

Módulo 1. Introducción a la ciencia de datos

Herramientas

Javier Cózar

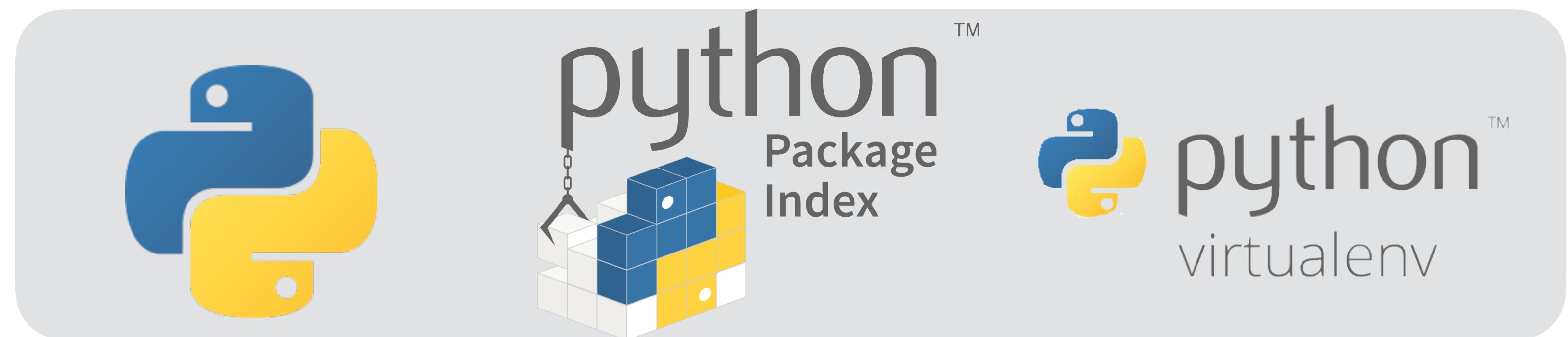
Máster en Ciencia de Datos e Ingeniería de Datos en la Nube





Contenidos

- Introducción
- Python, pip y entornos virtuales



- Jupyter notebooks

IP[y]:
IPython

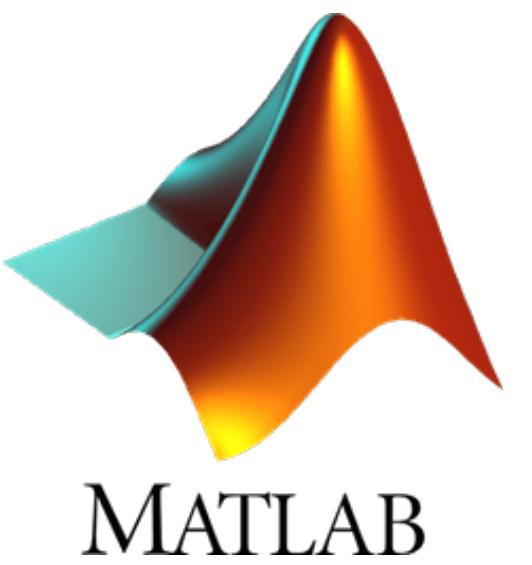


INTRODUCCIÓN



¿Por qué Python?

- El uso de lenguajes de programación fuera del ámbito informático se ha vuelto imprescindible



- ¿Qué tienen en común estos lenguajes?

