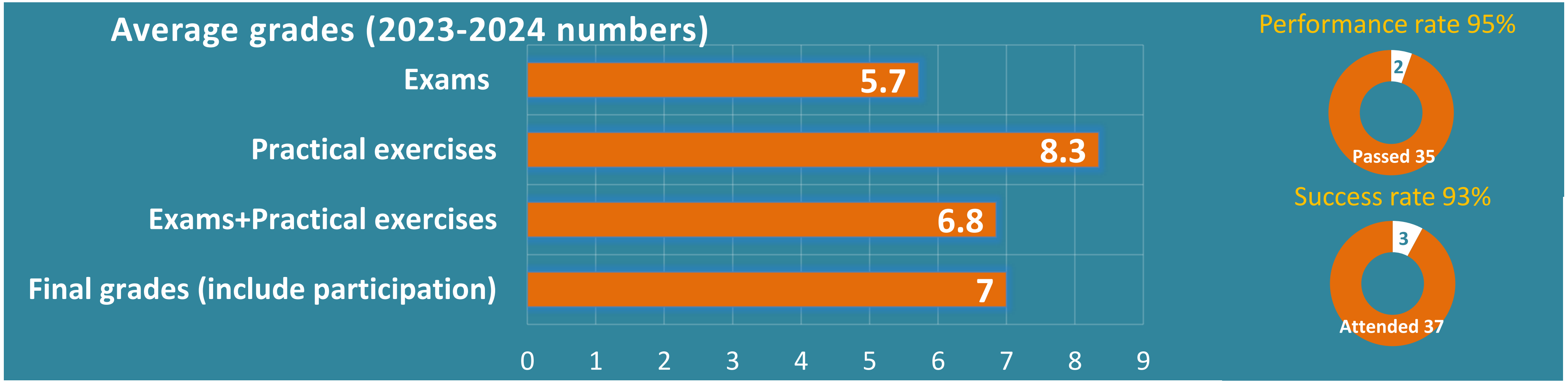
1. Practical exercises
    - Weight
    - Relevance
    - Workflow
2. Optional projects
3. The tool: JupyterLab
    - Components.
    - How to use.
    - Live demo.

1. Practical exercises: weight



- Practice:
  - 25%: Attendance **(70%)** and participation in lessons/forums.
  - 75% Weighted average of practical exercises.
- Once the subject is passed, other factors can **increase** the grades:
  - Extra/optional exercises.
  - Optional projects (up to 2 points).
  - Etc.

1. Practical exercises: relevance



	2019-2020	2020-2021	2021-2022	2022-2023
Exams	6,7	6,3	6,8	5,5
Practical exercises	7,7	7,7	8,2	7,3
Exams+Exercises	7,1	7,3	7,2	6,5
Final Grades	7,5	7,4	7,5	6,7
Performace rate	-	-	-	21/23 (91%)
Success rate	-	-	-	23/27 (85%)

1. Practical exercises: workflow

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

- Chapter 01. Welcome
- Chapter 02. Probability and statistic b...
- Chapter 03. Robot motion
- Chapter 04. Robot sensing
- Chapter 05. Localization
- Chapter 06. Mapping
- Chapter 07. SLAM
- Chapter 08. Motion planning
- Chapter 09. Robot control architecture

LECTURE MATERIAL

1. Introduction to Autonomous Robotics [pdf]
2. Probability and Statistics Bases for Robotics [pdf]
3. Robot Motion [pdf]
4. Robot Sensing [pdf]
5. Robot Localization [pdf]
6. Mapping [pdf]
7. SLAM [pdf]
8. Motion planning [pdf]
9. Robot Control Architecture + ROS [pdf]

