4.1 Jupyter Lab

Which IDE/environment/tool should be used? This is one of the most frequently asked questions when working on data science projects. As you can imagine, we have no shortage of options available-from language-specific IDEs such as R Studio or PyCharm to editors such as Sublime Text or Atom-too many choices may make it difficult for beginners to start.

If there are any tools that every data scientist should use or must understand, it is Jupyter Lab (previously also known as iPython notebook). Jupyter Lab is powerful, multi-functional, sharable, and provides the ability to perform data visualization in the same environment.

Jupyter Lab allows data scientists to create and share their documents, from code to comprehensive reports. They can help data scientists simplify work processes, achieve higher productivity and more convenient collaboration. For these and the reasons you will see below, Jupyter Lab has become one of the most commonly used tools for data scientists. In future courses, we will use Jupyter Lab to demonstrate.

4.1.1 What is Jupyter Lab?

Jupyter Lab (previously known as IPython notebook, Jupyter notebook) is an interactive notebook that supports running more than 40 programming languages. The essence of Jupyter Lab is an open source web application that facilitates the creation and sharing of literary program documents, and supports real-time code, mathematical equations, visualization and markdown. Uses include: data cleaning and conversion, numerical simulation, statistical modeling, machine learning, etc. We can use it to create and share code and documents. It uses Python (there are also cores of other languages such as R, Julia, Node) for code demonstration, data analysis, visualization, and teaching. Python is becoming more and more popular. And the leading position in the AI field has a great role in promoting.

Jupyter Lab is an extension of Jupyter that provides a better user experience. For example, you can open and edit multiple Notebooks, Ipython consoles and terminal terminals in a browser page at the same time, and it supports previewing and editing more types of files, such as code files. , Markdown documents, json, yml, csv, pictures in various formats, vega files (a language that uses json to define charts) and geojson (using json to represent geographic objects), you can also use Jupyter Lab to connect to cloud storage services such as Google Drive , Which greatly improves productivity.

It provides an environment in which you can write your code, run the code, view the output, visualize the data, and view the results without leaving this environment. Therefore, this is a convenient tool that can perform end-to-end data science workflows, including data cleaning, statistical modeling, building and training machine learning models, visualizing data, and more.

When you are still in the prototyping stage, the role of Jupyter Lab is even more compelling. This is because your code is written in the form of independent units, and these units are executed independently. This allows users to test specific code blocks in a project without having to execute code from the beginning of the project. Many other IDE environments (such as RStudio) have several other ways to do this, but I personally think that Jupyter's single unit structure is the best.

As described above, Jupyter Lab is very flexible and can provide data scientists with powerful interactive capabilities and tools. They even allow you to run languages other than Python, such as R, SQL, etc. Because they are more interactive than pure IDE platforms, they are widely used to present code in a more pedagogical way.

