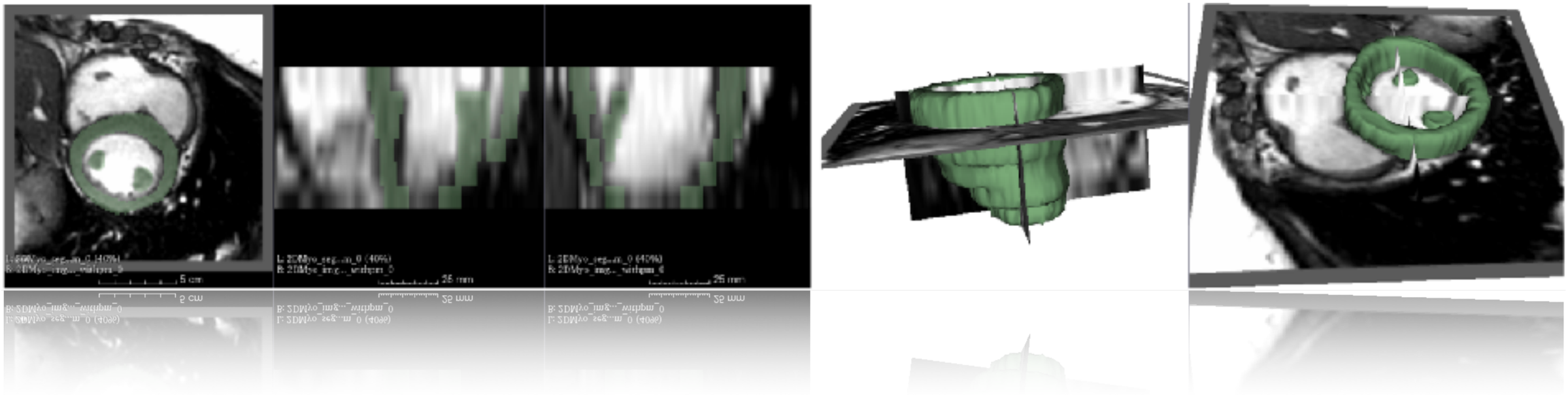


Departamento de Física Médica - Centro atómico Bariloche - IB

Instalación e Introducción a miniconda/anaconda

Ariel Hernán Curiale
ariel.curiale@cab.cnea.gov.ar

CONICET

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

Instituto Balseiro

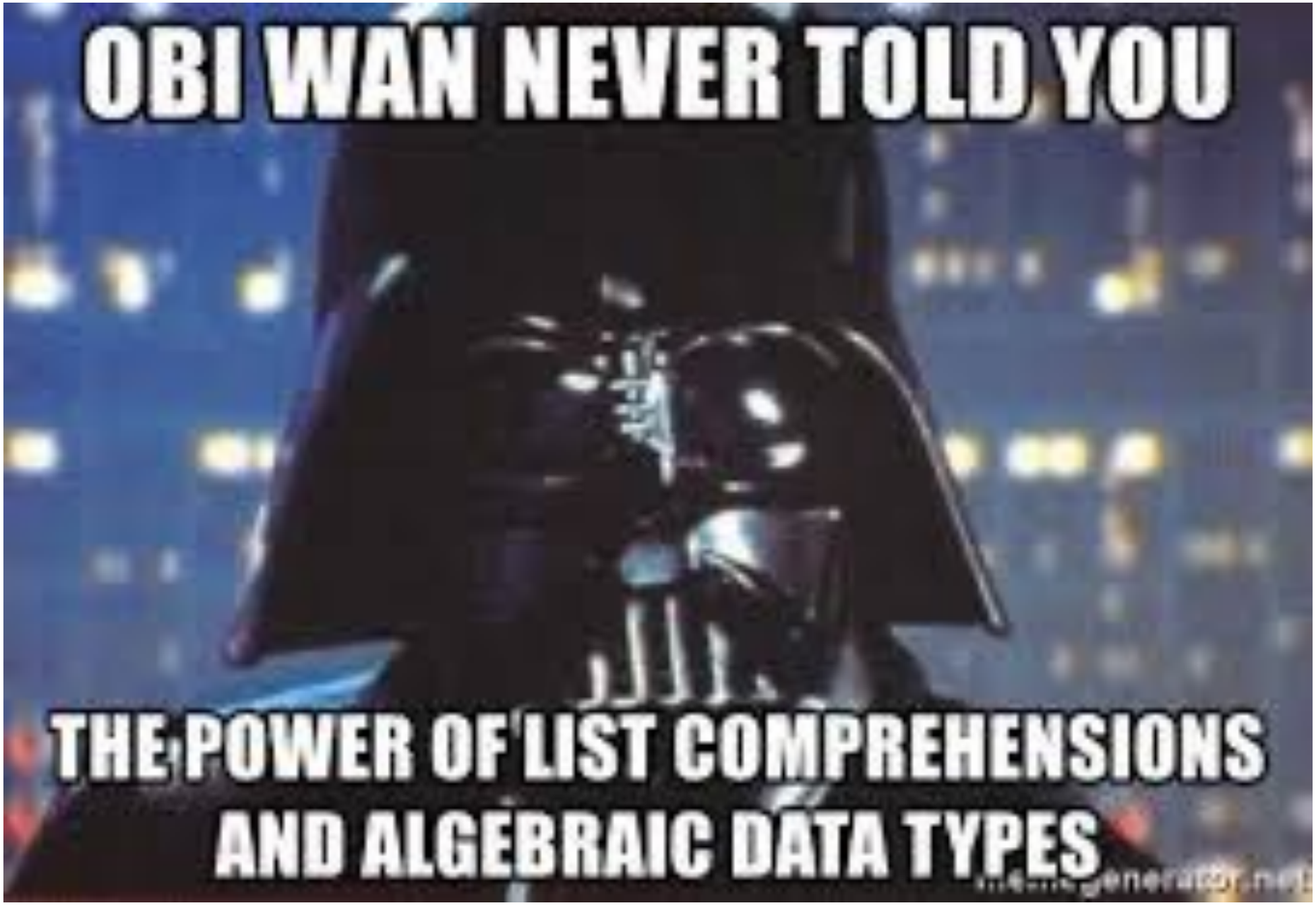
CNEA

Antes que nada

Erratas, bugs, comentarios,….

todos bienvenidos, y espero que los encuentren

Pero antes de comenzar con Miniconda ……


OBI WAN NEVER TOLD YOU
THE POWER OF LIST COMPREHENSIONS
AND ALGEBRAIC DATA TYPES

Listas por comprensión

- Nos quedo pendiente algo super útil y muy bonito

```
[ i for i in range(200) if i%2 ]
```

```
reverse_word = dict([(val,key) for key, val in word_index.items()])
```

```
layers = [l for l in model.layers[1:9] if 'conv' in l.name]
```

```
layers = [ l.weights[0].shape for l in model.layers[1:]
                                  if len(l.weights)>0 ]
```


FOR LOOP SUCKS
USE LIST COMPREHENSION
imgflip.com

Abrir el notebook 3 sección 2.9 para comentar sobre estructurar el código y seguir con manipulación de archivos

Anaconda - Miniconda

¿Qué es, y porque usarlo?

Download miniconda

- Miniconda (google download miniconda), si prefieren pueden usar anaconda o el OS con virtualenv o como sea: https://docs.conda.io/en/latest/miniconda.html

miniconda

- Algunos comando útiles (https://docs.conda.io/projects/conda/en/latest/commands.html):
  - **conda list**
  - **conda search**
  - **conda install**
  - **conda clean --all**
  - **conda update --all** o **conda update nombre_paquete**
  - **conda info --envs**
  - **conda create -n myenv python=3.6 numpy ….**
  - **conda remove —name myenv —all**
  - **conda -h** o **conda list -h** o **conda update -h ….**

miniconda

- **export PATH="/home/ariel/miniconda3/bin:$PATH"**

Instalando paquetes en el base

- Si ya esta instalado miniconda crear un entorno virtual: **conda create -n deep_learning python=3.7**
- Instalar los siguientes paquetes (si se creo un entorno activarlo antes): **conda install numpy matplotlib scipy scikit-image scikit-learn ipython imageio**
  - GPU: **conda install -c anaconda tensorflow-gpu**
  - PC: **conda install -c anaconda tensorflow**
  - Versión más completa (todos los backends): **conda install keras**

