

### 3.1 Jupyter Lab

#### 1. What is Jupyter Lab?

Jupyter Lab (IPython notebook, Jupyter notebook) is an interactive notebook that supports more than 40 programming languages. The essence of the Jupyter Lab is an open source web application that facilitates the creation and sharing of literary program documentation, supporting real-time code, mathematical equations, visualization and markdown.

Wide range of using:

Data cleansing and transformation, numerical simulation, statistical modeling, machine learning, and more. We can use it to create and share code and documentation. It uses Python (also has kernels in other languages, such as R, Julia, Node, etc.) for good documentation, data analysis, visualization, and teaching.

The Jupyter Lab is an extension of Jupyter that provides a better user experience, such as the ability to simultaneously open and edit multiple Notebook, Ipython console and terminal terminals in a single browser page, and to preview and edit more types of files, such as code files. , Markdown documents, json, yml, csv, images in various formats, vega files (a language that uses json to define charts) and geojson (geographic objects with json), and you can use Jupyter Lab to connect to cloud storage services such as Google Drive. .

It provides us with an environment where you can write your code, run code, view output, visualize data, and view results. Therefore, it is a convenient tool for performing end-to-end data science workflows, including data cleansing, statistical modeling, building and training machine learning models, visualizing data, and more.

As described above, the Jupyter Lab is very flexible and provides powerful interactive capabilities and tools for data scientists. They are more interactive than the pure IDE platform, they are widely used to present code in a more pedagogical way.

#### 2. How to install Jupyter Lab?

The installation method on Jetbot is already in **【2.4 Install Jetbot】---【5.Install Jupyter Lab】**

#### 3. Get started!

First, you need a PC to connect to Jetbot.

If our Jetbot is connected to the same LAN as the PC network via WIFI, we can log in directly on the PC-side Google Chrome or other browser (Jetbot IP



address): 8888 to Jetbot's Jupyter Lab, for example: <http://192.168.1.67:8888>

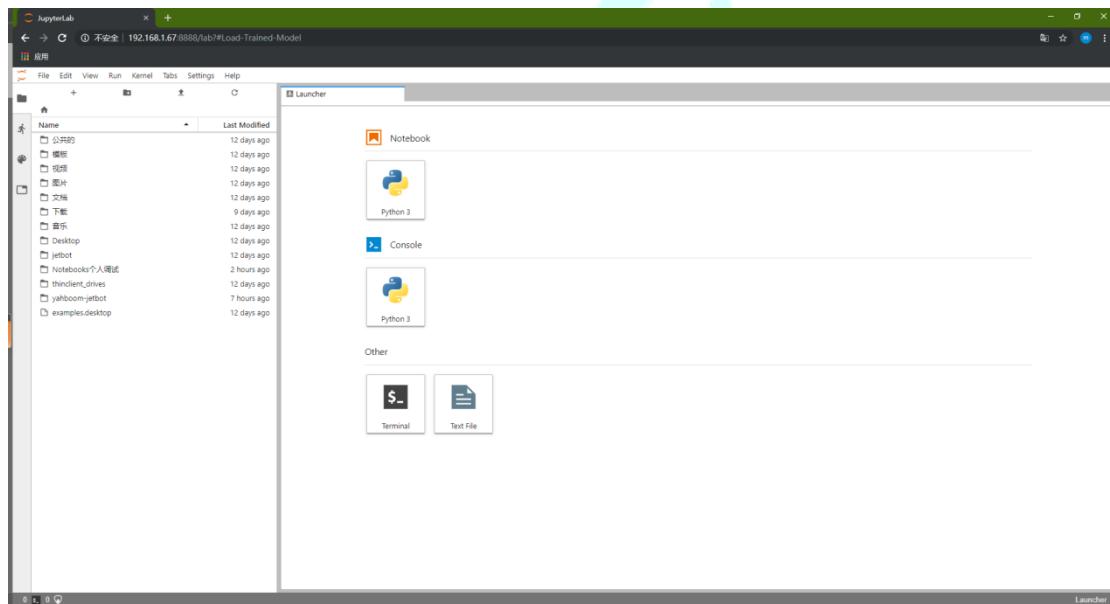
If there is no network environment support, we can use Jetbot's "headless mode" to connect to the network channel established by the PC via USB, and log in to the browser<http://192.168.55.1:8888>

