



# Workshop Housekeeping

## Python Fundamentals for Engineers and Manufacturers

Hosted By

SME Chapter 112  
Chicagoland



Workshop repo  
<http://bit.ly/2opFlZ6>



SME Chapter 430  
Greater Charleston



Where is all the info for this workshop?



<http://bit.ly/2opFlZ6>

All code is licensed under [The MIT License](#).

All written content is licensed under [Creative Commons Attribution 4.0 International Public License](#).

Workshops are just the beginning.

Workshop repo

<http://bit.ly/2opFlZ6>



- Open-source projects on [GitHub](#).
- Targeted webinars on [YouTube](#).
- Real-world, industry use-cases on [YouTube](#).  
Check out <http://bit.ly/2Flc0xG>.
- Industrial hardware/software [hackathons](#).
- Virtual assistance with SME student chapter projects.
- Live stream coding sessions on YouTube.

We ❤️ feedback and suggestions!

<http://bit.ly/2G42GXy>

**Robot Operating System**

<http://www.ros.org/>

**OpenCV**

<https://opencv.org/>

**CUDA**

<http://bit.ly/2F9tpkH>

**NVIDIA Jetson Boards**

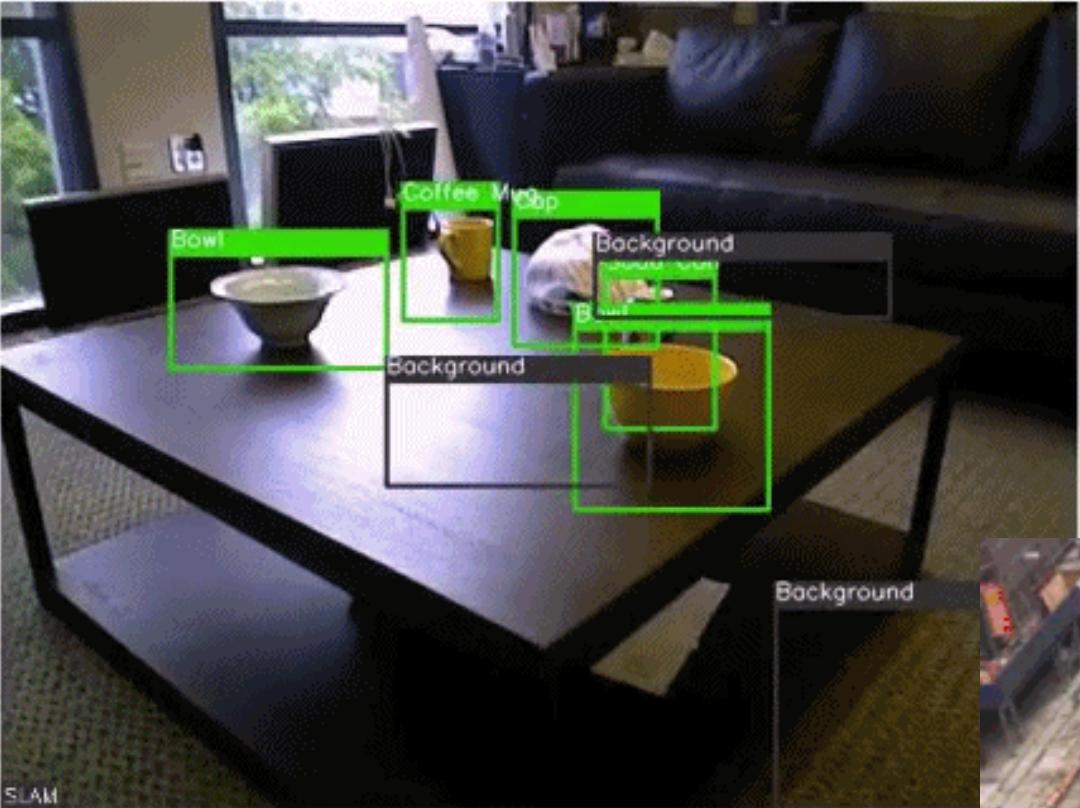
<http://bit.ly/2FOnx1f>



# PX4 – Flight Stack and Autonomous Middleware

<http://px4.io/>

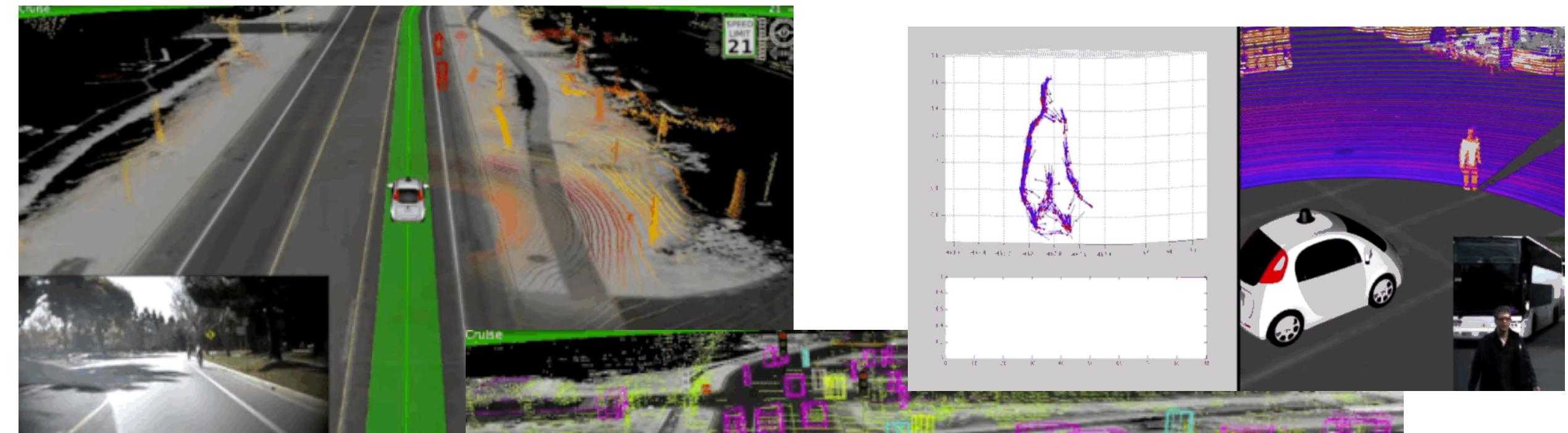




**OpenCV**  
<https://opencv.org/>

**NVIDIA Jetson Boards**  
<http://bit.ly/2FOnx1f>

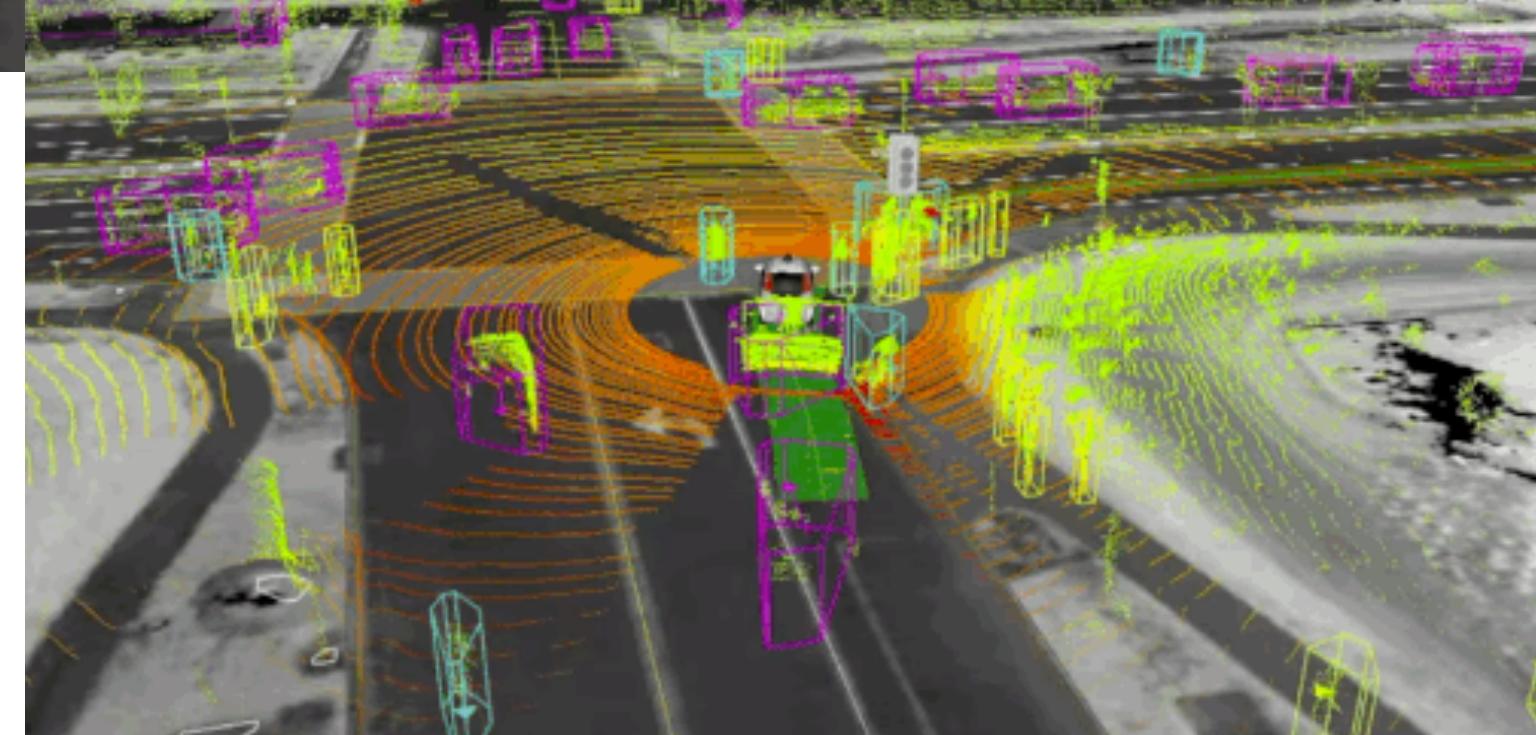




**CARLA**  
<http://www.carla.org/>

**Udacity Self-Driving Simulator**  
<http://bit.ly/2oKNKFY>

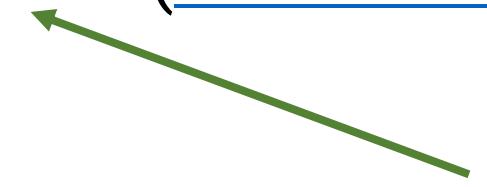
**Unity**  
<https://unity3d.com/>



- Introduction to [Git](#). Check out <https://bit.ly/2teAz5x>.
- Introduction to Anaconda, the Python interpreter and the [pip](#) ecosystem.
- Introduction to [Jupyter](#) Notebooks and [Visual Studio Code](#).
- Data types and objects.
- Reading and writing data.
- Control flow.
- Loops and iterables.
- Functions.
- Classes.
- Packages.

- Documentation.
- Testing.
- Debugging.
- Profiling.
- Network communication.
- Data exploration and visualization (with [NumPy](#) and [Pandas](#)).

- Concurrency and parallelism.
- Hardware description and verification ([HDL](#)).
- Computational geometry (pretty good video [here](#)).
- Image manipulation and machine vision.
- Neural networks ([CNN or ConvNet](#)).



Check out the [TensorFlow Playground](#).

