CURSO NO PDP: Python

Dia 1: Introducción a Python y entornos de desarrollo

https://github.com/fiorelacl/curso_cah

IGP Instituto Geofísico del Perú

- Instructores:
  - Fiorela Castillón (fcastillon@igp.gob.pe)
  - Miguel Saavedra(msaavedra@igp.gob.pe)

- Fechas:
  - ❑30/04/2024 : Intro a Python
  - ❑07/05/2024 : Fundamentos de Programación
  - ❑15/05/2024 : Numpy/Pandas
  - ❑22/05/2024 : Matplotlib
  - ❑06/06/2024 : Xarray I
  - ❑12/06/2024 : Xarray II
  - ❑19/06/2024 : Cartopy
  - ❑26/06/2024 : Aplicaciones / SCAHpy

- Horario:
  - 2 pm – 5 pm

- Lugar:
  - Presencial> Sala SUM – Mayorazgo
  - Virtual> Google Meet

Ilustración adaptada de: Allison Horst

- Tareas:
  - Cada sesión tendrá tareas cortas individuales que se subirán al Google Classroom.

- Proyecto Final:
  - Grupos 2 personas
  - Generar un jupyter-notebook a manera de un pequeño reporte/informe con la aplicación de Python a su área de trabajo o interés.
  - Fecha de Entrega: 27 al 1 Julio
  - Se sube al Google Classroom

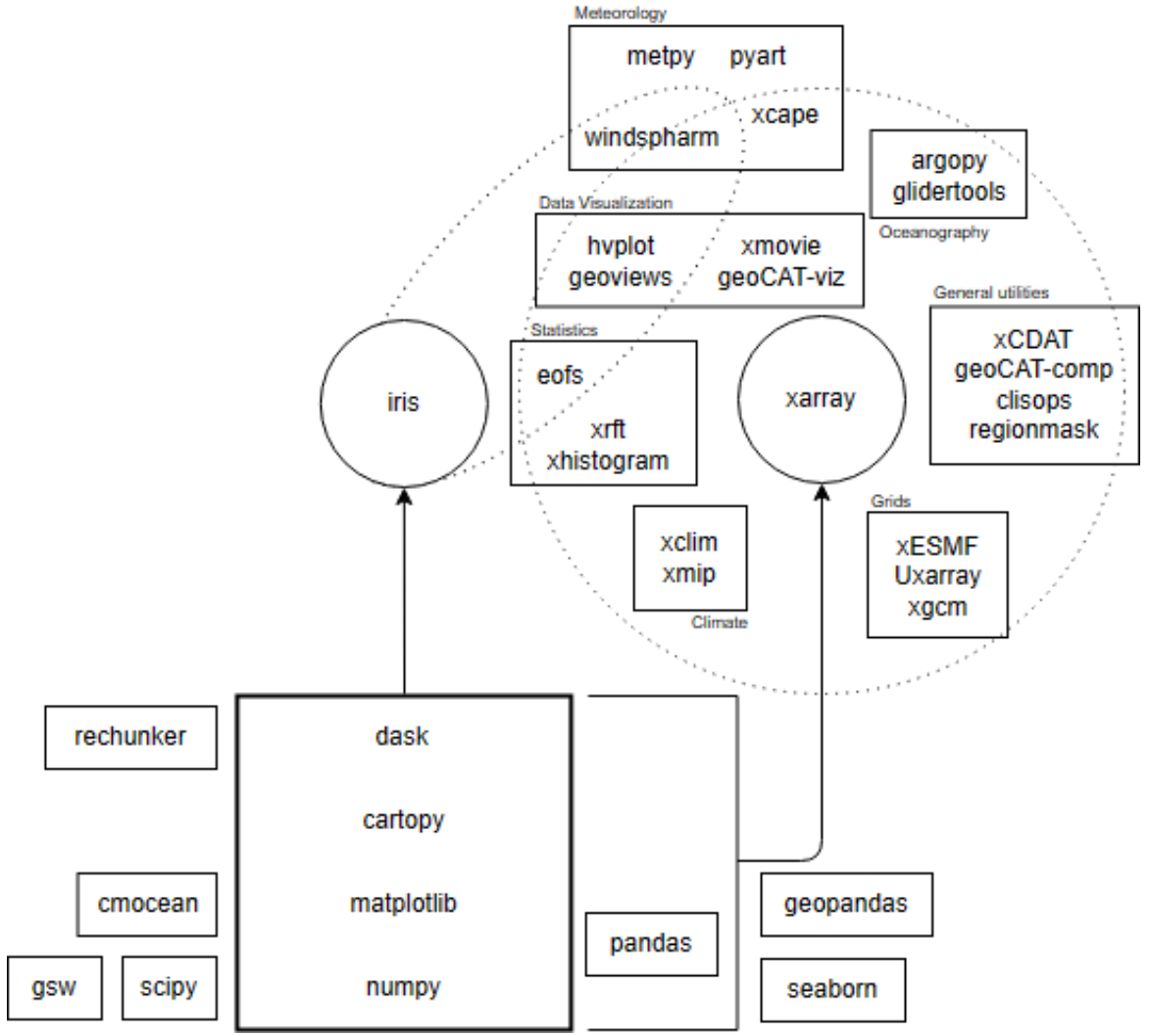
Ilustración adaptada de: Allison Horst

Agenda

- ¿Qué es Python?
- Popularización en OA
- Instalación y entornos
- JupyterLab
- Sintaxis y Estructura de Datos

