

# Configuración del ambiente de desarrollo: *brew, apt, conda & pip*

Capacitación de analítica avanzada



# Agenda

**1**

Introducción y  
objetivos de la sesión

**3**

CONDA: ambiente de  
desarrollo para ciencia  
de datos

**2**

Gestión de paquetes  
en sistemas *Unix*: APT  
y BREW

**4**

PIP: gestor de  
paquetes y librerías  
para Python

# Recordemos cómo se integra esta sesión en el programa

---

● Intervenciones teóricas

● Intervenciones prácticas en modo demo/tutorial

#	Modulo	Área	Tipo	Sesión
2.1	Fundamentos para el análisis de datos	Coding	●	Manejo de terminal con <i>bash, shell</i>
2.2	Fundamentos para el análisis de datos	Coding	●	<b>Ambiente de desarrollo:</b> <i>conda, pip, brew, apt get install</i>
2.3	Fundamentos para el análisis de datos	Coding	●	Principios de Python
2.4	Fundamentos para el análisis de datos	Coding	●	Principios de Python
2.5	Fundamentos para el análisis de datos	Data engineering	●	<b>Pandas:</b> <i>toolkit</i> para el procesamiento de datos

**Mini-proyecto**

# Introducción y objetivos de la sesión

---



## Introducción

Para empezar con el análisis avanzado de datos usando técnicas de ciencia de datos, debemos iniciar por configurar un ambiente de desarrollo propicio que nos de acceso a todas las herramientas computacionales que necesitamos. Esta sesión describe las herramientas más importantes para completar esta tarea



## Objetivos

- Conocer las herramientas básicas para gestionar el ambiente de desarrollo en tecnología usando sistemas unix
- Aprender sobre la estructura de la instalación de paquetes en unix
- Gestionar paquetes y librerías de Python con CONDA y PIP
- Aprender a instalar, actualizar y eliminar librerías de Python
- Crear y personalizar ambientes virtuales con CONDA

# ¿Por qué es importante la gestión de software en nuestro equipo?

Hay una nueva tendencia para el desarrollo

**El ecosistema  
ha  
evolucionado  
drásticamente**

**Antes**

Anteriormente, el desarrollo de modelos para el análisis de datos estaba encapsulado en plataformas de analítica con diversas funcionalidades (almacenamiento, modelación, implementación) en un mismo software. Algunas de esas plataformas eran, por ejemplo:



**Ahora**

Gracias a iniciativas tanto privadas como de código abierto (*open source*), se han desarrollado soluciones de forma descentralizada que cubren cada uno funciones específicas en la solución E2E. Esto ha creado infraestructuras heterogéneas, adaptativas y modulares. En ciencia de datos, algunas de estas tecnologías son:



# ¿Cuáles son las principales diferencias?

## Cambio de paradigma

---

Presentan un conjunto de técnicas de modelación más limitado para incluir nuevos desarrollos



Los últimos desarrollos teóricos son implementados desde el inicio con estas herramientas

Cubre un espectro menor del desarrollo E2E



A través de la integración de diversas soluciones crean una solución E2E

No adaptados para integrarse fácilmente con nuevas infraestructura, ejemplo: Big Data



Funcionan con sistemas modernos de procesamiento de información

Funcionamiento a base de licenciamiento



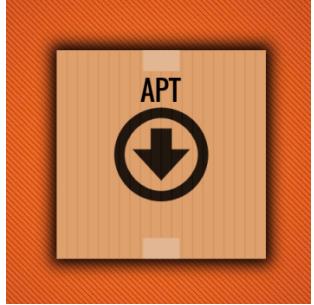
En su mayoría de libre acceso

## Luego

Para acceder a todas estas herramientas es importante aprender a gestionar paquetes y software utilizando herramientas especiales

# Gestión de paquetes y software en unix

---



## APT – advance packing tool

Apt es un programa de gestión de paquetes que simplifica la instalación, eliminación de programas en los sistemas linux, por ejemplo: Ubuntu