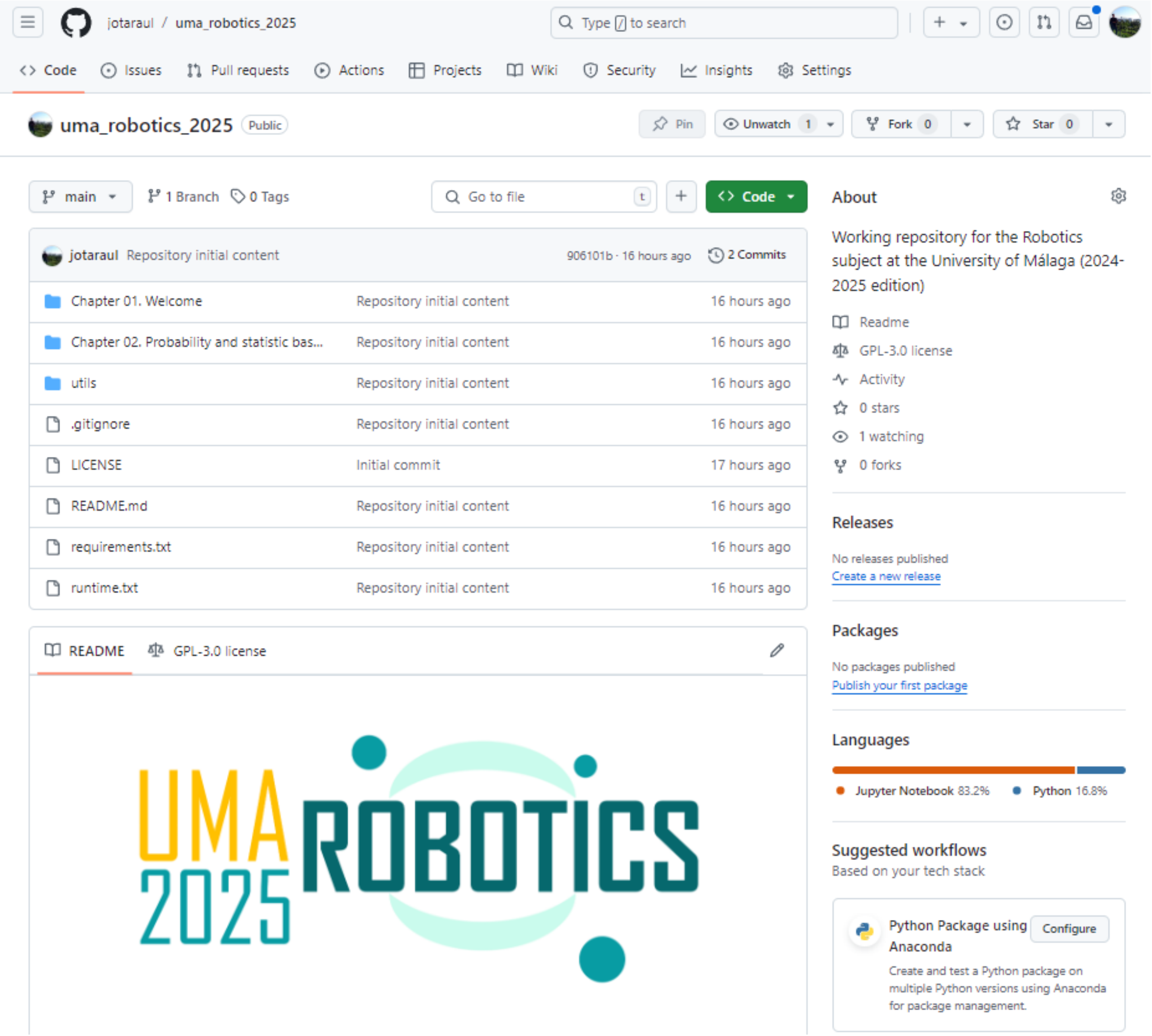
1. Practical exercises: workflow

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

- Chapter 02. Probability and statistic b...
  - 2-Fundamentals-1-GaussianDistributi...
  - 2-Fundamentals-2-PropertiesOfGauss...
  - 2-Fundamentals-3-BidimensionalDistr...
- Chapter 03. Robot motion
  - 3-Robot motion.ipynb
- Chapter 04. Robot sensing
  - 4-Robot sensing.ipynb
- Chapter 05. Localization
  - 5-Localization-0.ipynb
  - 5-Localization-1-LeastSquares.ipynb
  - 5-Localization-2-EKF.ipynb
- Chapter 06. Mapping
  - 6-Mapping-0.ipynb
  - 6-Mapping-1-EKF.ipynb