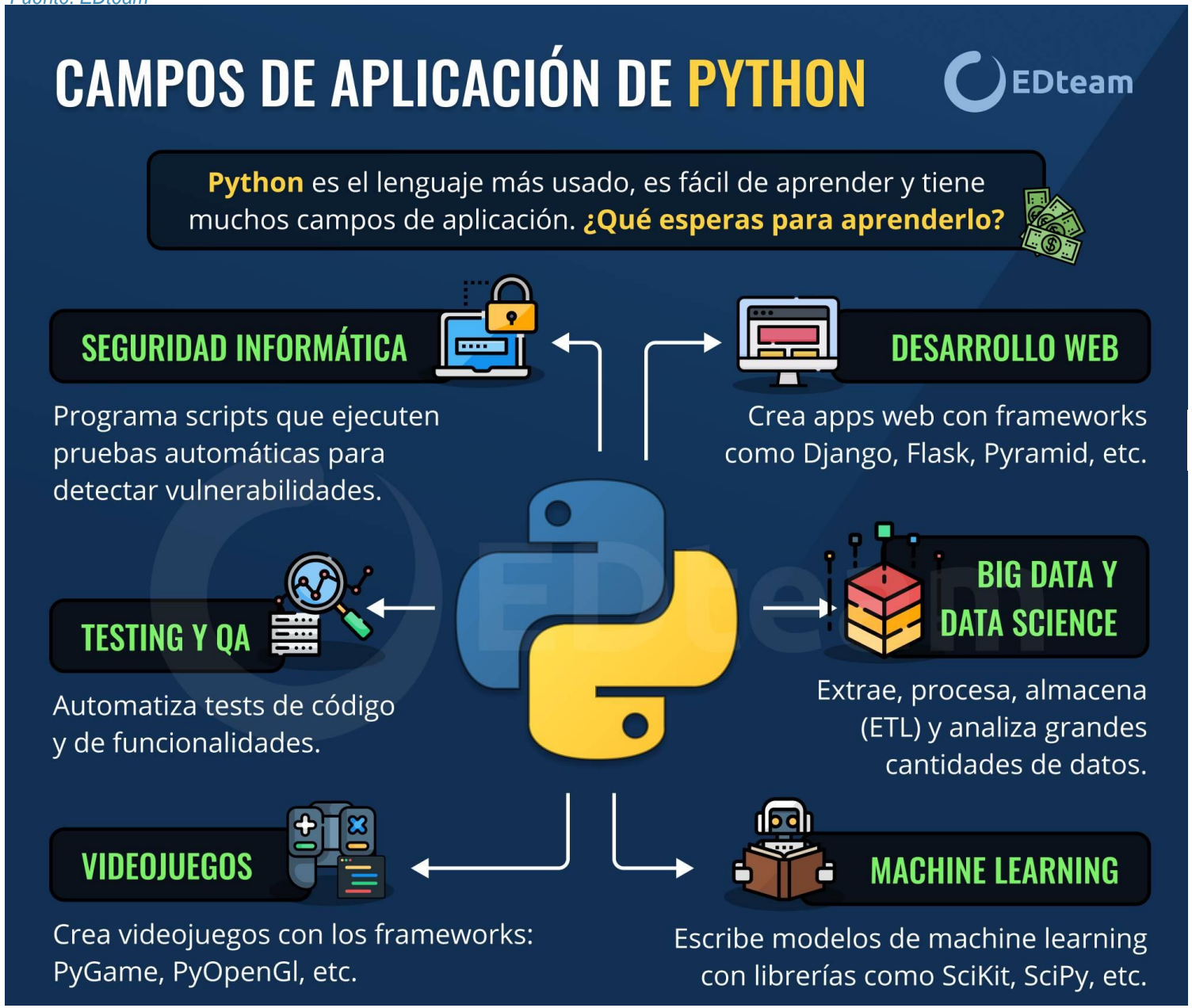
PYTHON

- Código abierto (Open Source)
- Lenguaje de programación de alto nivel.
- Lenguaje interpretado
- Tipado dinámico
- Sensible a Mayúsculas
- Orientado a Objetos
- Sencillo de aprender
- Interfaz con otros lenguajes (Fortran, C)
- Multiplataforma
- Amplia gama de librerías
- Comunidad amplia y activa



Referencia: Python programming and visualization for scientists , 2nd edition / Alex J. DeCaria ; Grant W. Petty.