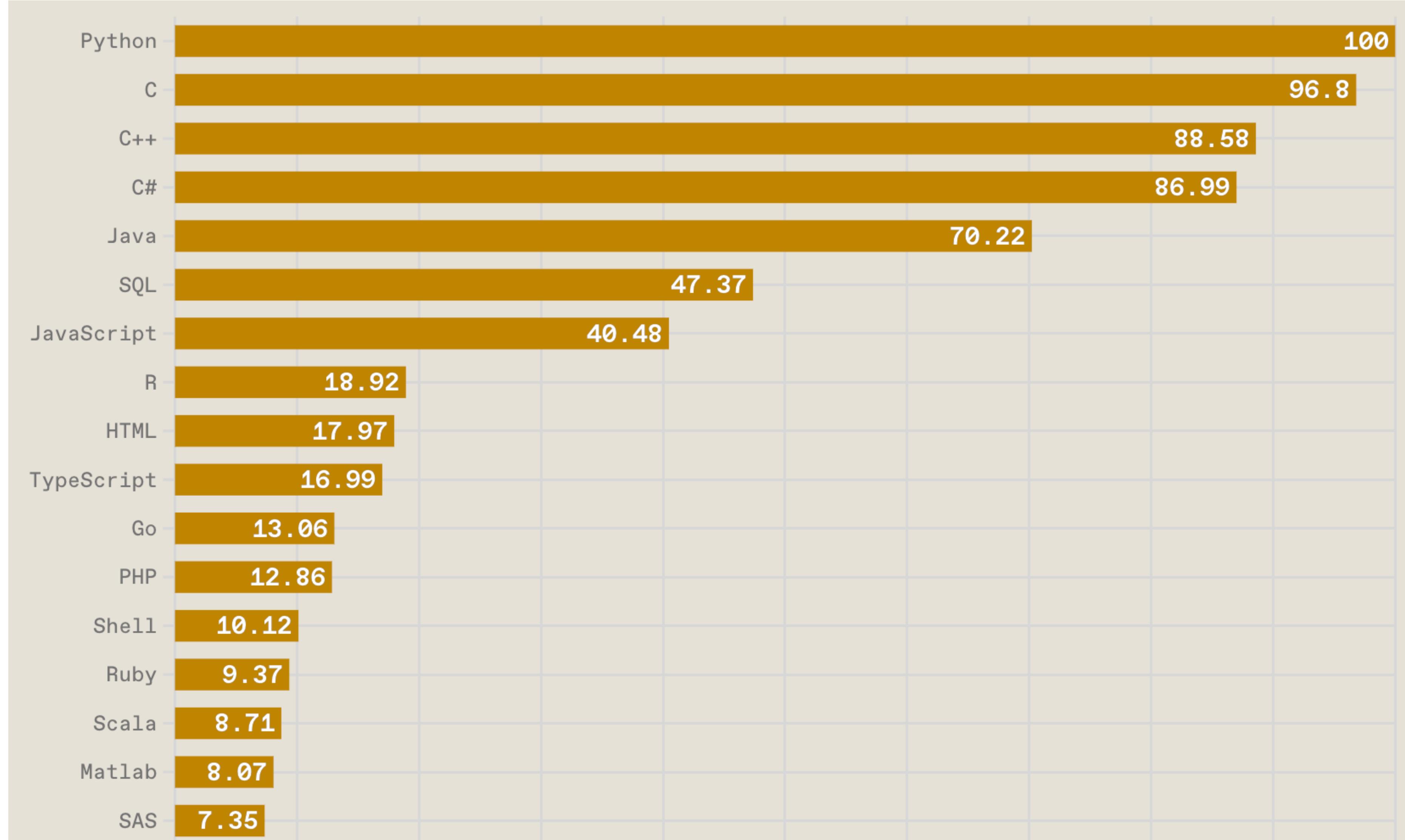
Interactivo

Gran comunidad

Aprendizaje

- Python, además es un potente lenguaje de programación de propósito general

¿Por qué Python?



Datos del [IEEE Spectrum \(2022\)](#)



Programa





Tratamiento de datos

- Tidy data: “Wickham, H. (2014). Tidy data. Journal of Statistical Software, 59(10), 1-23.”