## Homebrew

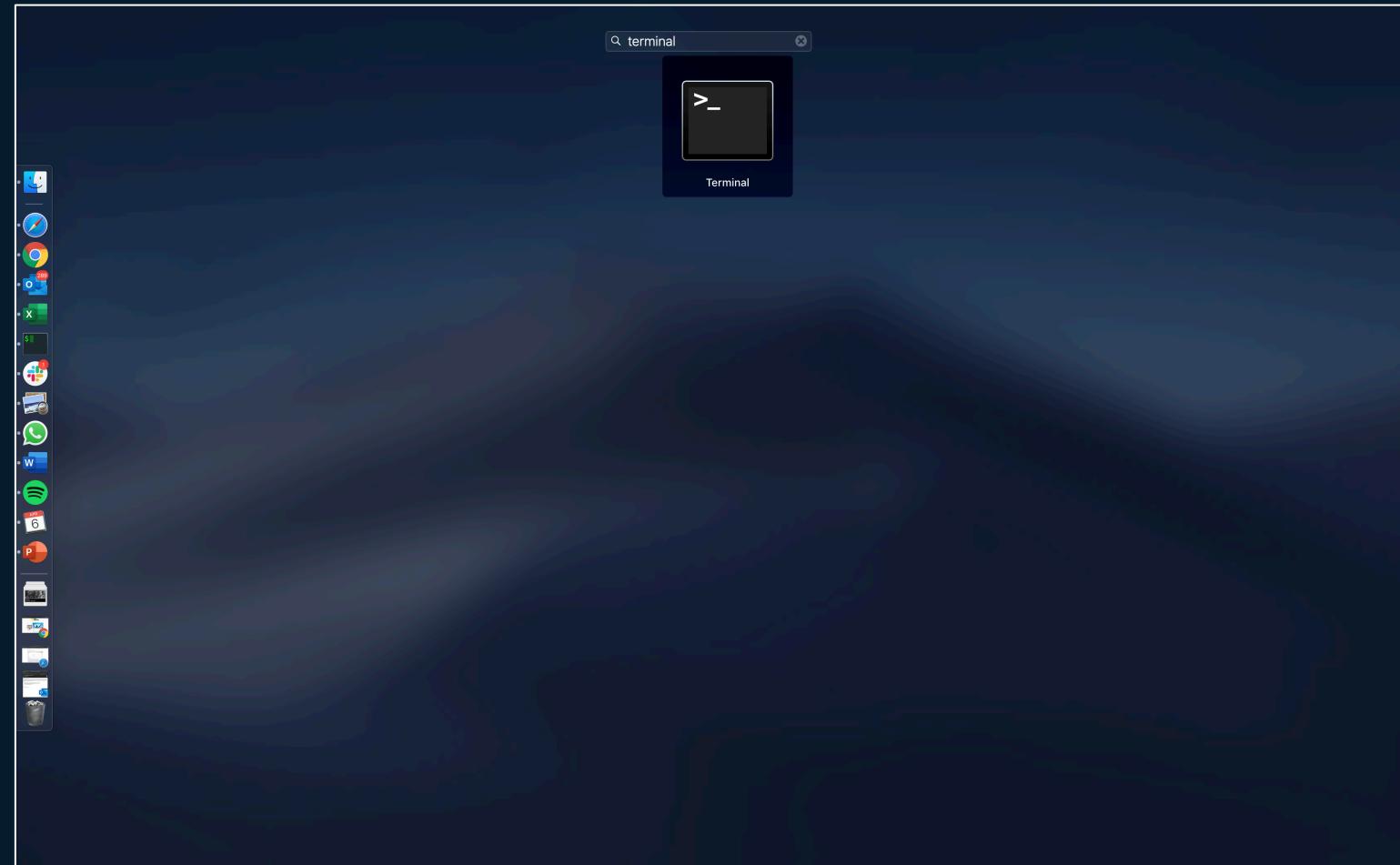
Homebrew es un sistema de gestión de paquetes que simplifica la instalación, eliminación de programas en sistemas operativos Mac OS

# ¿Cómo accedemos a estas herramientas?

Tomando como referencia el sistema Mac OS, buscamos en nuestras herramientas la aplicación *terminal*

Esta aplicación nativa es la puerta de entrada para comunicarnos con nuestro sistema operativo. Allí podemos:

- Manejar y explorar nuestros archivos
- Ejecutar programas
- Instalar nuevos programas a través de los gestores