Fuente: EDteam



PANGEO

A community platform for Big Data geoscience

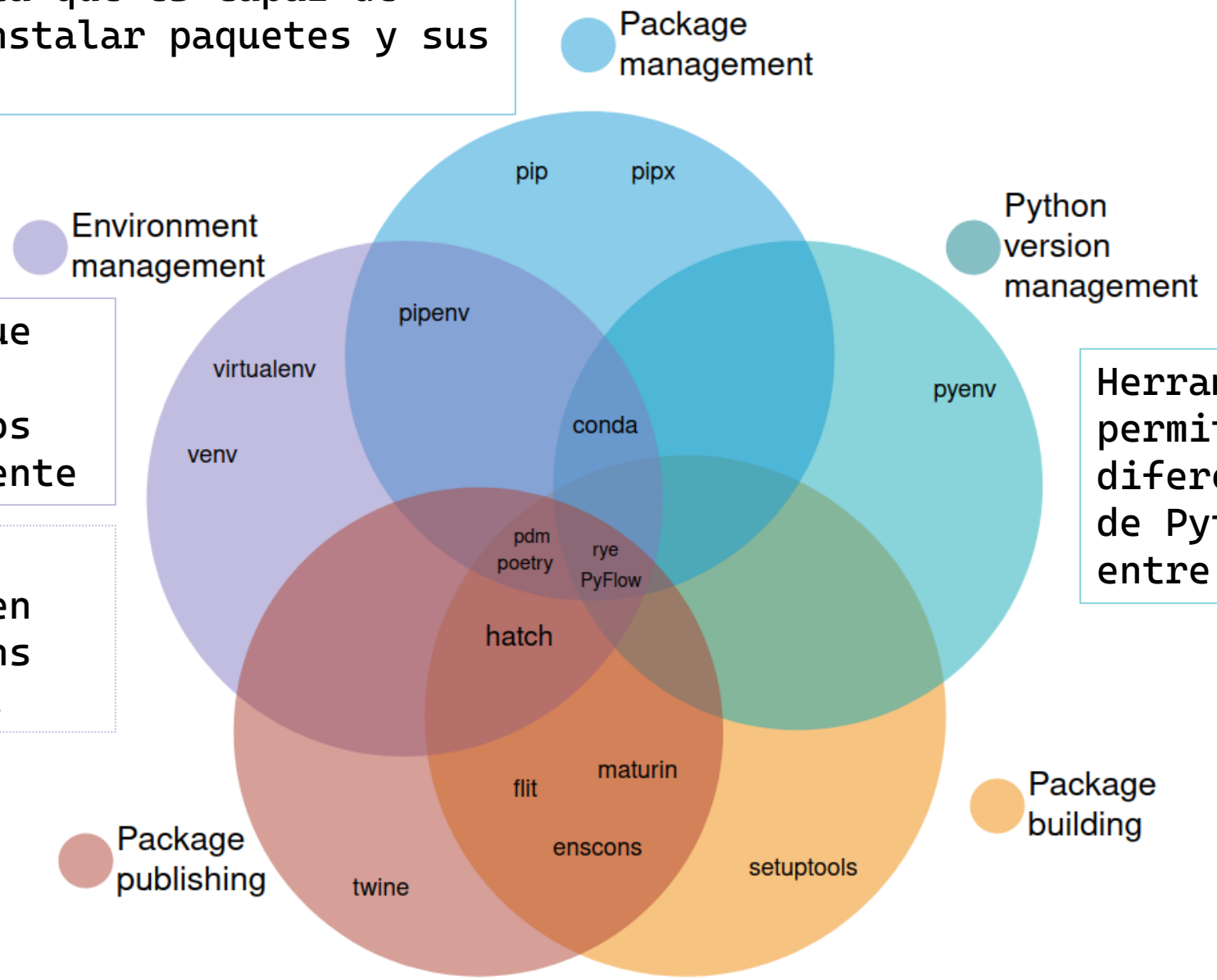
PROJECT PYTHIA

¿Python? ¿Jupyterlab?

- ✓ Python es el lenguaje de programación que realizará todos los cálculos.

- ✓ Entornos de desarrollo Integrado(IDE), entorno de desarrollo web y en la nube son interfaces que gracias a sus herramientas permiten usar Python de manera más sencilla.

Una herramienta que es capaz de descargar e instalar paquetes y sus dependencias

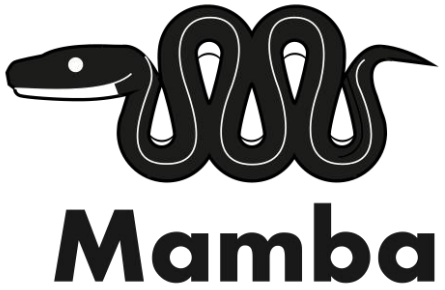


Una herramienta que permite crear y gestionar (entornos virtuales) fácilmente

Herramientas que te permite instalar diferentes versiones de Python y cambiar entre ellas fácilmente

Ej. Diferentes proyectos requieren diferentes versions del mismo paquete.

Fuente: Anna-Lena Popkes



Mamba Installation — documentation

Reimplementación de *conda* que acelera la creación de entornos virtuales

1. GitHub - conda-forge/miniforge: A conda-forge distribution.

OS	Architecture	Download
Linux	x86_64 (amd64)	Miniforge3-Linux-x86_64
Linux	aarch64 (arm64) (**)	Miniforge3-Linux-aarch64
Linux	ppc64le (POWER8/9)	Miniforge3-Linux-ppc64le
OS X	x86_64	Miniforge3-MacOSX-x86_64
OS X	arm64 (Apple Silicon) (***)	Miniforge3-MacOSX-arm64
Windows	x86_64	Miniforge3-Windows-x86_64

(**) For Raspberry PI that include a 64 bit processor, you must also use a 64-bit operating system such as Raspberry Pi OS 64-bit or Ubuntu for Raspberry PI. The versions listed as "System: 32-bit" are not compatible with the installers on this website.