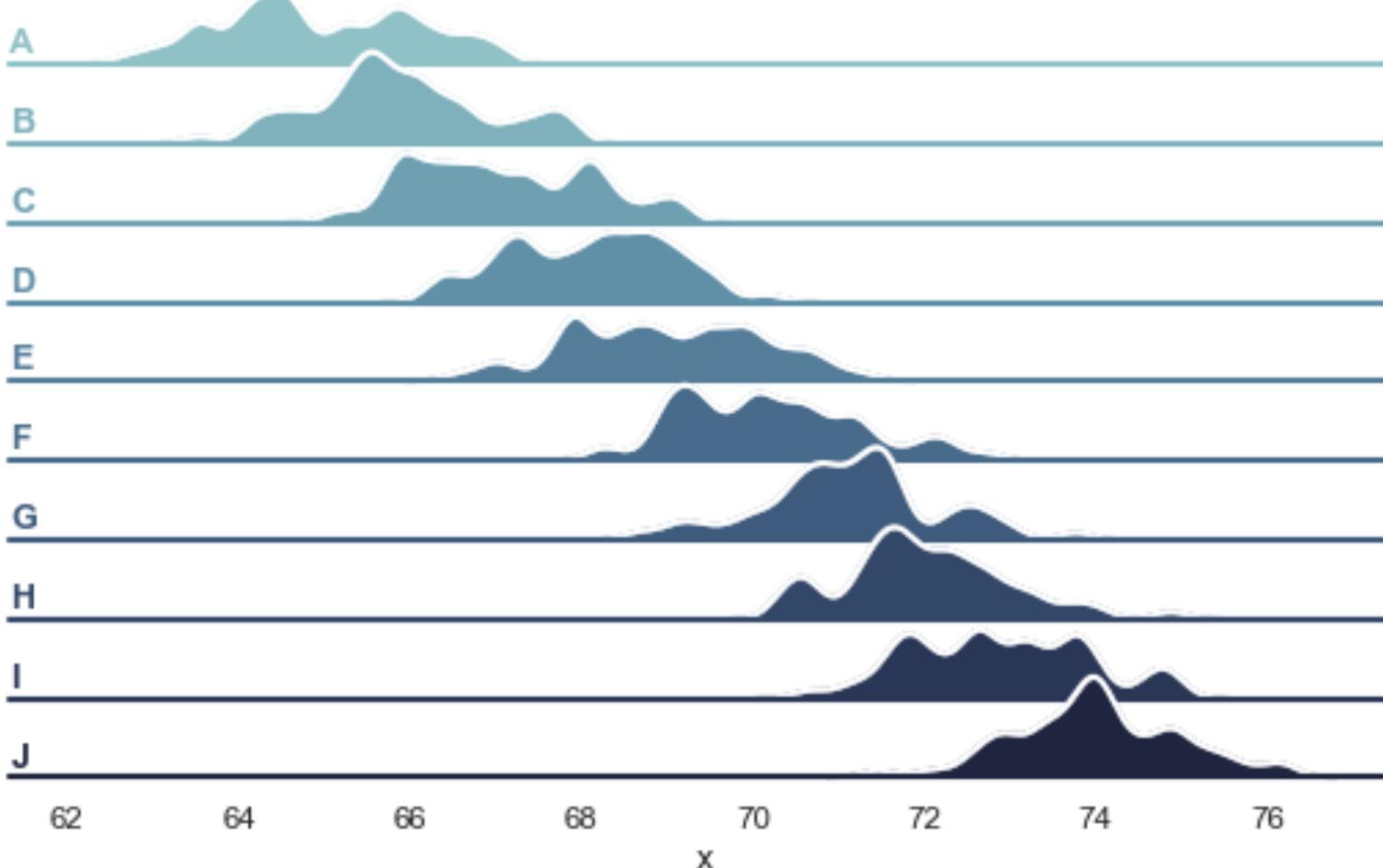
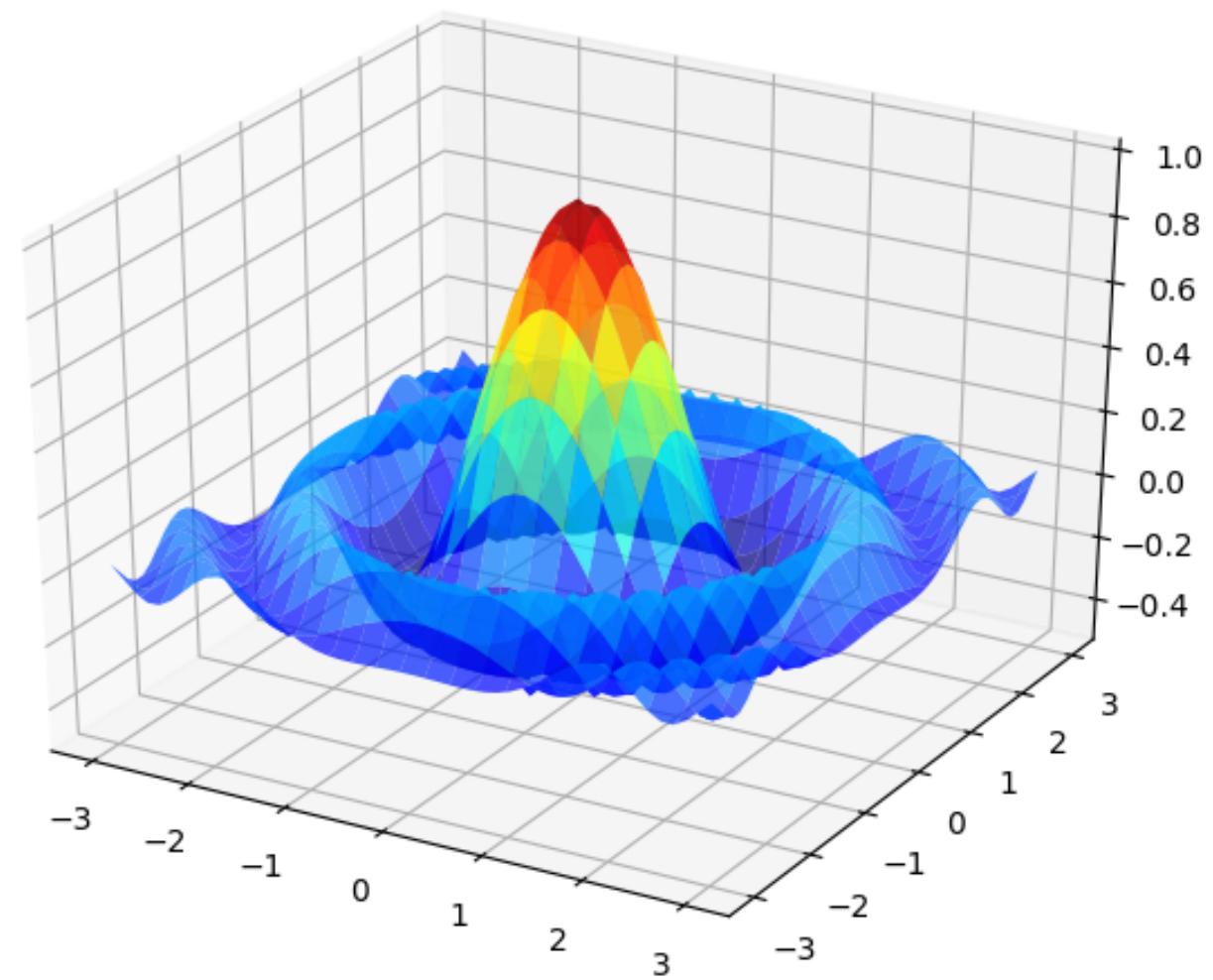
- Pandas: Representación y tratamiento de datos siguiendo el formato DataFrame de R