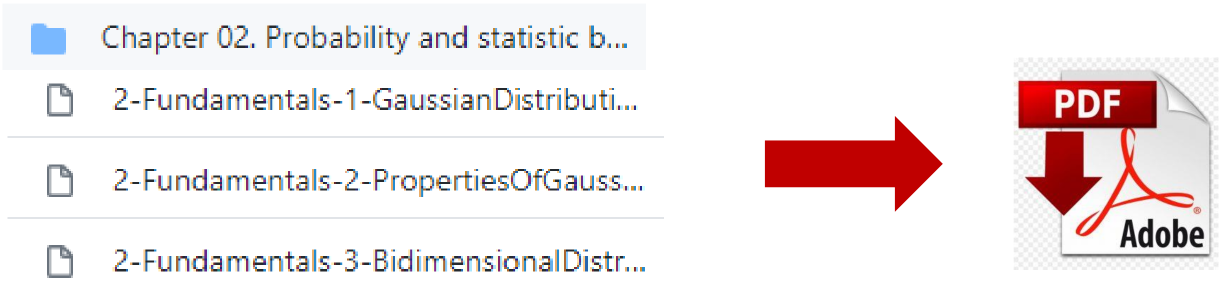
1. Practical exercises: workflow

- Notebooks are provided to students through a GitHub repository: **uma_robotics_2025**
- URL: https://github.com/jotaraul/uma_robotics_2025



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**.



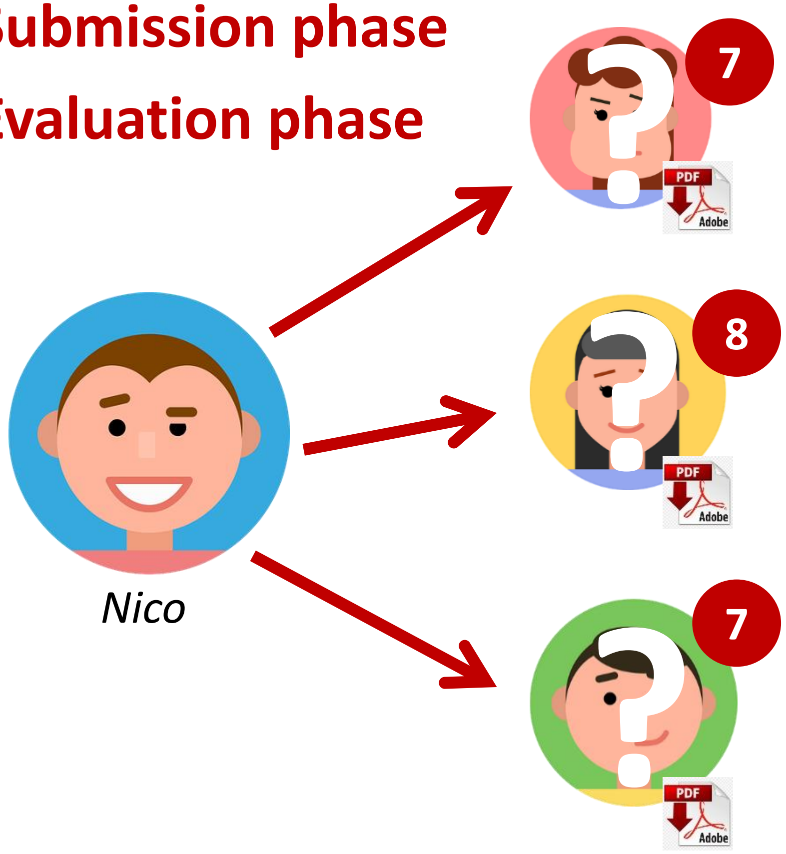
- A **workshop** will be enabled for that in the *Campus Virtual*. Workshop Chapter 6
- Such submissions will have a **hard deadline**.
  - Submissions beyond this deadline will not be accepted.

1. Practical exercises: workflow

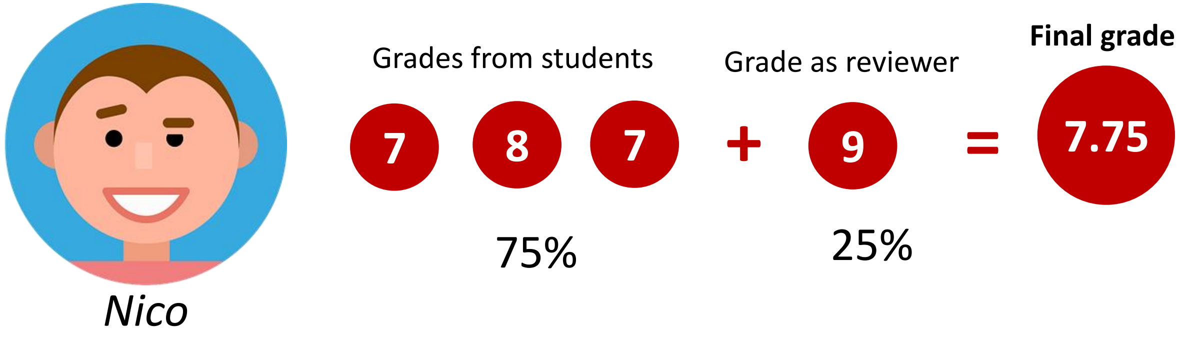
- **Evaluation** will be done through the **Workshop (taller)** activity of the *Campus Virtual*.