To Jupyter Lab of Jetbot robot car , if we are using the current PC environment to log in to Jetbot for the first time, we need to enter the password for verification. The password is 【2.4 Install Jetbot】---【5.Install Jupyter Lab】the password set when Jupyter Lab installed.

If you are using **Yahboom\_jetbot\_car\_image** our official Jetbot image, the default password is **yahboom**.

When you enter the password once in this environment, Jupyter Lab will record the current environment. You will not need to enter the password here next time. The URL will be available to the Jupyter Lab interface for immediate use.

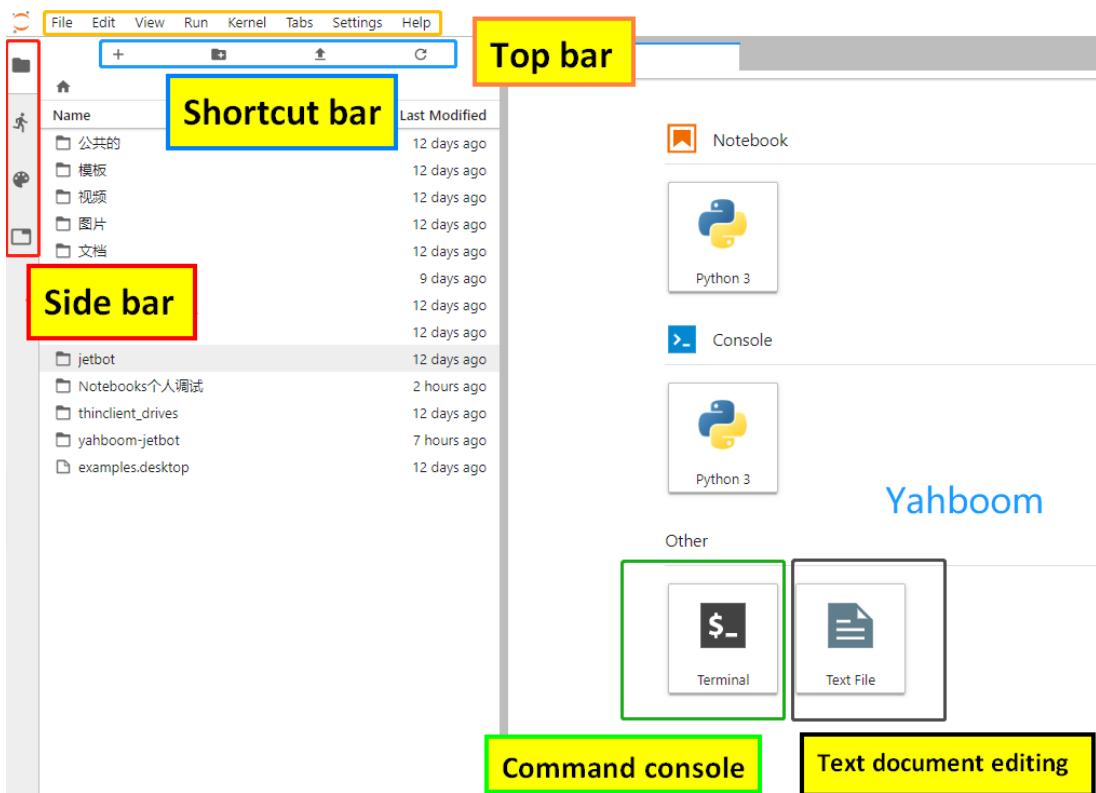
The interface after login is as shown below:



The JupyterLab interface is a dashboard that provides access to interactive iPython notebooks, as well as Jetbot's folder structure and terminal windows into the Ubuntu operating system.