Visualización

- Disponemos de librerías muy potentes, con un alto poder de customización como **matplotlib**
- Librerías de más alto nivel nos permiten construir potentes gráficos de forma sencilla como **seaborn**, **altair**, **plotly** o **dash**





Machine Learning y Big Data

- *Machine Learning* aplicado a problemas tradicionales



- *Machine Learning* aplicado a problemas de *Big Data*





Deep Learning

- Los frameworks más utilizados (**Tensorflow, Keras, Pytorch**) usan Python
- Aunque están implementados en C, la comunidad desarrolla en Python





Entornos Cloud

- Los tres principales proveedores cloud ofrecen APIs y SDKs en Python
- Junto con Javascript, es el lenguaje que más se utiliza (y se actualiza!)



Google Cloud

PYTHON



Python

- Es el autor del lenguaje de programación Python

Hace seis años, en diciembre de 1989, estaba buscando un proyecto de programación como hobby que me mantuviera ocupado durante las semanas de Navidad. [...] Elegí el nombre de Python para el proyecto (siendo un gran fan de Monty Python's Flying Circus)



Guido Van Rossum



Python

- **Scripting:** no necesita compilación. Permite ejecutar código directamente en un intérprete
- **POO:** programación orientada a objetos intuitiva y muy potente
- **Orden superior:** Elementos de la programación funcional y funciones de orden superior muy útiles para gestionar datos

Documentación developers: <https://docs.python.org/3/library/>





Python - ¿Qué es un intérprete?

- Es un **entorno de trabajo** que contiene la información de lo que se está ejecutando (variables, funciones, paquetes importados, etc).
- En Python todo código se ejecuta dentro de un **intérprete**
- Estos entornos de trabajo son **aislados**: *los elementos en un entorno no son visibles en otros*

```
Python 3.7.7 (v3.7.7:d7c567b08f, Mar 10 2020, 02:56:16)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hola mundo!")
Hola mundo!
>>>
```

PYTHON

- Instalación de Anaconda
- Intérprete de Python 3

GESTIÓN DE PAQUETES



Python - Gestor de paquetes

- Python incluye una serie de paquetes de por defecto “core”
- Se puede **extender** la funcionalidad instalando nuevos paquetes en el entorno de trabajo (numpy, pandas, requests, ...)
- Como hemos dicho, Python es un lenguaje interpretado
 - Se puede descargar el código fuente de un paquete e importarlo (utilizarlo) en nuestro código

```
Python 3.7.7 (v3.7.7:d7c567b08f, Mar 10 2020, 02:56:16)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> requests.get('http://ifconfig.me').text
```