Instalando paquetes en el base

- Paquetes para trabajar
  - Consolas:
    conda install ipython
    conda install jupyter-console (instala el notebook)
  - IDES (entornos gráficos) estilo matlab:
    conda install spyder
    conda install -c conda-forge jupyterlab
    conda install -c jupyter (notebooks, instala la consola)
- Existe una GUI para instalar cosas:
  conda install -c anaconda anaconda-navigator

Instalando paquetes en el base

- Otro paquetes útiles

  conda install -c anaconda natsort

  conda install -c bioconda medpy

  conda install -c conda-forge ipdb

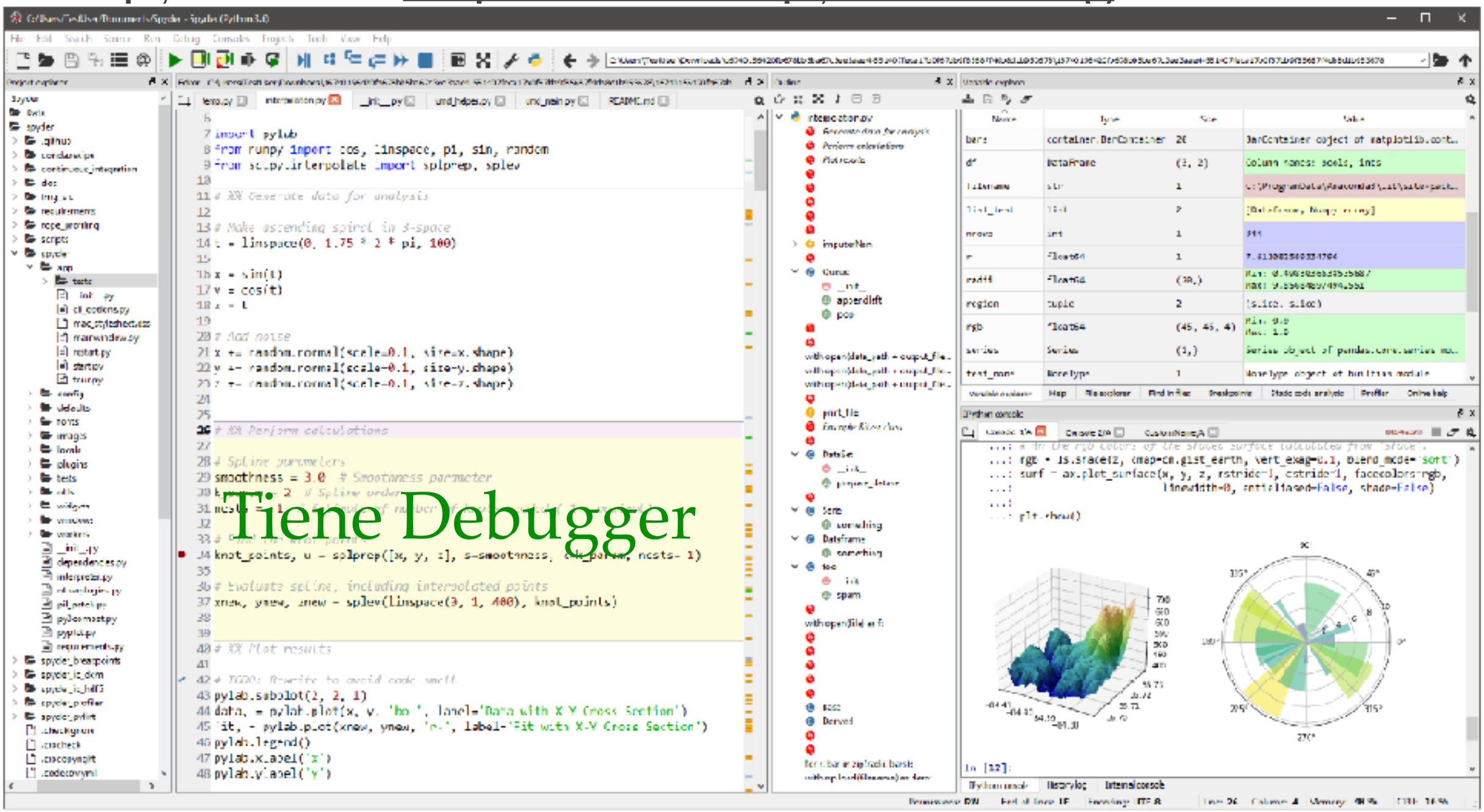
Instalando paquetes en el base

- Solo cuando no encontremos los paquetes en conda, anaconda, conda-forge podemos instalar los paquetes con pip

```
pip install <paquete>
```

IDE

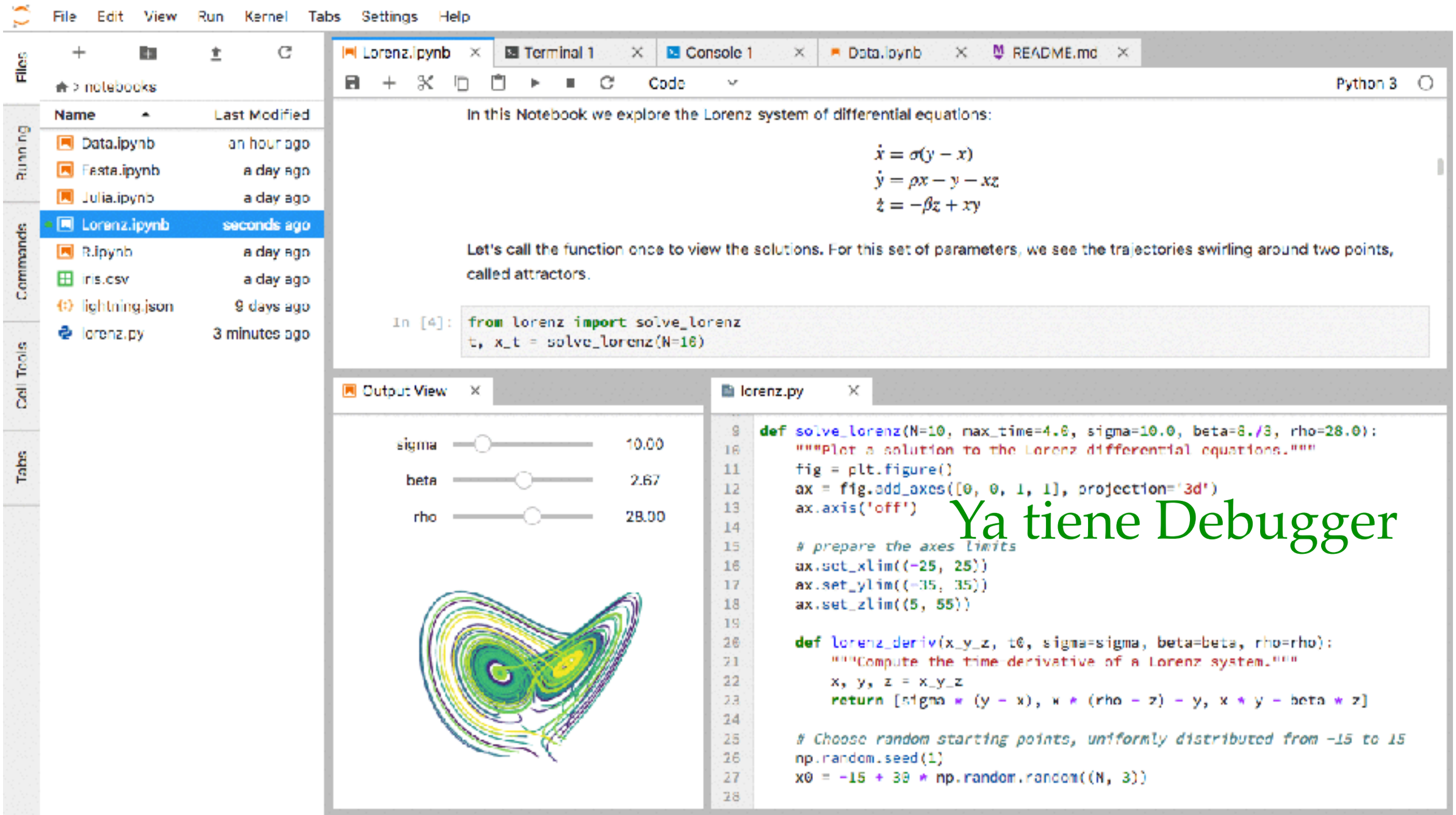
- Spyder IDE: https://www.spyder-ide.org