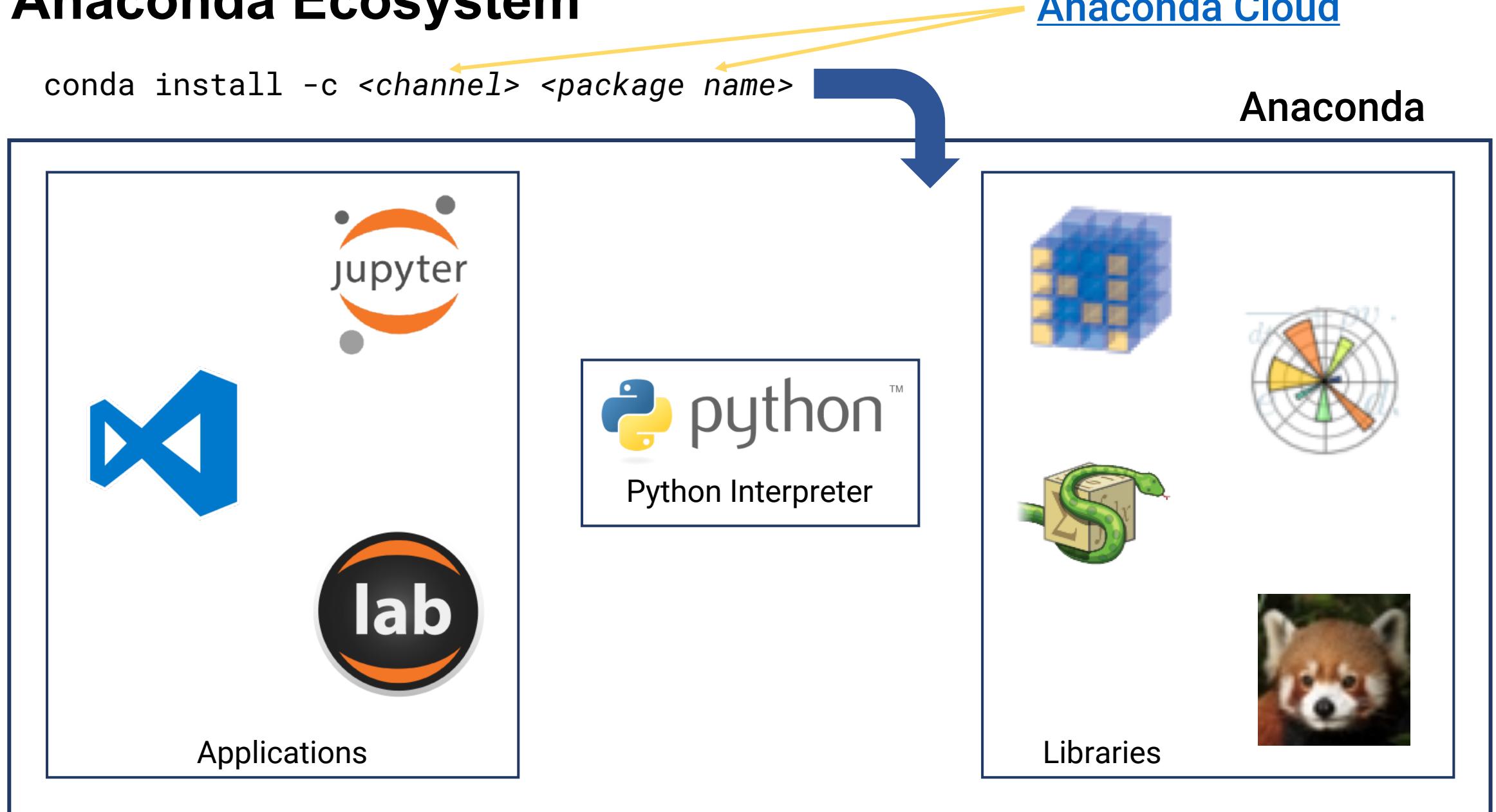
All SME Virtual Network projects will support **beginner friendly** labels on issues.

You can “claim” an issue by adding a comment to the issue which contains the word **claim**.