Python - Gestor de paquetes

- Por razones de **comodidad y reproducibilidad** se utiliza un gestor de paquetes y dependencias
- **pip** es el gestor de dependencias nativo de Python
- Existe un repositorio de paquetes **pypi**, de donde pip instala los módulos
- Se utiliza ejecutando el paquete con python (opción -m)

```
python -m pip --version
```



Python - Gestor de paquetes

- Instalar un paquete

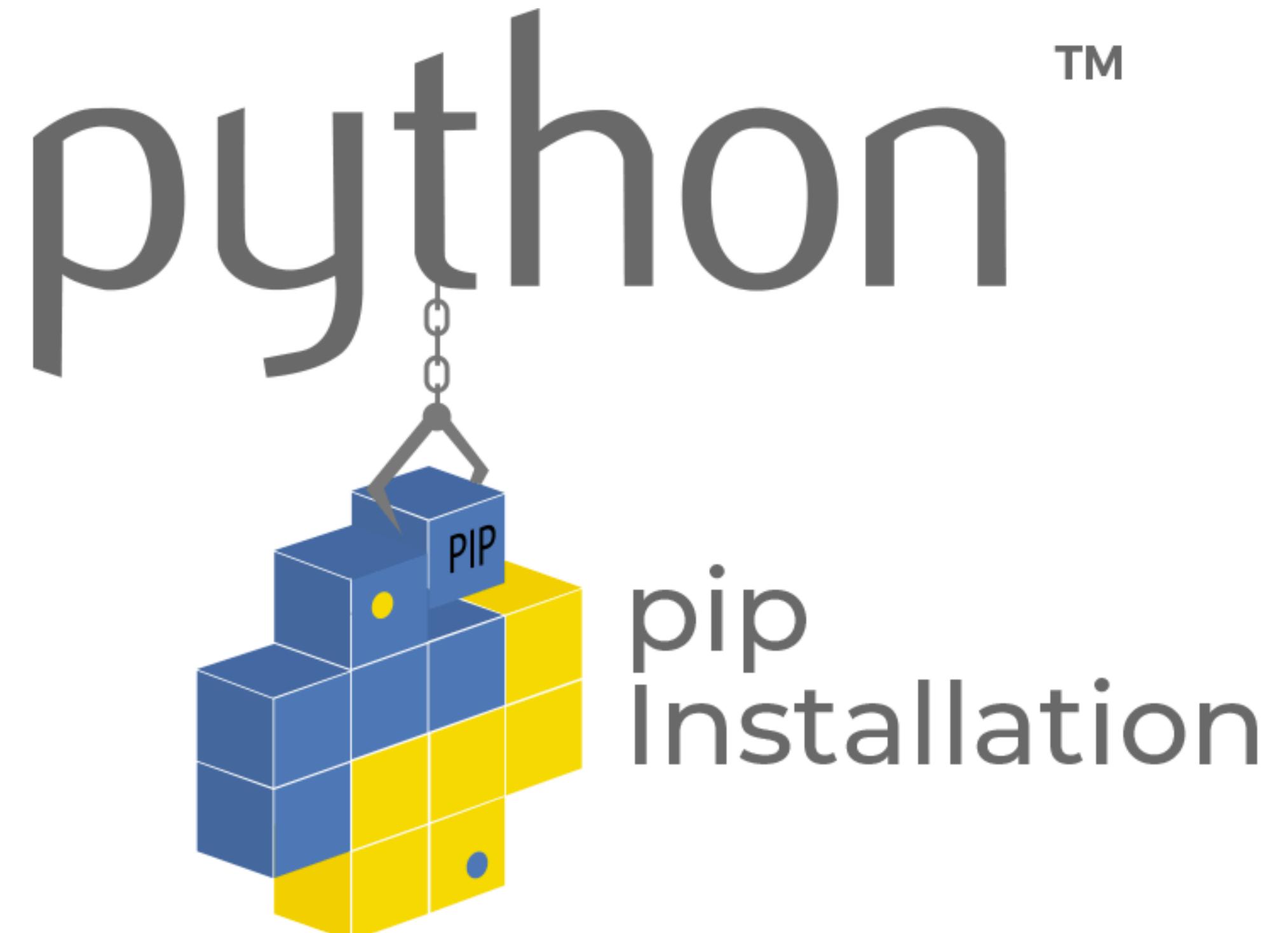
```
python -m pip install requests
```

- Actualizar un paquete

```
python -m pip install -U requests
```

- Desinstalar un paquete

```
python -m pip uninstall requests
```





Python - Gestor de paquetes

- Los paquetes se suelen versionar siguiendo un **versionado semántico (semver)**

<major>.<minor>.<patch>

```
python -m pip install requests==2.20.1
```

- Por reproducibilidad se suelen almacenar las dependencias que necesitemos en un fichero (una por línea) llamado requirements.txt

```
python -m pip install -r requirements.txt
```



Python - ¿Dónde se instalan estos paquetes?