Tiene Debugger

IDE

- JupyterLab IDE: https://jupyterlab.readthedocs.io/en/stable/



Editor favorito + Consola

- Ipython: editor de texto + ipython

Tiene debugger de consola (ipdb)

Editor favorito + Consola

- Jupyter console: editor de texto + juyter console

Tiene debugger de consola (ipdb)

Local console kernel remoto

- Con Jupyter console podemos levantar de forma local una consola que realmente ejecuta en un servidor

```
1. Averiguamos el directorio donde corre jupyter
[server] $ jupyter --runtime-dir
/run/user/1000/jupyter

2. Iniciamos el ipython
[server] $ ipython kernel
[IPKernelApp] To connect another client to this kernel, use:
[IPKernelApp] --existing kernel-1234.json

3. Copiamos el .json
[client] $ scp server:/run/user/1000/jupyter/kernel-1234.json ./

4. Lanzamos la consola en el clinete de ipython

4.1 console
[client] $ jupyter console --existing ./kernel-1234.json --ssh user@server

4.2 qtconsole
[client] $ ipython qtconsole --existing ./kernel-1234.json --ssh user@server
```

Editor favorito + consola + kernel remoto

```
"""
File: 07_TrainingNetworks_Part1.py
Author: Ariel Hernán Curiale
Email: curiale@gmail.com
Github: https://gitlab.com/Curiale
Description:
    La implementación la saque del curso de Fei-Fei Li. Vamos a ver como quedan
    en las capas internas la activación de las neuronas si hacemos una
    inicialización pequeña normal.
"""

import numpy as np
import matplotlib.pyplot as plt



def small_normal(fan_in, fan_out):
    W = np.random.randn(fan_in, fan_out) * 1e-2
    return W

def glorot_normal(fan_in, fan_out):
    '''Xavier initialization Glorot et al., 2010'''
    W = np.random.randn(fan_in, fan_out) * (2/ np.sqrt(fan_in+fan_out))
    return W
def glorot_uniform(fan_in, fan_out):
    '''Xavier initialization Glorot et al., 2010'''
    limit_uniform =  [-np.sqrt(6/(fan_in+fan_out)), np.sqrt(6/(fan_in+fan_out))]
    W = (limit_uniform[1] - limit_uniform[0]) * np.random.random((fan_in, fan_out)) + limit_uni
form[0]
    return W
def he_normal(fan_in, fan_out):
    '''He et al., 2010'''
    W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in/2)
    return W


n_layers = 10
n_neurons=500
n_samples = 10000
D = np.random.randn(n_samples, n_neurons)
n_neurons_by_layer = [n_neurons] * n_layers
act = {'relu': lambda x:np.maximum(0,x), 'tanh': lambda x:np.tanh(x)}
# Como todas las capas tienen la misma cantidad de neuronas puedo agrupar todo
# en una gran matriz
Hs = np.zeros((n_samples, n_neurons, n_layers))
for i in range(n_layers):
    X = D if i==0 else Hs[..., i-1]
    fan_in = X.shape[-1] # neuronas de la capa anterior
    fan_out = n_neurons_by_layer[i] # neuronas de la capa i
    # Inicializamos con media cero y std 1e-2
    #  W = small_normal(fan_in, fan_out)
    #  # glorot uniform
    #  W = glorot_uniform(fan_in, fan_out)
    #  #glort normal
    W = glorot_normal(fan_in, fan_out)
    # He et. al 2015
    #  W = he_normal(fan_in, fan_out)

    H = X.dot(W) # producto niterno
    H = act['relu'](H)
    Hs[..., i] = H

NORMAL  07_TrainingNetworks_Part1.py[+]   python  utf-8[unix]  29%  58/195 ln :  1  E:[148(41)]
```

```
ariel@Goku: ~ (python3.6)
Executing transaction: done
ariel@Goku:~$
    miniconda3
Applications        Downloads           Logos               Phd                 Scripts
Datos               Dropbox             Movies              Pictures            Sites