(***) Apple silicon builds are experimental and haven't had testing like the other platforms

```
wget enlace_a_Miniforge3 # Para descargar en el cluster por ejemplo
bash Miniforge3-Linux-x86_64.sh -b  # or similar for other installers for unix platforms
```

Comandos útiles en *mamba*

```
mamba activate nameofmyenv
mamba deactivate
mamba list env
mamba list

mamba create -n nameofmyenv <list of packages>
mamba create -n myjlabenv jupyterlab -c conda-forge
mamba activate myjlabenv  # activate our environment

mamba install bqplot
mamba install "matplotlib>=3.5.0" cartopy


mamba remove -n nameofmyenv bqplot
mamba env remove -n nameofmyenv
mamba env create --file environment.yml -n nameofmyenv
```

environment.yml

```
name: nameofmyenv
channels:
  - conda-forge
dependencies:
  - xarray
  - dask
  - cartopy
  - numpy
  - pandas
  - matplotlib
  - jupyterlab
  - pip
  - pip:
     - datetime
```

Instalar MAMBA en el cluster

Descarga: `curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-$(uname)-$(uname -m).sh"`

Instalación: `bash Miniforge3-$(uname)-$(uname -m).sh`

Salir de la sesión y volver a ingresar (*logout*)
conda config --set auto_activate_base false (para que no se active el entorno base de manera automática en cada login)
conda deactivate (para salir del entorno 'base' o de cualquier entorno que esté activo)

```
mamba activate nameofmyenv
mamba deactivate
```

```
mamba list env
mamba list
```

```
mamba create -n nameofmyenv <list of packages>
mamba create -n myjlabenv jupyterlab -c conda-forge
mamba activate myjlabenv  # activate our environment
```

```
mamba install bqplot
mamba install "matplotlib>=3.5.0" cartopy
```

```
mamba remove -n nameofmyenv bqplot
mamba env remove -n nameofmyenv
```

Documentación mamba: https://mamba.readthedocs.io/en/latest/

Script .sh para lanzar un jupyterlab en el cluster

```
#!/bin/bash -l

#SBATCH --nodes=1
#SBATCH --partition=mpi_long2
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH -w, --nodelist=n15
#SBATCH --time=0-08:00:00
#SBATCH --mem=12G
#SBATCH --job-name=py_val
#SBATCH --output=jupyter-log-%J.txt
# get tunneling info

ipnport=$(shuf -i8000-9999 -n1)
ipnip=$(hostname -i)
user=$USER
host='10.10.90.12'
EXT_PORT='2123'
EXT_IP='190.187.237.250'

mamba activate pyval

## print tunneling instructions to jupyter-log-{jobid}.txt
echo -e "
           Copy/Paste this in your local terminal to ssh tunnel with remote
           -----------------------------------------------------------------
           ssh -N -L 8889:$ipnip:$ipnport -L 8787:$ipnip:8787 $user@$host
           -----------------------------------------------------------------
           From outside the network
           ssh -N -L 8889:$ipnip:$ipnport -L 7373:$ipnip:7373 -p $EXT_PORT $user@$EXT_IP

           Then open a browser on your local machine to the following address
           -----------------------------------------------------------------
           localhost:8889  (prefix w/ https:// if using password)
           -----------------------------------------------------------------
           "
## start an ipcluster instance and launch jupyter server
jupyter lab --no-browser --port=$ipnport --ip=$ipnip
```

Importante: Activar el entorno de python que van a usar. En este caso se llama pyval

Cuando se lanza el trabajo se genera un archivo con el nombre colocado en output: ejm: *jupyter-log-86451.txt*

El contenido del archivo es similar a este:

```
Copy/Paste this in your local terminal to ssh tunnel with remote
-----------------------------------------------------------------
ssh -N -L 8889:172.20.0.15:8740 -L 8787:172.20.0.15:8787
fcastillon@10.10.90.12
-----------------------------------------------------------------
From outside the network
ssh -N -L 8889:172.20.0.15:8740 -L 7373:172.20.0.15:7373 -p 2123
fcastillon@190.187.237.250

Then open a browser on your local machine to the following address
-----------------------------------------------------------------
localhost:8889  (prefix w/ https:// if using password)
-----------------------------------------------------------------
```

Se debe copiar y pegar en otro terminal los inicios de sesión, dependiendo de si se está trabajando con la red local o externa, se coloca la contraseña de su cuenta y luego en un navegador se copia y pega: *localhost:8889*

Nota: La primera vez que se inicia sesión se puede definir el paswd, en el archivo .txt generado habrá un enlace extenso con la palabra token, se copia-pega y luego se coloca la contraseña que se empleará cada que se requiera.