4.1.2 How to install Jupyter Lab?

The installation method on Jetbotmini has been explained in chapter 3.4.4

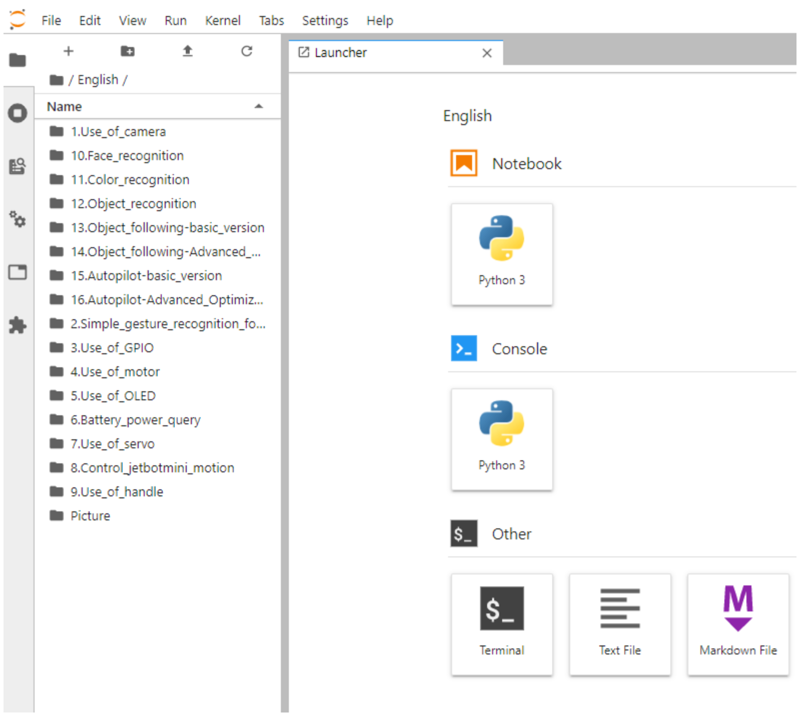
4.1.3 Get started

Now that you know what these notebooks are, start using them now! To run your Jupyter Lab, you first need a PC connected to Jetbotmini.

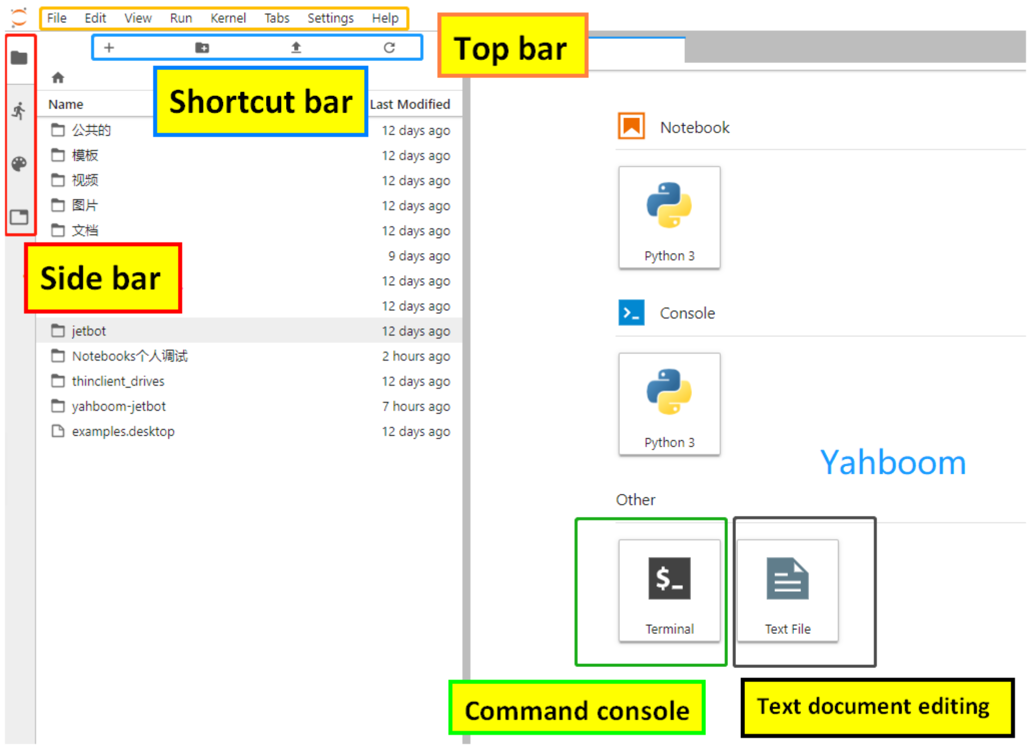
When studying the courses of Chapter 5 and Chapter 6, the most convenient way is to run and observe the code through jupyter lab. First of all, our Jetbotmini is connected to the same local area network as the PC network through WIFI. When logging in, just enter the IP address displayed on the OLED screen of the car on the Google browser, and add: 8888. For example: http://192.168.2.63:8888. Please note that it can only be Google Chrome, other browsers are not easy to use, and even cannot be logged in. The login password is "yahboom".