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. Optional projects

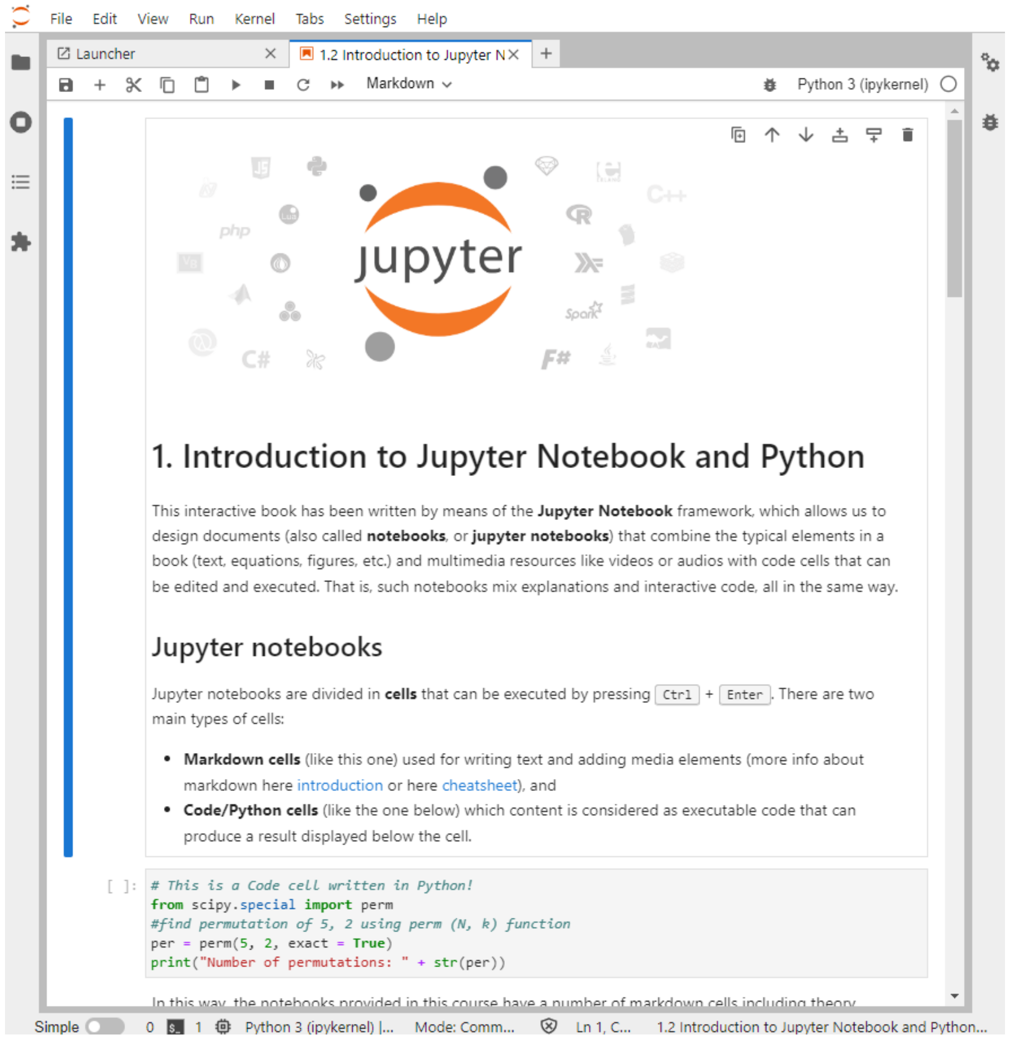
- The student is challenged to develop a robotics based optional project.
- How it works:
  1. A list of project proposals is provided, but the student may carry out one of his/her choice.
  2. The student send a project proposal.
  3. The student develop the project. He/she can integrate third party code.
  4. A workshop/personal interview will take place where each student has the opportunity to present the project.
  5. It can be done in pairs (twice work to get the maximum mark).

Final grade of optional projects:

A combination of the work done and the project presentation.

3. The tool: JupyterLab

- 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.