The interface shown below includes the menu bar at the top, the directory tree in the left sidebar, and the main workspace that was originally opened to the Launcher page.

As shown below:



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:

### ① 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.

### ② 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:

The screenshot shows a Jupyter Notebook interface with the title bar "My-TrainingTest1.ipynb". On the left, there's a vertical toolbar with a "SHUTDOWN" button highlighted in orange. A green arrow points from the text "Run button" to the top-left corner of the code editor. Inside the code editor, a blue box highlights the text "Number of executions". A blue arrow points from the text "Number of executions" to the same highlighted area. The code itself is written in Python and includes imports for ServoSerial, ipywidgets, jetbot, and other modules. It also includes a section for camera configuration and a try/except block for directory handling.

```

# 从库端口输入
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

# 检查线程
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 have to be 224
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 exist
try:

```

Yahboom

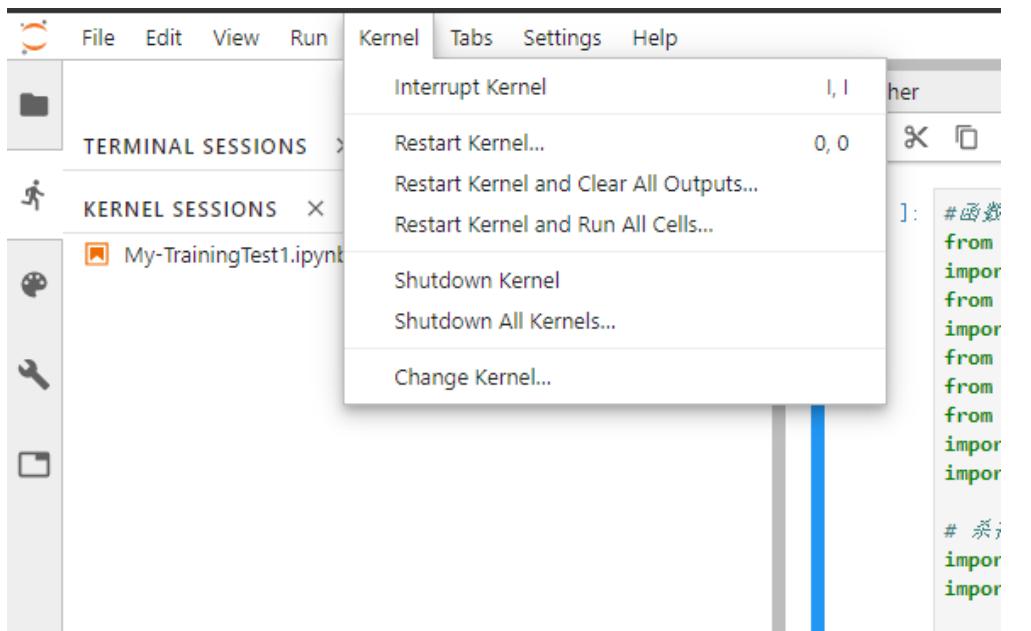
### ③Kernel operation:

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.