If there is no network environment support, we can use Jetbotmini's "headless mode" to connect to the network channel established by the PC via USB. The headless mode is the default IP of 192.168.55.1. Log in to http://192.168.55.1 on the browser: 8888, to use JupyterLab of Jetbotmini. If we log in to Jetbotmini remotely using the current PC environment for the first time, we need to enter a password for verification. The password is the password we set when installing JupyterLab in section 3.4.4. If you If you are using Yahboom-Jetbotmini our official Jetbotmini factory image, the default password is **yahboom**. After entering the password once in this environment, JupyterLab will record the current environment. You don’t need to enter the password here next time. Log in to the URL to enter the Jupyter Lab interface and use it immediately.

Select "English" after login, the interface is as shown in the figure below:



The JupyterLab interface is a dashboard that can access interactive iPython notebooks, as well as the folder structure of Jetbotmini and a terminal window into the Ubuntu operating system. The first view you will see includes the **Menu Bar** at the top, the directory tree in the **Left Sidebar**, and the **Main Workspace** initially opened to the Launcher page.



For complete details on all features and menu operations, see the JupyterLab interface:

https://jupyterlab.readthedocs.io/en/stable/user/interface.html

Here are some key features that are especially useful in this course:

- **File Browser:**

The file browser in the left column allows you to navigate the Jetson Nano file structure. Double-clicking on a notebook or file will open it in the main workspace.

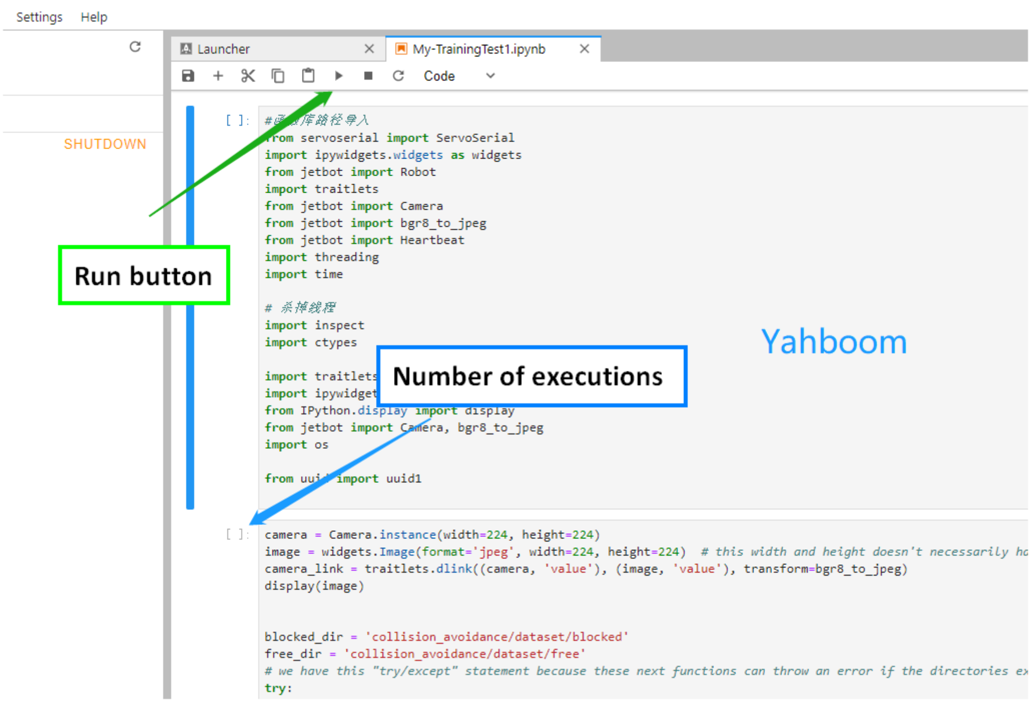
- **iPython notebook:**

The interactive notebook used in this course has an ".ipynb" file extension.

When you double-click a notebook from the file browser, it will open in the main workspace and the process will begin. The notebook includes text and code "cells". When the code unit is "running", by clicking the Run button at the top of the notebook or the keyboard shortcut 【CTRL】 + 【ENTER】, the code block in the cell is executed and the output is displayed below the laptop.