- Existe una carpeta de sistema donde se descarga el código de los paquetes



Si instalamos una versión diferente de un paquete, sobreescribe la versión anterior

- Se desaconseja trabajar con el entorno de python del sistema

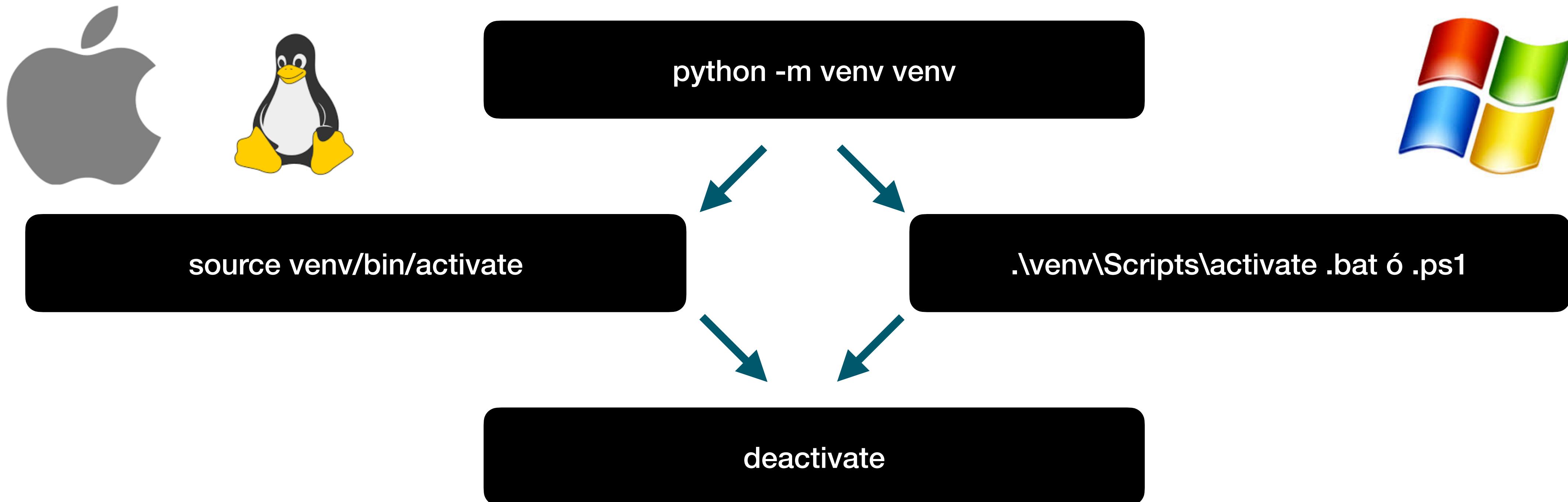


Los entornos virtuales permiten usar diferentes carpetas para almacenar las dependencias



Python - Entornos virtuales

- Entornos de trabajo **aislados**, y totalmente **reproducibles**
- Un entorno virtual se crea a partir de una **versión de python** (paquete **venv**)





Distribuciones alternativas de Python

- Aunque Python es interpretado, existen ciertas librerías que se traducen a código compilado, como **numpy**
- Hay **distribuciones** que se encargan de mantener un correcto versionado de los paquetes, precompilados con respecto a sistemas operativos, arquitecturas hardware, etc
- Uno de los más populares en el stack científico, y en particular orientado a data science y machine learning, es **Anaconda**
- **Intel** dispone de una distribución propia sobre Anaconda, optimizado para la ejecución en procesadores Intel



Anaconda

- Es una distribución *open source* de los lenguajes **Python** y **R**
- Incluye un gestor de paquetes así como de entornos virtuales llamado **conda**
- Anaconda-Navigator es una **interfaz visual** que facilita esta gestión
- El entorno **base** incluye una serie paquetes orientados al *stack* científico



ANACONDA NAVIGATOR

Interfaz visual de anaconda