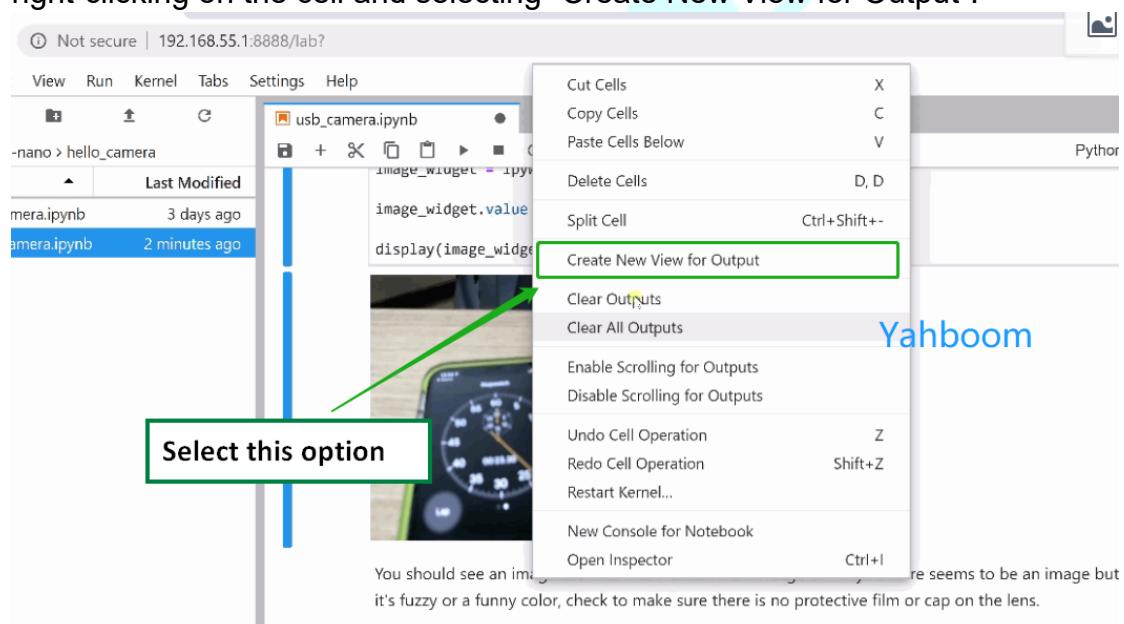
When the kernel is restarted, all memory is lost due to imported packages, variable assignments, and so on.

As shown below:



#### ④Cell label:

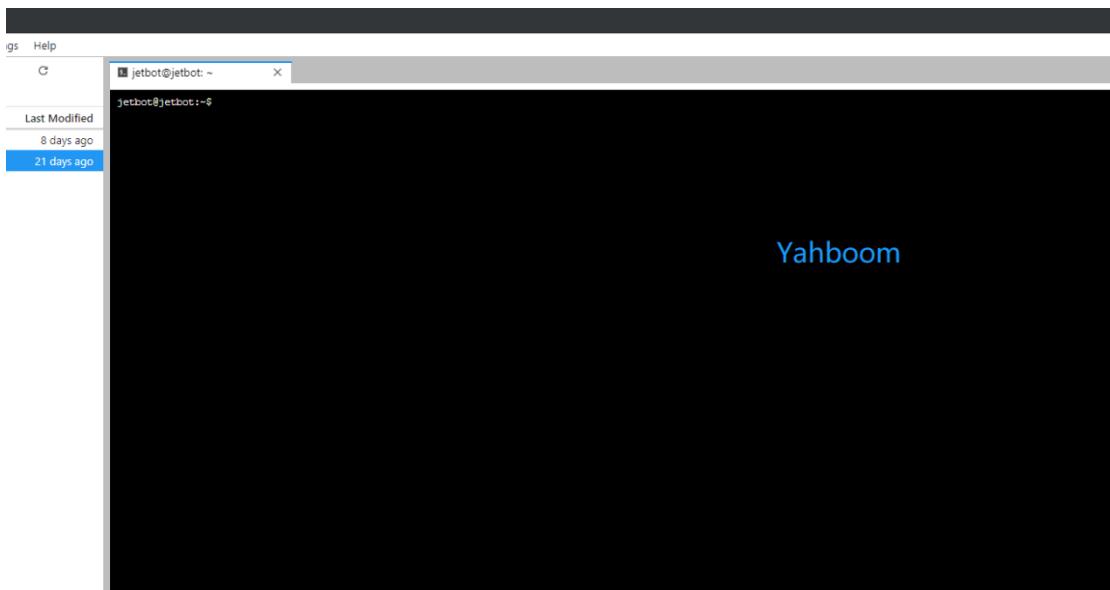
You can move any cell to the new window tab in the main workspace by right-clicking on the cell and selecting “Create New View for Output”.