On the left side of each executable cell, there is an "execution count" or "prompt number" in parentheses. If the cell is running for more than a few seconds, you will see an asterisk mark there, indicating that the cell has not completed execution. When the cell is processed, a number will appear in parentheses.

As shown below:



```
[ ]: #函数库路径导入
     from servoserial import ServoSerial
     import ipywidgets.widgets as widgets
     from jetbot import Robot
     import traitlets
     from jetbot import Camera
     from jetbot import bgr8_to_jpeg
     from jetbot import Heartbeat
     import threading
     import time

     # 杀掉线程
     import inspect
     import ctypes

     import traitlets
     import ipywidget
     from IPython.display import display
     from jetbot import Camera, bgr8_to_jpeg
     import os

     from uuid import uuid1

[ ]: camera = Camera.instance(width=224, height=224)
     image = widgets.Image(format='jpeg', width=224, height=224)  # this width and height doesn't necessarily ha
     camera_link = traitlets.dlink((camera, 'value'), (image, 'value'), transform=bgr8_to_jpeg)
     display(image)


     blocked_dir = 'collision_avoidance/dataset/blocked'
     free_dir = 'collision_avoidance/dataset/free'
     # we have this "try/except" statement because these next functions can throw an error if the directories ex
     try:
```

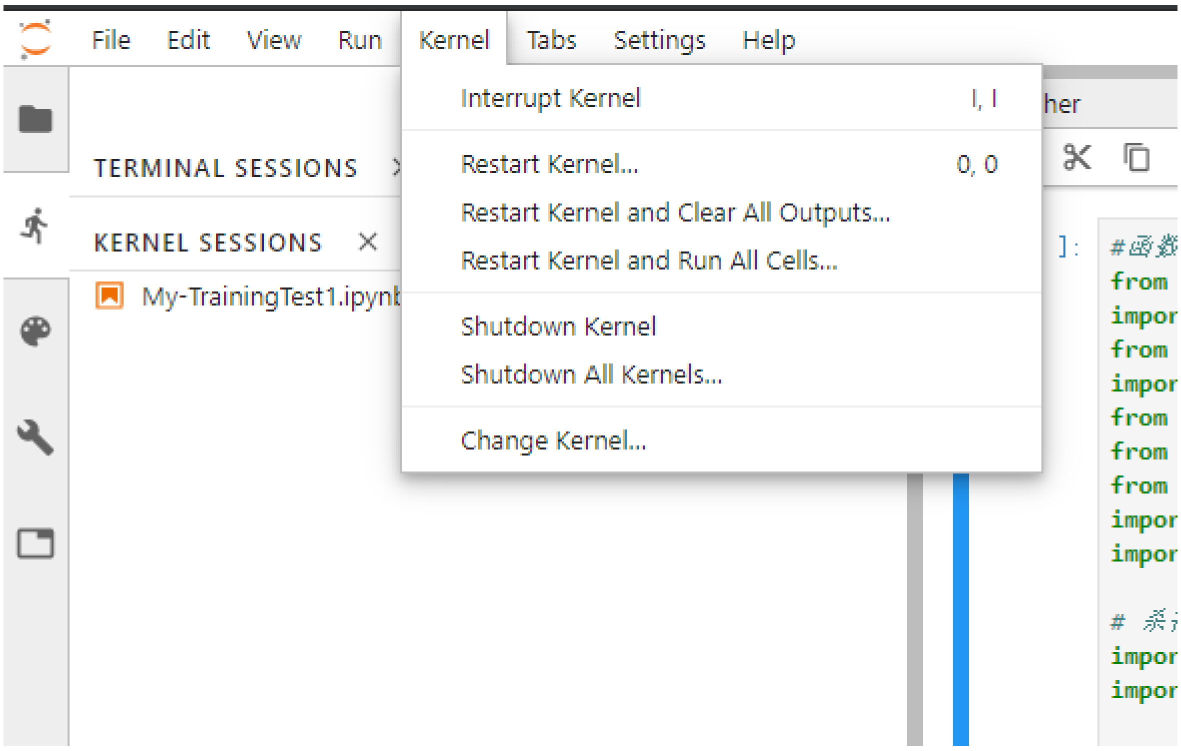
- **Kernel operation:**

The kernel of each running notebook is a separate process running user code. When you open your notebook from the file browser, the kernel starts automatically.