# Conozcamos los gestores de paquetes (1/2)

```
torroledo@x86_64-apple-darwin13: ~
(base) ➔ ~ brew help
Example usage:
  brew search [TEXT|/REGEX/]
  brew info [FORMULA...]
  brew install FORMULA...
  brew update
  brew upgrade [FORMULA...]
  brew uninstall FORMULA...
  brew list [FORMULA...]

Troubleshooting:
  brew config
  brew doctor
  brew install --verbose --debug FORMULA

Contributing:
  brew create [URL [--no-fetch]]
  brew edit [FORMULA...]

Further help:
  brew commands
  brew help [COMMAND]
```

Usamos *brew help* para conocer la herramienta, saber las acciones disponibles

## Conozcamos los gestores de paquetes (2/2)

```
torroledo@x86_64-apple-darwin13: ~
(base) ➔ ~ brew commands
==> Built-in commands
--cache      desc      missing      tap-unpin
--cellar     diy       options      tap
--env        doctor    outdated    uninstall
--prefix     fetch     pin         unlink
--repository gist-logs postinstall unpack
--version    help      readall    unpin
analytics   home      reinstall  untap
cask         info      search     update-report
cat          install   sh         update-reset
cleanup     leaves    shellenv   update
command     link      style      upgrade
commands   list      switch    uses
config      log       tap-info   vendor-install
deps        migrate   tap-pin    test
                                         install-bundler-gems release-notes
                                         irb                  ruby
                                         linkage   tap-new
                                         man
```

Para una explicación detallada de los comandos y las formulas que se pueden crear usamos *brew commands*

# Instalemos algunos paquetes (1/2)

```
(base) ➔ ~ brew install wget
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 1 tap (homebrew/core).
==> New Formulae
beancount           emacs-dracula        pass-git-helper
==> Updated Formulae
cairo ✓             media-info          memcached
glib ✓              mercurial          mesa
graphite2 ✓         meson              micronaut
acpica              minetest          minikube
adios2              minio              minio-mc
anjuta              minetest          mk-configure
apache-arrow        minikube          mksh
arcade-learning-environment  minio          mkvtoolnix
astrometry-net      minio-mc          monolith
aubio              minio              mpd
autopep8            minio-mc          monolith
aws-cdk              minio              mpd
aws-sdk-cpp          minio              monolith
awscli              minio              mpd
baidupcs-go        minio              monolith
```

La gran utilidad de homebrew en Mac OS es que nos permite instalar de forma rápida herramientas nativas en unix, pero no en el default del sistema de Apple

## Instalemos algunos paquetes (2/2)

```
(base) ➔ ~ brew install tmux
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 1 tap (homebrew/core).
==> Updated Formulae
exploitdb

==> Installing dependencies for tmux: openssl@1.1, libevent, ncurses and utf8proc
==> Installing tmux dependency: openssl@1.1
==> Downloading https://homebrew.bintray.com/bottles/openssl@1.1-1.1.1f.mojave.bottle.tar.gz
==> Downloading from https://akamai.bintray.com/25/25ab844d2f14fc85c7f52958b4b89
#####
==> Pouring openssl@1.1-1.1.1f.mojave.bottle.tar.gz
```

Por ejemplo tmux es una aplicación útil cuando queramos ejecutar de forma ininterrumpida acciones en nuestra maquina en la nube

# Conda: ambiente de desarrollo en ciencia de datos

---



Anaconda es la plataforma más popular del mundo para el desarrollo de ciencia de datos y aprendizaje automático. Es un proyecto *open source* que distribuye Python y R en conjunto con otros complementos para la aplicación de ciencia de datos.

## Anaconda permite:

- Colectar datos de archivos bases de datos odatalakes
- Manejar ambientes virtuales con **conda**
- Compartir, colaborar y reproducir proyectos
- Desarrollar proyectos a nivel de producción con un simple botón
- Configurar de forma simple las herramientas principales de Desarrollo: python, jupyter, spyder, R, command line.

# CONDA

Conda es una poderosa herramienta para la gestión de paquetes usando CLI perteneciente a la plataforma Anaconda. Originalmente fue creada para la gestión de librerías de Python, sin embargo en la actualidad puede gestionar también paquetes para R, Ruby, Scala, Java, JavaScript, entre otros.