#### ⑤Terminal 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.



Yahboom

#### 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:

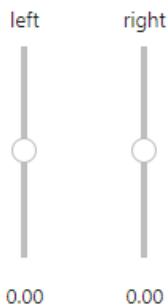
```
[2]: import ipywidgets.widgets as widgets
from IPython.display import display

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

# 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)
```

Yahboom



#### 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

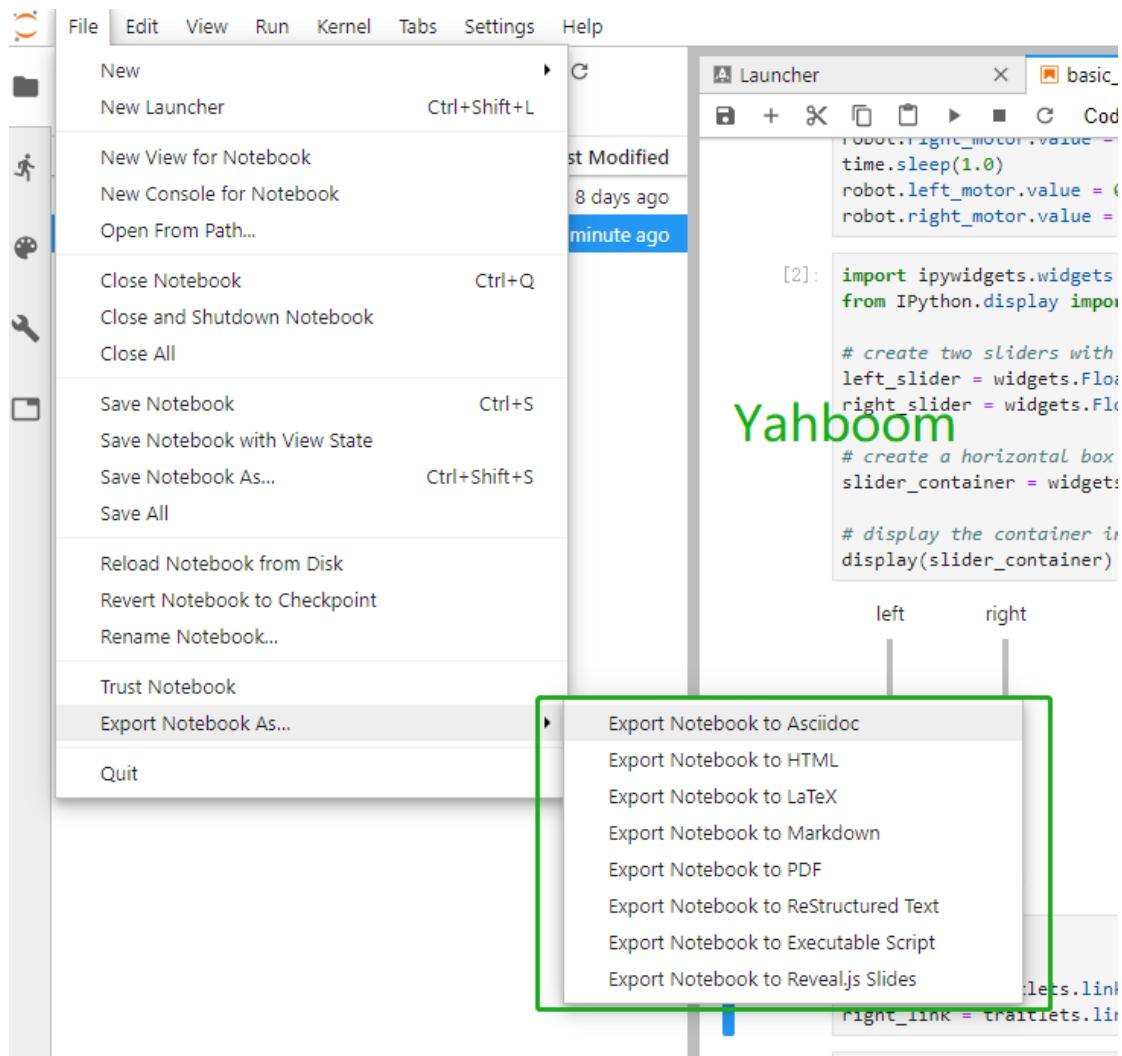
**B**: Insert a new unit below

**X**: Cut the selected unit

**Shift +V**: Paste the unit above

## 6. Save and share your notebook

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



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.

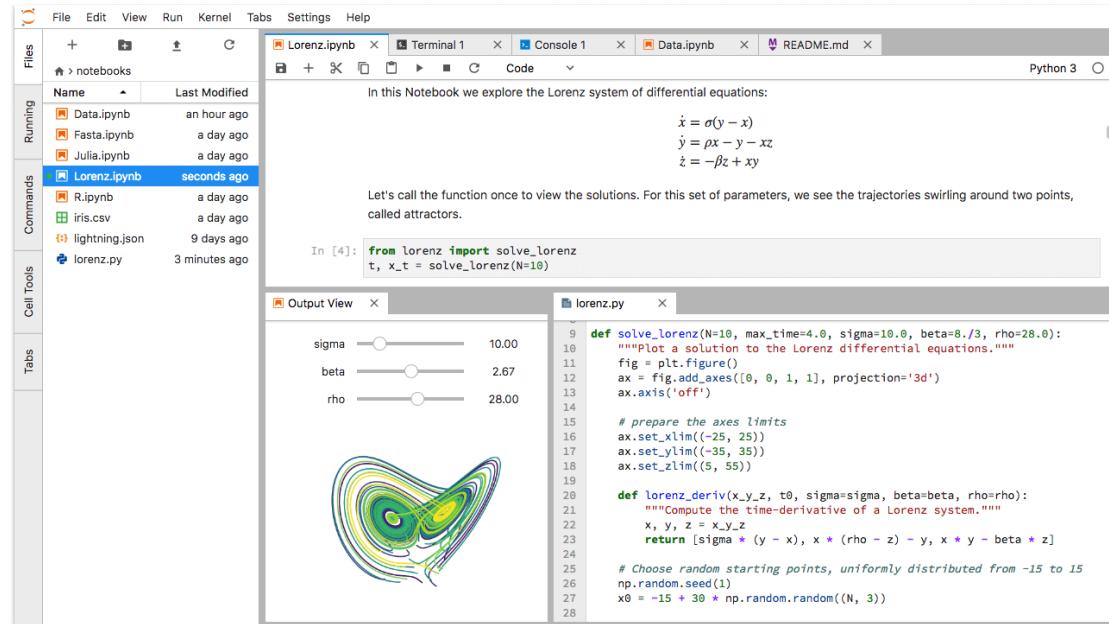
You can also use jupyterhub at <https://github.com/jupyterhub/jupyterhub>. It allows you to host your laptop on its servers and share it with multiple users.

## 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.



The screenshot shows a Jupyter Notebook interface with several tabs open:

- Files**: Shows notebooks like Data.ipynb, Fasta.ipynb, Julia.ipynb, and Lorenz.ipynb.
- Running**: Shows processes like R.ipynb, iris.csv, lightning.json, and lorenz.py.
- Commands**: Shows recent commands like from lorenz import solve\_lorenz.
- Cell Tools**: Shows sliders for parameters sigma (10.00), beta (2.67), and rho (28.00).
- Output View**: Shows a 3D plot of the Lorenz attractor.
- Lorenz.ipynb**: The active notebook, containing text about the Lorenz system and differential equations, and code for plotting solutions.
- Terminal 1**, **Console 1**, **Data.ipynb**, **README.md**: Other open tabs.
- Python 3**: Kernel selection.

```

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

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((N, 3))

```