The kernel menu on the main menu bar contains commands to close or restart the kernel, and you need to use them regularly. No code units can be executed after the kernel is shut down.

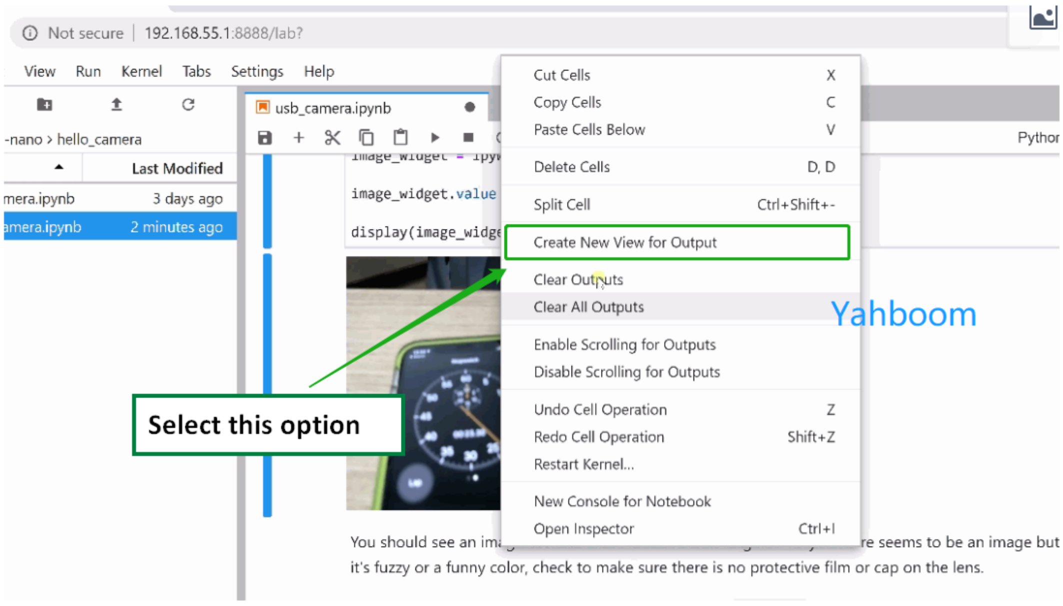
When the kernel is restarted, all memory is lost due to imported packages, variable assignments, and so on.

As shown below:



- **Cell label:**

You can move any cell to the new window tab in the main workspace by right-clicking on the cell and selecting “Create New View for Output”.



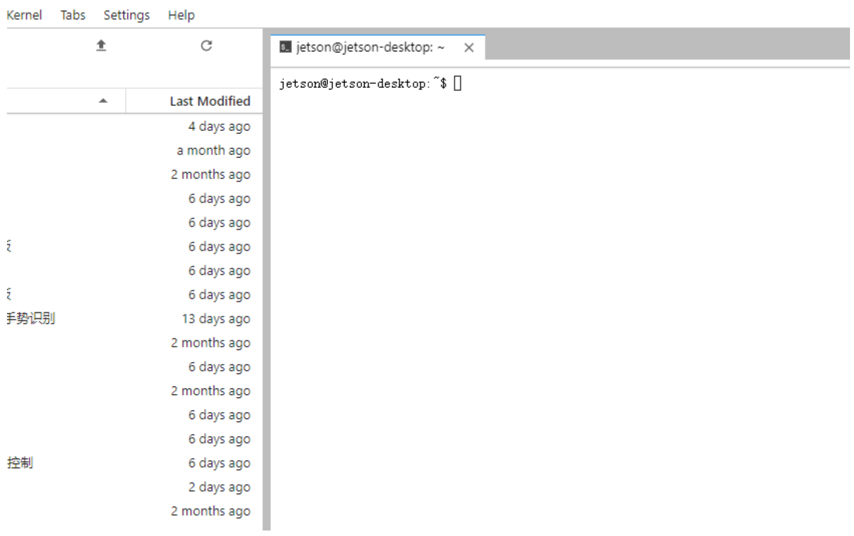
```
image_widget = ipyw
image_widget.value
display(image_widge
```

You should see an ima re seems to be an image but it's fuzzy or a funny color, check to make sure there is no protective film or cap on the lens.