3. The tool: JupyterLab

- GitHub public Jupyter notebooks:
  - 2015: 200K
  - 2018: 2,5M
  - 2020: 9,7M
  - 2024: 10M
- 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:

3. JupyterLab: 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

3.1 Pose composition

Given an initial pose p_1 and a pose differential Δp, *i.e.* how much the robot has moved during an interval of time, we compute the final pose p using the **composition of poses** function:

Equations

$$p_1 = \begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \end{bmatrix}, \Delta p = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix}$$

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = p_1 \oplus \Delta p = \begin{bmatrix} x_1 + \Delta x \cos\theta_1 - \Delta y \sin\theta_1 \\ y_1 + \Delta x \sin\theta_1 + \Delta y \cos\theta_1 \\ \theta_1 + \Delta\theta \end{bmatrix}$$

Text

The differential Δp, although we are using it as control in this exercise, normally is calculated given the robot's locomotion or sensed by the wheel encoders.

Assignment

Take a look at the `Robot()` class provided and its methods. Then, modify the main function in the next cell for the robot to describe a $8m \times 8m$ square path as seen in the figure below.

The robot starts in the bottom-left corner $(0, 0)$ heading north and moves at increments of $2m$ each step. Each 4 steps it will turn right.

Example

Figures

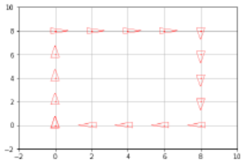


Fig. 1: Route of our robot.

INTERACTIVE CODE

In [2]:

```
class Robot():
    '''Mobile robot implementation

        Attr:
            pose: Expected position of the robot
    '''
    def __init__(self, mean):
        self.pose = mean

    def step(self, u):
        self.pose = tcomp(self.pose, u)

    def draw(self, fig, ax):
        DrawRobot(fig, ax, self.pose)
```

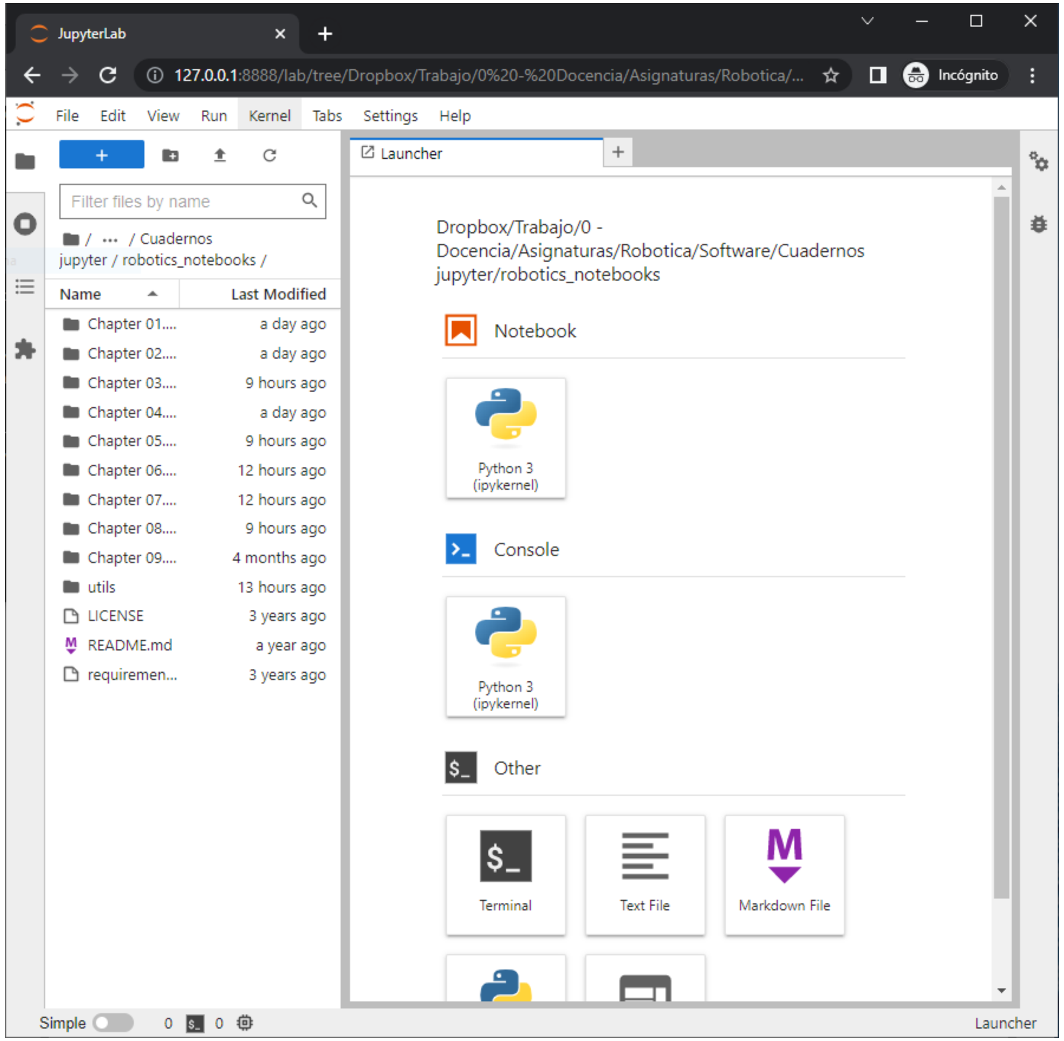
In [3]:

```
def main(robot):
    # TO DO
    None
```