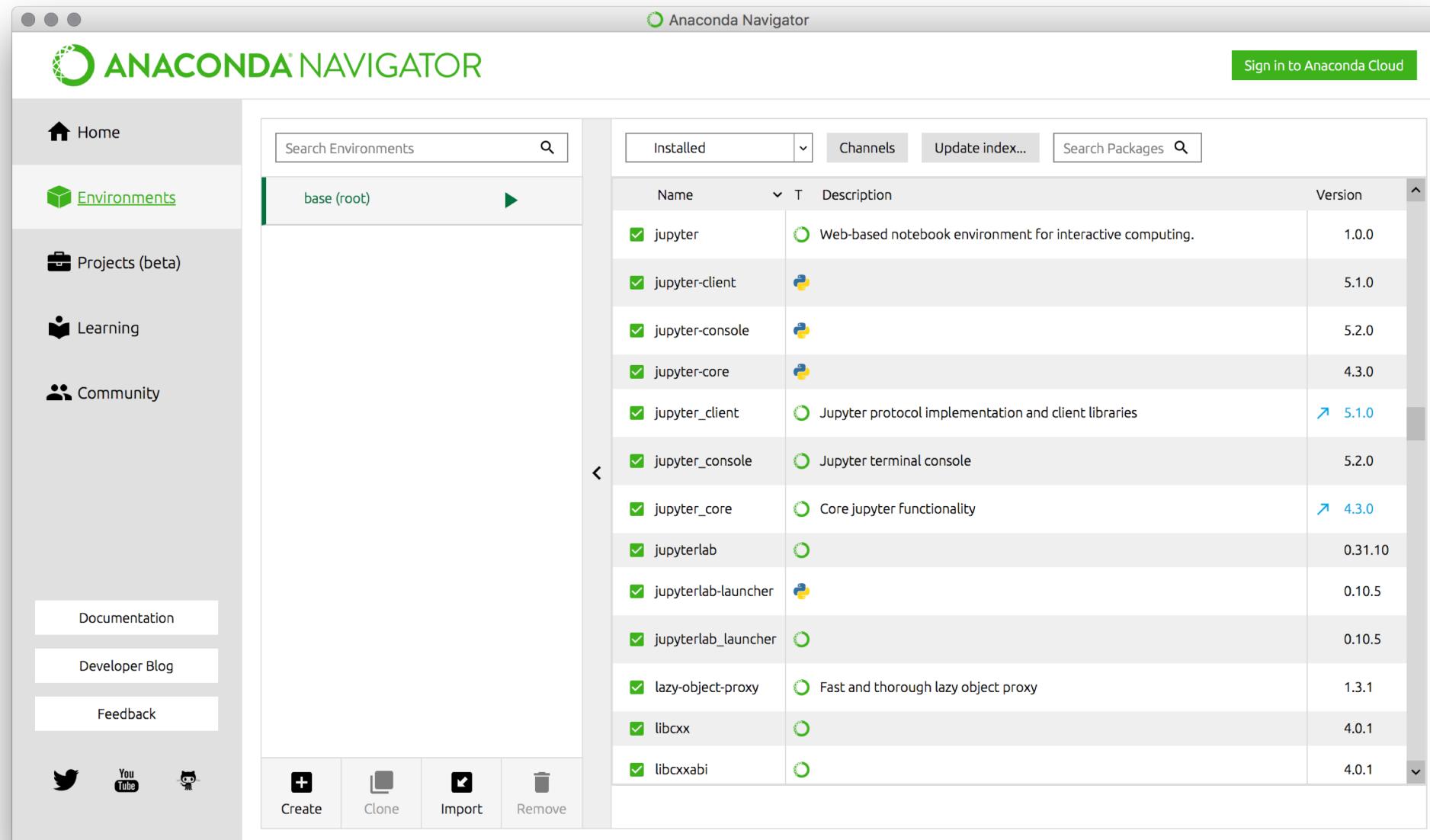
Want a list of other “beginner-friendly” Python open-source projects?