- **Terminal window:**

You can work directly in the terminal window of the Jetbot Ubuntu OS via Jupyter remote login, As shown below.

In the Launcher page, click the terminal icon under "Other" to bring up the Launcher page. If it is no longer visible, click the "+" icon at the top of the left column.



```
jetson@jetson-desktop:~$ 
```

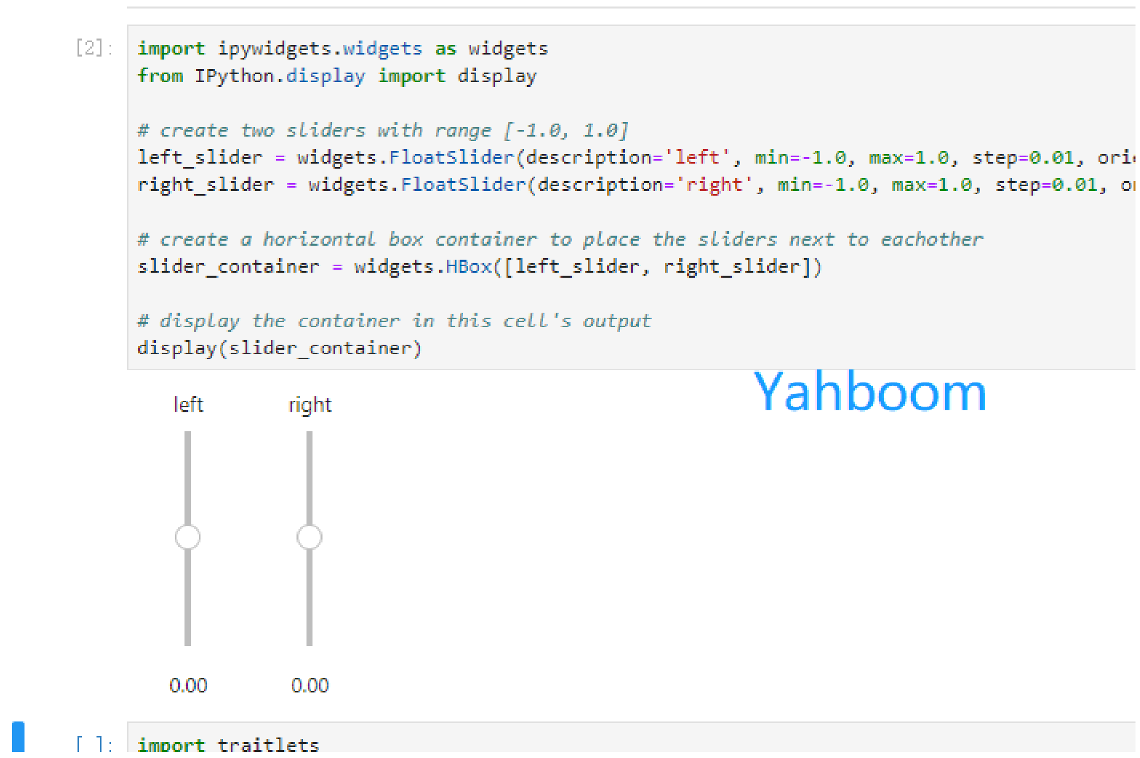
4.1.4 Interactive Dashboard in Notebooks

Before you consider adding widgets, you need to import the widgets package:

from ipywidgets import widgets

The basic types of widgets are typical text input widgets, input-based widgets, and button widgets.