Desktop             GoogleDrive         Music               Proyectos           kernel-7878-ssh.j
son
Docencia            Latex_resources     OneDrive            Public              kernel-7878.json
ariel@Goku:~$ rm kernel-7878-ssh.json
ariel@Goku:~$ jupyter console --existing ./kernel-7878.json --ssh 10.73.27.83
[ZMQTerminalIPythonApp] Forwarding connections to 127.0.0.1 via 10.73.27.83
SSH Password for 10.73.27.83:
[ZMQTerminalIPythonApp] To connect another client via this tunnel, use:
[ZMQTerminalIPythonApp] --existing kernel-7878-ssh.json
Jupyter console 6.0.0

Python 3.7.3 (default, Mar 27 2019, 22:11:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.7.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:

ipython kernel
NOTE: When using the 'ipython kernel' entry point, Ctrl-C will not work.

To exit, you will have to explicitly quit this process, by either sending
"quit" from a client, or using Ctrl-\ in UNIX-like environments.

To read more about this, see https://github.com/ipython/ipython/issues/2049


To connect another client to this kernel, use:
    --existing kernel-7878.json
OMP: Info #212: KMP_AFFINITY: decoding x2APIC ids.
OMP: Info #210: KMP_AFFINITY: Affinity capable, using global cpuid leaf 11 info
OMP: Info #154: KMP_AFFINITY: Initial OS proc set respected: 0-11
OMP: Info #156: KMP_AFFINITY: 12 available OS procs
OMP: Info #157: KMP_AFFINITY: Uniform topology
OMP: Info #179: KMP_AFFINITY: 1 packages x 6 cores/pkg x 2 threads/core (6 total cores)
OMP: Info #214: KMP_AFFINITY: OS proc to physical thread map:
OMP: Info #171: KMP_AFFINITY: OS proc 0 maps to package 0 core 0 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 6 maps to package 0 core 0 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 1 maps to package 0 core 1 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 7 maps to package 0 core 1 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 2 maps to package 0 core 2 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 8 maps to package 0 core 2 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 3 maps to package 0 core 3 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 9 maps to package 0 core 3 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 4 maps to package 0 core 4 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 10 maps to package 0 core 4 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 5 maps to package 0 core 5 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 11 maps to package 0 core 5 thread 1
OMP: Info #250: KMP_AFFINITY: pid 7878 tid 7878 thread 0 bound to OS proc set 0
OMP: Info #250: KMP_AFFINITY: pid 7878 tid 8225 thread 1 bound to OS proc set 1
OMP: Info #250: KMP_AFFINITY: pid 7878 tid 8226 thread 2 bound to OS proc set 2
OMP: Info #250: KMP_AFFINITY: pid 7878 tid 8227 thread 3 bound to OS proc set 3
OMP: Info #250: KMP_AFFINITY: pid 7878 tid 8228 thread 4 bound to OS proc set 4
OMP: Info #250: KMP_AFFINITY: pid 7878 tid 8229 thread 5 bound to OS proc set 5
```

Tiene debugger de consola (ipdb)

Jupyter notebooks

Por último

- En ipython y jupyter existen los magics commands. Son comando que comienzan con %

  %paste
  %run
  %timeit
  %who
  %reset
  %autoreload
  .....

  https://ipython.readthedocs.io/en/stable/interactive/magics.html

¿Mostramos como usar ipdb?