<http://bit.ly/2Fb2KUM>

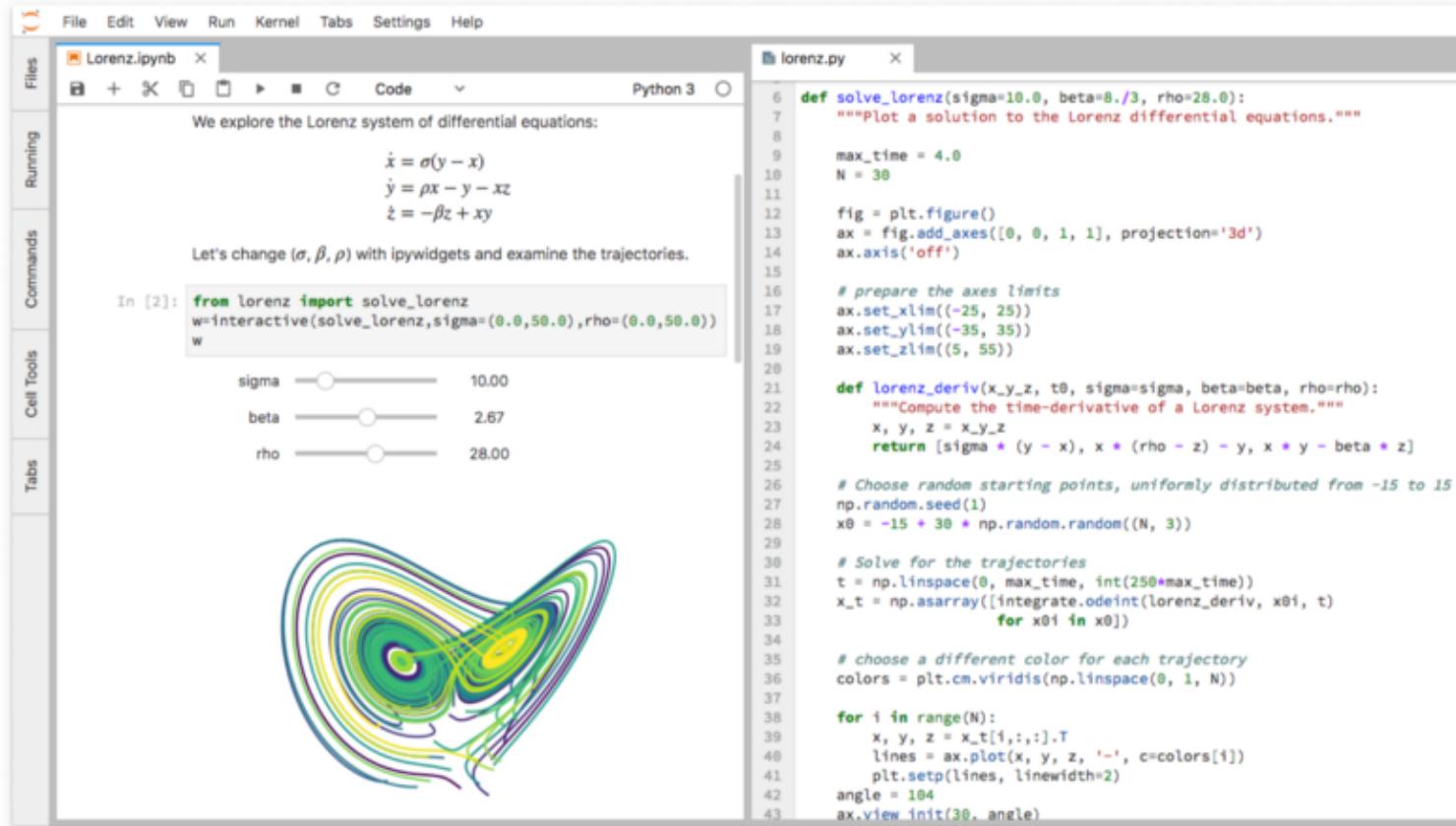
# Anaconda Ecosystem



# Python Virtual Environments



# JupyterLab



The screenshot shows the JupyterLab interface with two open files: `Lorenz.ipynb` and `lorenz.py`. The `Lorenz.ipynb` file contains a notebook cell with the following text and code:

```
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 change  $(\sigma, \beta, \rho)$  with ipywidgets and examine the trajectories.
```

```
In [2]: from lorenz import solve_lorenz  
w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))  
w
```

Below the code are three sliders for `sigma`, `beta`, and `rho` with initial values 10.00, 2.67, and 28.00 respectively. To the right, the `lorenz.py` file contains the following Python code:6 def solve\_lorenz(sigma=10.0, beta=8./3, rho=28.0):  
 """Plot a solution to the Lorenz differential equations."""  
8  
9 max\_time = 4.0  
N = 30  
11  
12 fig = plt.figure()  
13 ax = fig.add\_axes([0, 0, 1, 1], projection='3d')  
14 ax.axis('off')  
15  
16 # prepare the axes limits  
17 ax.set\_xlim((-25, 25))  
18 ax.set\_ylim((-35, 35))  
19 ax.set\_zlim((5, 55))  
20  
21 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]  
25  
26 # Choose random starting points, uniformly distributed from -15 to 15  
27 np.random.seed(1)  
x0 = -15 + 30 \* np.random(N, 3)  
28  
29 # Solve for the trajectories  
30 t = np.linspace(0, max\_time, int(250\*max\_time))  
x\_t = np.asarray([integrate.odeint(lorenz\_deriv, x0i, t)  
for x0i in x0])  
34  
35 # choose a different color for each trajectory  
36 colors = plt.cm.viridis(np.linspace(0, 1, N))  
37  
38 for i in range(N):  
 x, y, z = x\_t[:, :, i].T  
 lines = ax.plot(x, y, z, 'r-', c=colors[i])  
 plt.setp(lines, linewidth=2)  
 angle = 104  
 ax.view\_init(30, angle)

The visualization shows a 3D plot of the Lorenz attractor with multiple colored trajectories forming the characteristic butterfly shape.

Install JupyterLab into your Anaconda install:  
`conda install -c conda-forge jupyterlab`

The actual Anaconda Cloud entry is here:  
<http://bit.ly/2Facu5r>