The following example from Dominodatalab gives some appearance of interactive widgets:



```
[2]: import ipywidgets.widgets as widgets
     from IPython.display import display

     # create two sliders with range [-1.0, 1.0]
     left_slider = widgets.FloatSlider(description='left', min=-1.0, max=1.0, step=0.01, ori
     right_slider = widgets.FloatSlider(description='right', min=-1.0, max=1.0, step=0.01, or

     # create a horizontal box container to place the sliders next to eachother
     slider_container = widgets.HBox([left_slider, right_slider])

     # display the container in this cell's output
     display(slider_container)
```

```
[ ]: import traitlets
```

4.1.5 Keyboard shortcuts

Shortcuts are one of the biggest advantages of Jupyter Notebooks. When you want to run arbitrary blocks of code, just press **Ctrl+Enter**. Jupyter Notebooks provides a lot of keyboard shortcuts that can help us save a lot of time.

Jupyter Notebooks offers two different keyboard input modes - command and editing.

The command mode is to bind keyboard and notebook level commands and is represented by a gray cell border with a blue left margin.

Edit mode allows you to enter text (or code) in the active cell, represented by a green cell border.

You can switch between command mode and edit mode using **Esc** and **Enter** respectively.

Press Esc button on your keyboard to enter the command mode, you can try the following shortcuts:

Shift+Enter : Run this unit and select the next unit

Ctrl+Enter : Run this unit

Alt+Enter : Run this unit and insert a new unit below it

Y: unit into code status

M: unit goes to markdown state

A : Insert a new unit above

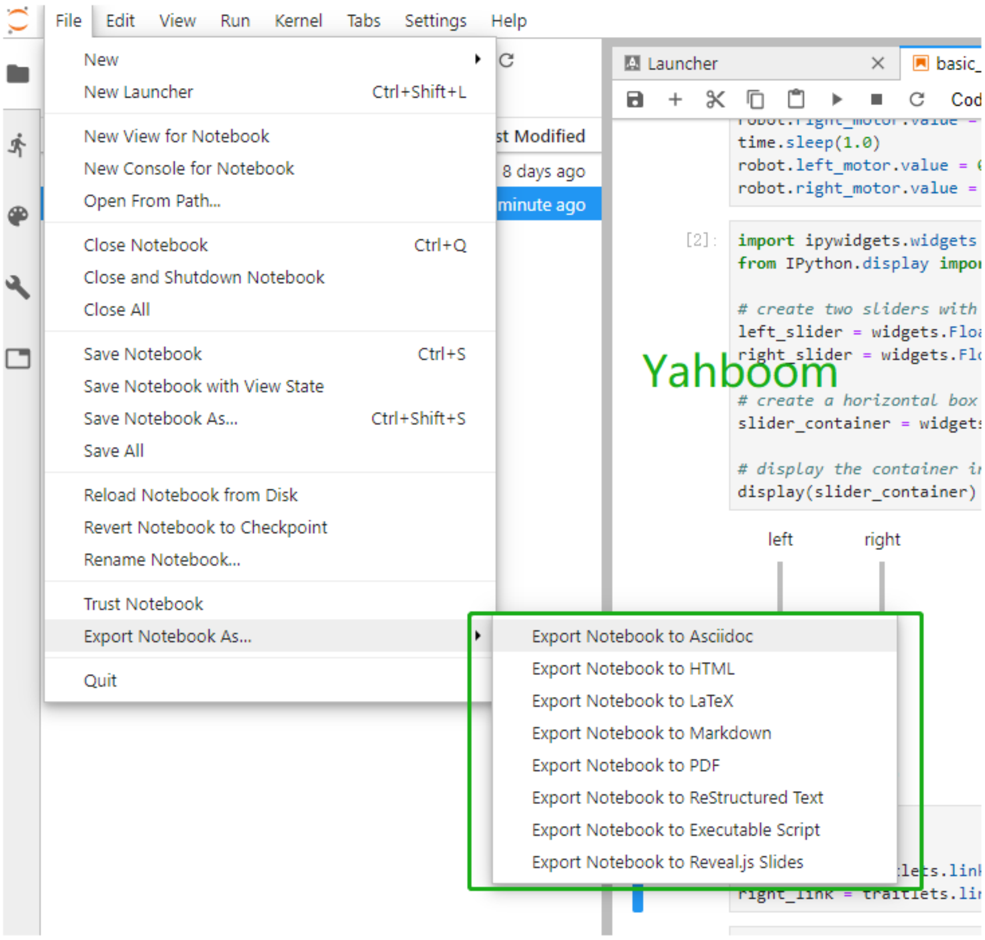
B: Insert a new unit below

X: Cut the selected unit

Shift +V: Paste the unit above

4.1.6 Save and share your notebook

Go to the 【Files】 menu and you will see the 【Export Notebook As...】 option:



```
time.sleep(1.0)
robot.left_motor.value = (
robot.right_motor.value =
```

```
[2]: import ipywidgets.widgets
     from IPython.display impor

     # create two sliders with
     left_slider = widgets.Floa
     right_slider = widgets.Flo

     # create a horizontal box
     slider_container = widget

     # display the container i
     display(slider_container)
```

You can export and save your notebook in 8 optional formats.

The most common ones are .ipynb files and .html files. Using the .ipynb file allows others to copy your code to their machine, and .html files can be opened in web format (it's handy when you need to save images embedded in your notebook).

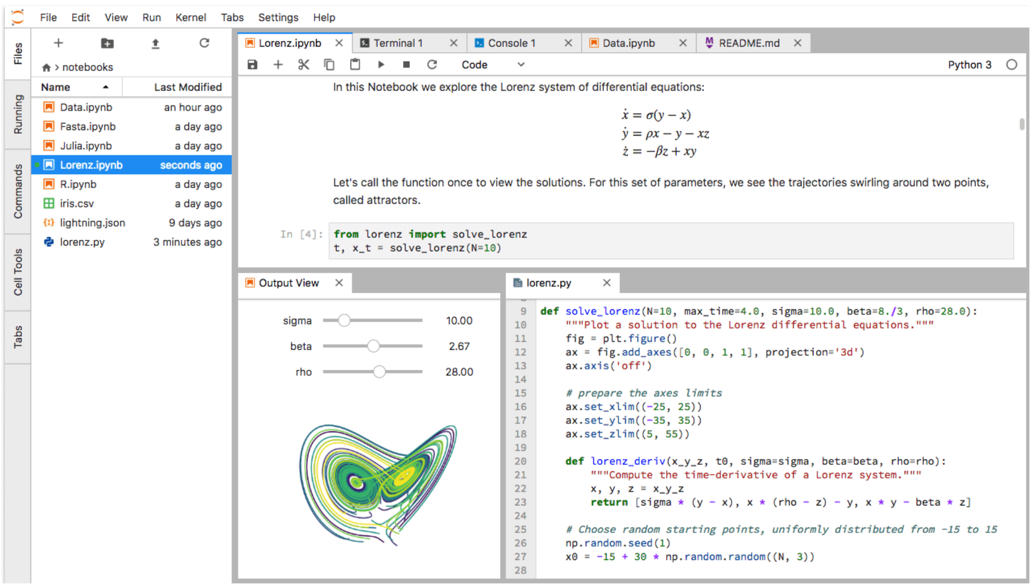
You can also manually convert your notebook to HTML or PDF using the nbconvert option.

You can also use jupyterhub at https://github.com/jupyterhub/jupyterhub. It allows you to host your laptop on its servers and share it with multiple users.

4.1.7 7. JupyterLab - Evolution of Jupyter Notebook

JupyterLab supports a more flexible and powerful way of working with projects, but with the same components as Jupyter Notebooks. The JupyterLab environment is exactly the same as the Jupyter Notebooks environment.

JupyterLab allows you to arrange your notebook, terminal, text file and output result workspaces in one window. You just drag and drop the units you need. You can also edit common file formats such as Markdown, CSV and JSON. The impact of real-time preview changes.



Name	Last Modified
Data.ipynb	an hour ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ipynb	seconds ago
R.ipynb	a day ago
iris.csv	a day ago
lightning.json	9 days ago
lorenz.py	3 minutes ago

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

```
In [4]: from lorenz import solve_lorenz
        t, x_t = solve_lorenz(N=10)
```

sigma	10.00
beta	2.67
rho	28.00

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))

```