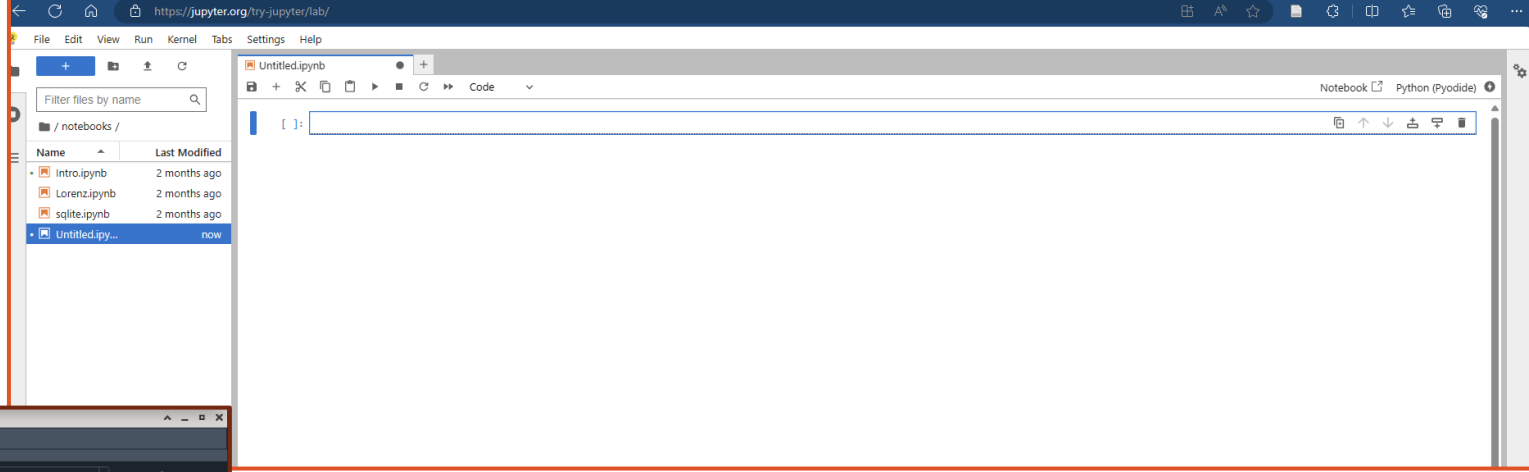
Práctica

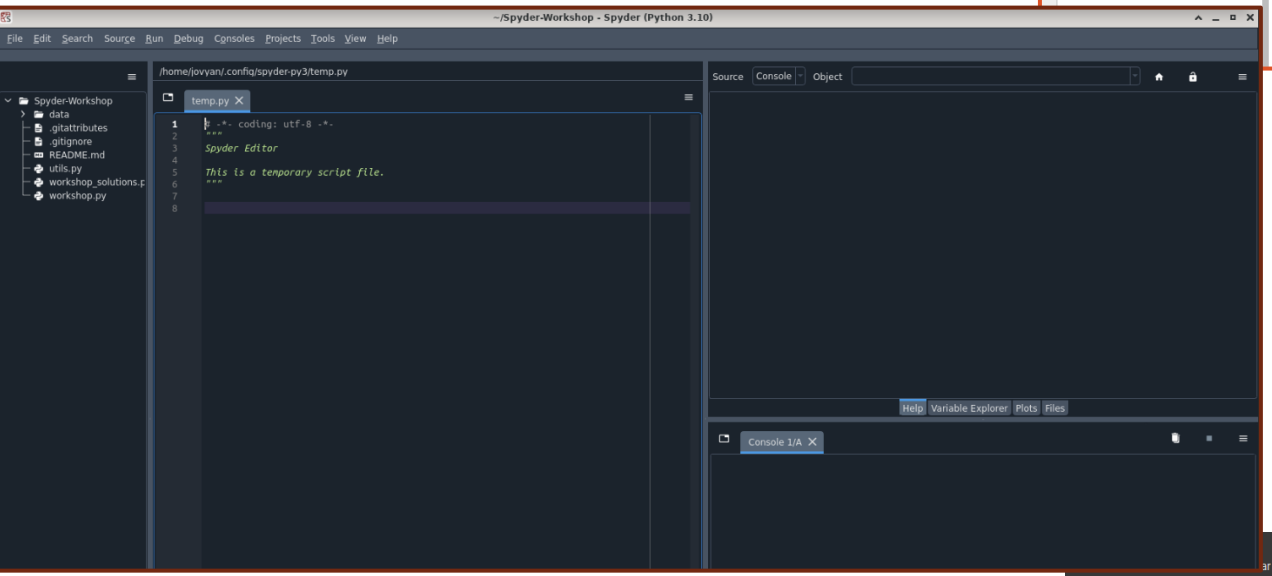
Crear el entorno nº1 llamado: *jupylab_env* que contenga Python 3.8 y los paquetes numpy matplotlib y jupyterlab

Crear el entorno nº2 llamado: *spyder_env* que contenga los siguientes paquetes spyder numpy pandas y matplotlib

Crear el entorno nº3 a partir de un archivo *environment.yaml: el nombre del entorno debe ser yaml_env* que contenga Python 3.10 y los paquetes xarray numpy matplotlib y jupyterlab del canal conda-forge y el paquete datetime con del canal pip.