Incomplete code

3. JupyterLab: components

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



3. JupyterLab: 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].

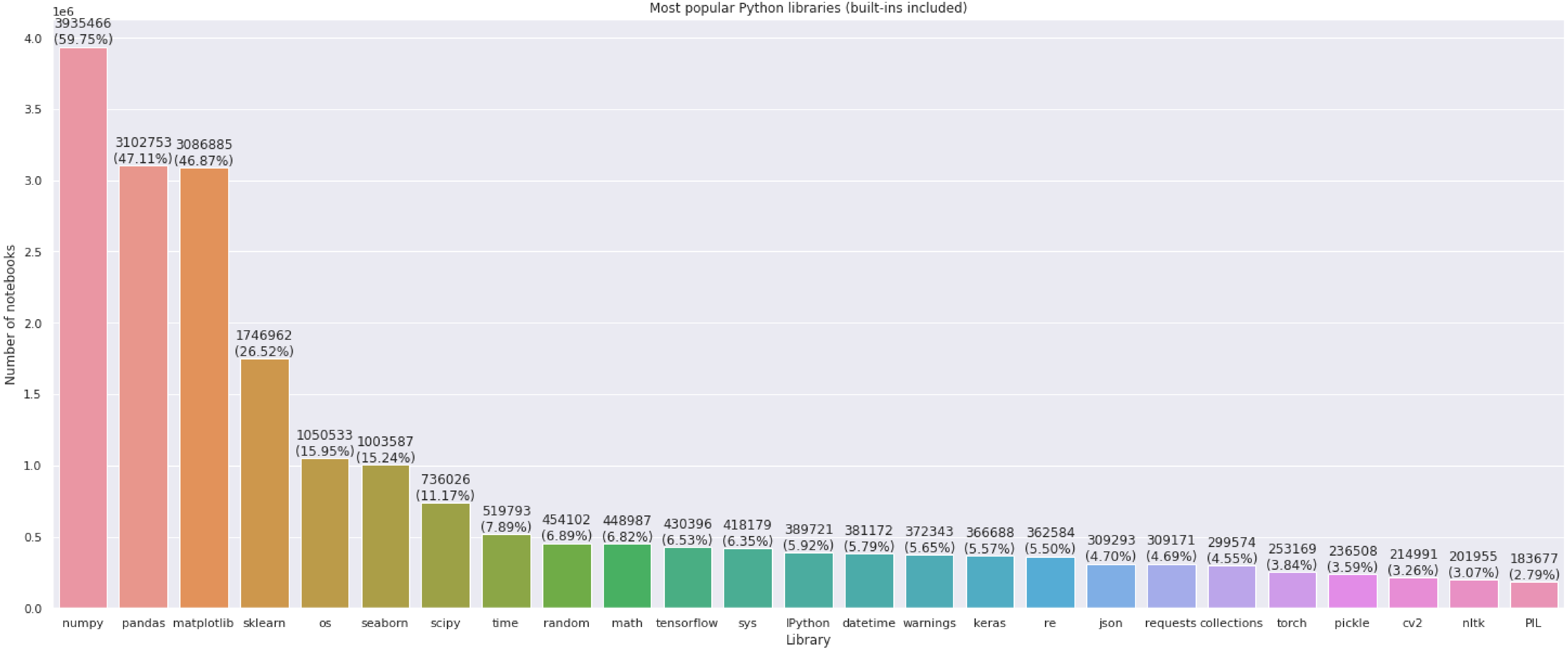
Jupyter kernels

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	

3. JupyterLab: python modules



3. JupyterLab: how to use

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

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

3. JupyterLab: live demo

Let’s see Jupyter notebooks in action!