JupyterLite

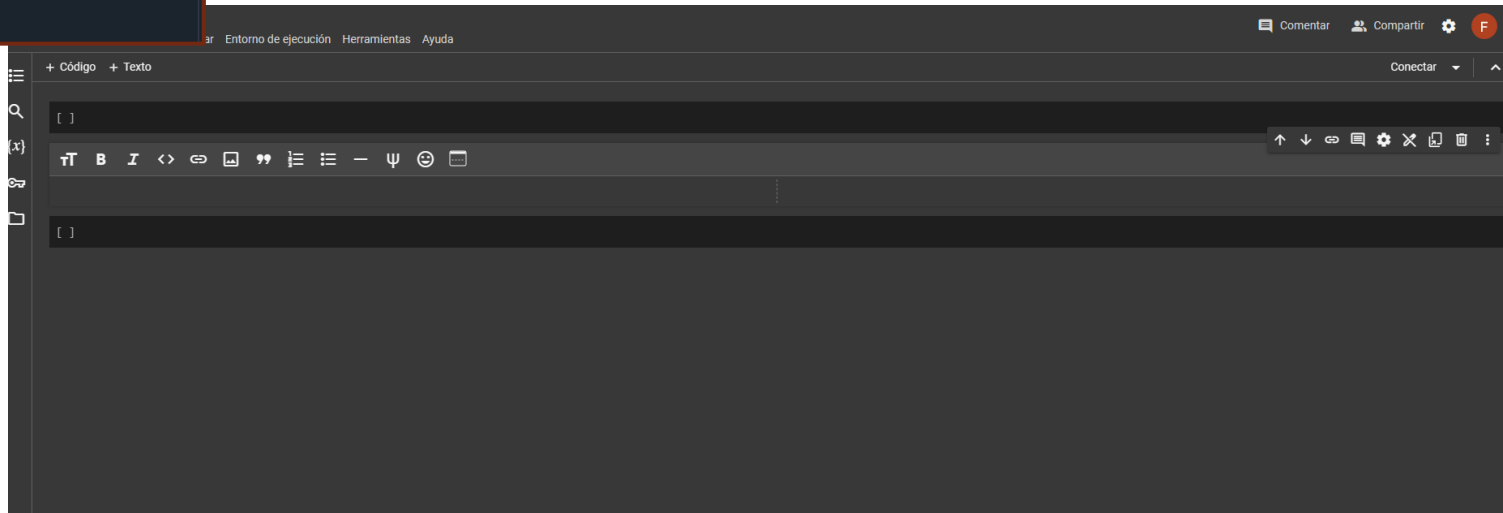




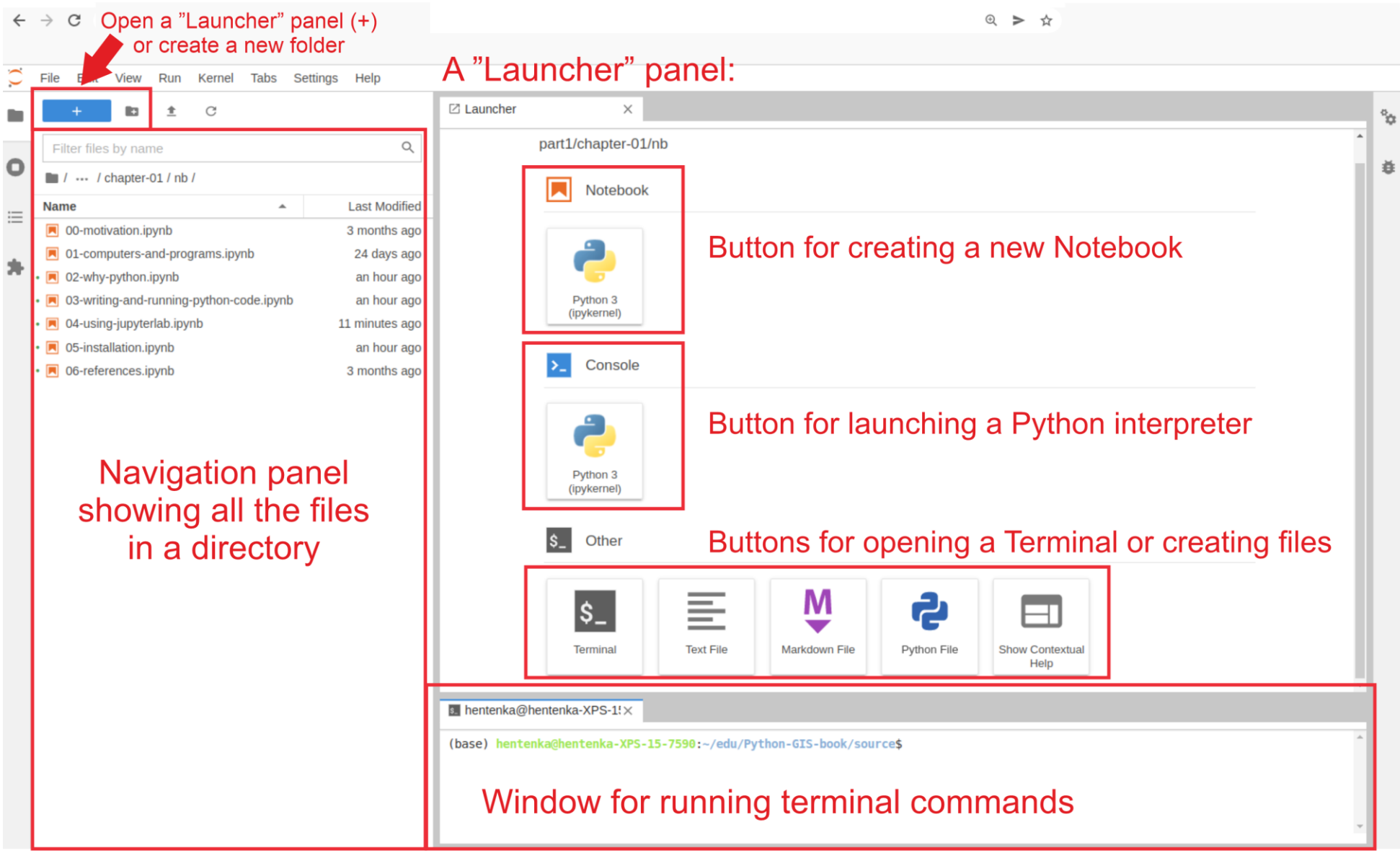
spyder | The Scientific Python Development Environment

Spyder Binder

colab



Entorno Jupyterlab



Algunos comandos y/o consideraciones

Jupyterlab_Cheat_Sheet.pdf (datacamp.com)

The JupyterLab Interface — JupyterLab 4.2.0rc0 documentation

File Edit View Run Kernel Tabs Settings Help

Files Running Commands Cell Tools Tabs

> notebooks

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

Lorenz.ipynb | Terminal 1 | Console 1 | Data.ipynb | README.md

Code — Python 3

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.

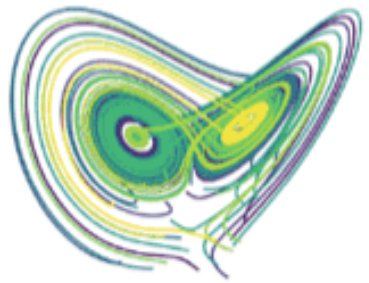
In [4]:
```
from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

Output View

sigma 10.00

beta 2.67

rho 28.00



lorenz.py

```
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))
```

Tipos de Archivos

Python Script
.py

FileInfo.com Example.py

```python
import sys
import time
from Phidget22.Devices.Gyroscope import *
from Phidget22.PhidgetException import *
from Phidget22.Phidget import *
from Phidget22.Net import *

try:
    ch = Gyroscope()
except RuntimeError as e:
    print("Runtime Exception %s" % e.details)
    print("Press Enter to Exit...\n")
    readin = sys.stdin.read(1)
    exit(1)

def GyroscopeAttached(self):
    try:
        attached = self
        print("\nAttach Event Detected (Information Below)")
        print("=========================================")
        print("Library Version: %s" % attached.getLibraryVersion())
        print("Serial Number: %d" % attached.getDeviceSerialNumber())
        print("Channel: %d" % attached.getChannel())
        print("Channel Class: %s" % attached.getChannelClass())
        print("Channel Name: %s" % attached.getChannelName())
        print("Device ID: %d" % attached.getDeviceID())
        print("Device Version: %d" % attached.getDeviceVersion())
        print("Device Name: %s" % attached.getDeviceName())
        print("Device Class: %d" % attached.getDeviceClass())
        print("\n")

    except PhidgetException as e:
        print("Phidget Exception %i: %s" % (e.code, e.details))
        print("Press Enter to Exit...\n")
        readin = sys.stdin.read(1)
        exit(1)
```

Jupyter Notebook Viewer

https://nbviewer.jupyter.org/urls/cantera.org/examples/jupyter/thermo/flame_temperature.ipynb

jupyter nbviewer

JUPYTER FAQ

Flame Temperature

This example demonstrates calculation of the adiabatic flame temperature for a methane/air mixture, comparing calculations which assume either complete or incomplete combustion.

In [1]:
```
%matplotlib notebook
import cantera as ct
import numpy as np
import matplotlib.pyplot as plt
```

Complete Combustion

The stoichiometric equation for complete combustion of a lean methane/air mixture ($\phi < 1$) is:

$$\phi CH_4 + 2(O_2 + 3.76N_2) \rightarrow \phi CO_2 + 2\phi H_2O + 2(1-\phi)O_2 + 7.52N_2$$

For a rich mixture ($\phi > 1$), this becomes:

$$\phi CH_4 + 2(O_2 + 3.76N_2) \rightarrow CO_2 + 2H_2O + (\phi - 1)CH_4 + 7.52N_2$$

To find the flame temperature resulting from these reactions using Cantera, we create a gas object containing only the species in the above stoichiometric equations, and then use the `equilibrate()` function to find the resulting mixture composition and temperature, taking advantage of the fact that equilibrium will strongly favor conversion of the fuel molecule.

In [2]:
```
# Get all of the Species objects defined in the GRI 3.0 mechanism
species = {S.name: S for S in ct.Species.listFromFile('gri30.cti')}

# Create an IdealGas object with species representing complete combustion
complete_species = [species[S] for S in ('CH4','O2','N2','CO2','H2O')]
gas1 = ct.Solution(thermo='IdealGas', species=complete_species)

phi = np.linspace(0.5, 2.0, 100)
T_complete = np.zeros(phi.shape)
for i in range(len(phi)):
    gas1.TP = 300, ct.one_atm
    gas1.set_equivalence_ratio(phi[i], 'CH4', 'O2:1, N2:3.76')
    gas1.equilibrate('HP')
    T_complete[i] = gas1.T
```

Jupyter Notebook
.ipynb

Paquetes en Python

- Un paquete se instala solo una vez, a menos que se quiera actualizar la versión de un paquete ya adquirido, no obstante, se cargan en cada archivo a emplear.

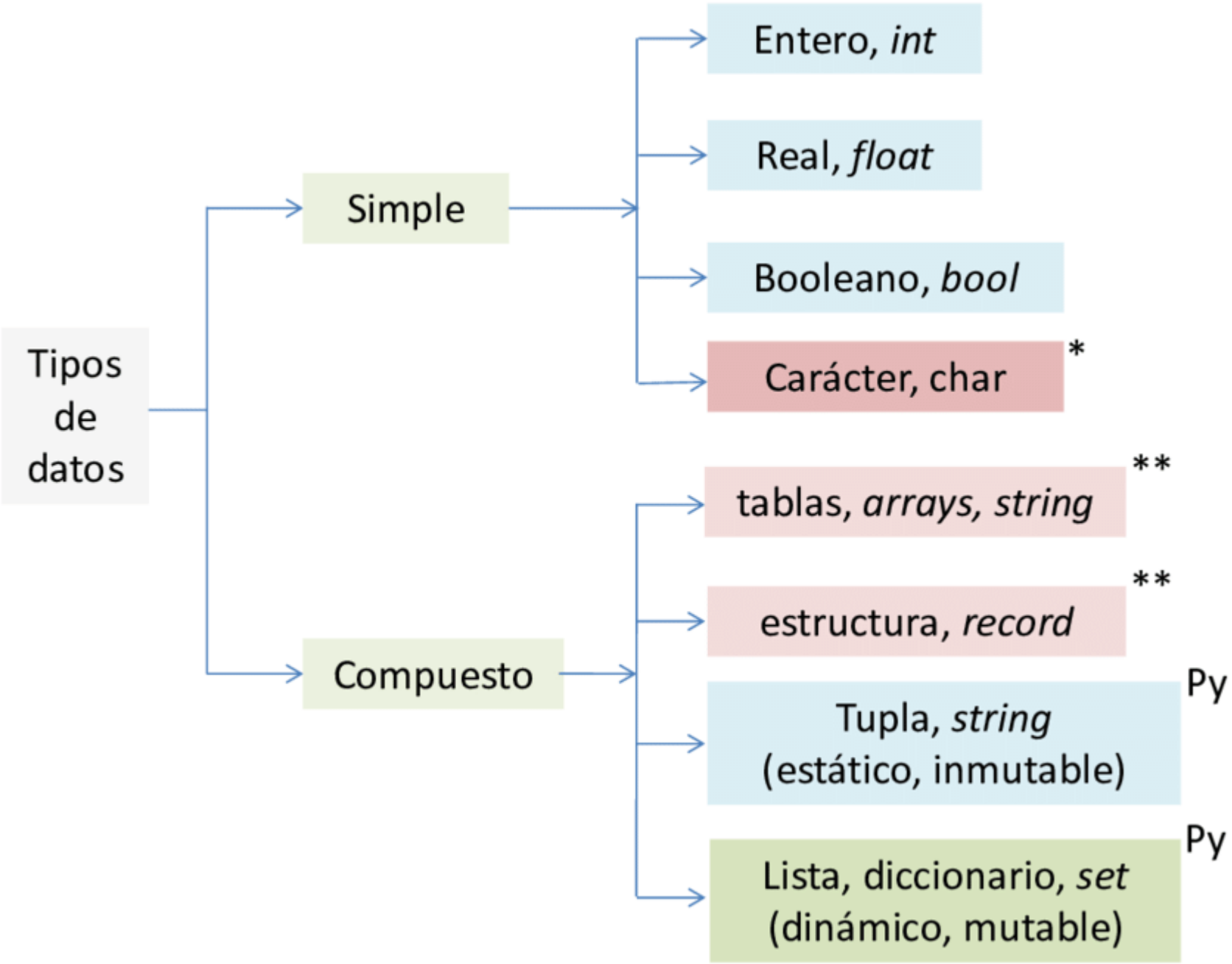
```
from numpy import *

# Uso de todas las funciones y clases del módulo numpy
arr = linspace(0, 10, 5)
```

```
from numpy import linspace as np_linspace

# Uso de la función importada
arr = np_linspace(0, 10, 5)
```

```
import numpy as np

# Uso de linspace
arr = np.linspace(0, 10, 5)
print(arr)
```

```
from numpy import linspace

# Uso de linspace
arr = linspace(0, 10, 5)
print(arr)
```

```
import numpy

# Uso de linspace
arr = numpy.linspace(0, 10, 5)
print(arr)
```



*El tipo de dato carácter no existe en Python, un carácter simple se representa como cadena de caracteres (string).

** Estructuras compuestas de lenguajes como C, FORTRAN, Pascal, Matlab, etc. Py: Estructuras compuestas en Python

TAREA N°1: Subir al Classroom

- Subir un archivo de datos que le gustaría trabajar durante las clases (.csv , .nc , entre otros).
- Escribir un archivo de texto en el que se describa brevemente el contenido de sus datos y qué le gustaría hacer con ellos en python.

“Un buen estilo de codificación es como la puntuación correcta: puede arreglárselas sin ella, pero seguroquehacealgomásfácildeleer.”
– La guía de estilo tidyverse

Ilustración adaptada de: Allison Horst