## Las principales ventajas de conda son:

- Centralizar el la instalación y gestión de librerías creadas de forma *open source* por la comunidad
- Permite la creación y gestión de ambientes virtuales de desarrollo
- Tomar la responsabilidad de gestionar el sistema en cada instalación

# Estructura de la instalación de Anaconda

---

## /anaconda3/envs/

Folder para los ambientes virtuales creados

## /anaconda3/Applications/

Folder para las aplicaciones como Rstudio

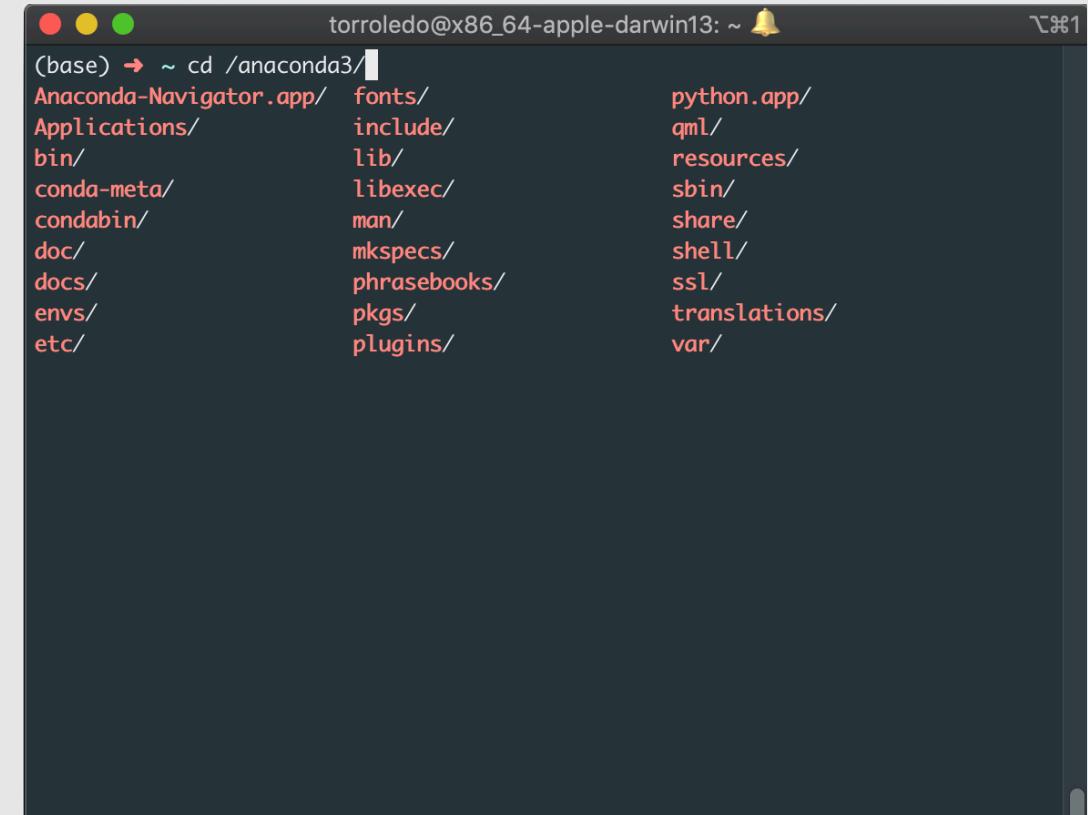
## /anaconda3/bin/

Folder para las librerías y paquetes instalados.

Aquí está python

## /anaconda3/include/

Folder para librerías C en nuestro sistema anaconda



```
(base) ~ cd /anaconda3/bin
.
├── Anaconda-Navigator.app/
├── Applications/
├── bin/
├── conda-meta/
├── condabin/
├── doc/
├── docs/
├── envs/
└── etc/
```

The terminal window shows the directory structure of the /anaconda3/bin folder. The structure includes subfolders like Anaconda-Navigator.app, Applications, bin, conda-meta, condabin, doc, docs, envs, and etc. To the right of the terminal window, there is a vertical scroll bar.

# Anaconda navigator

## Jupyter notebook y lab

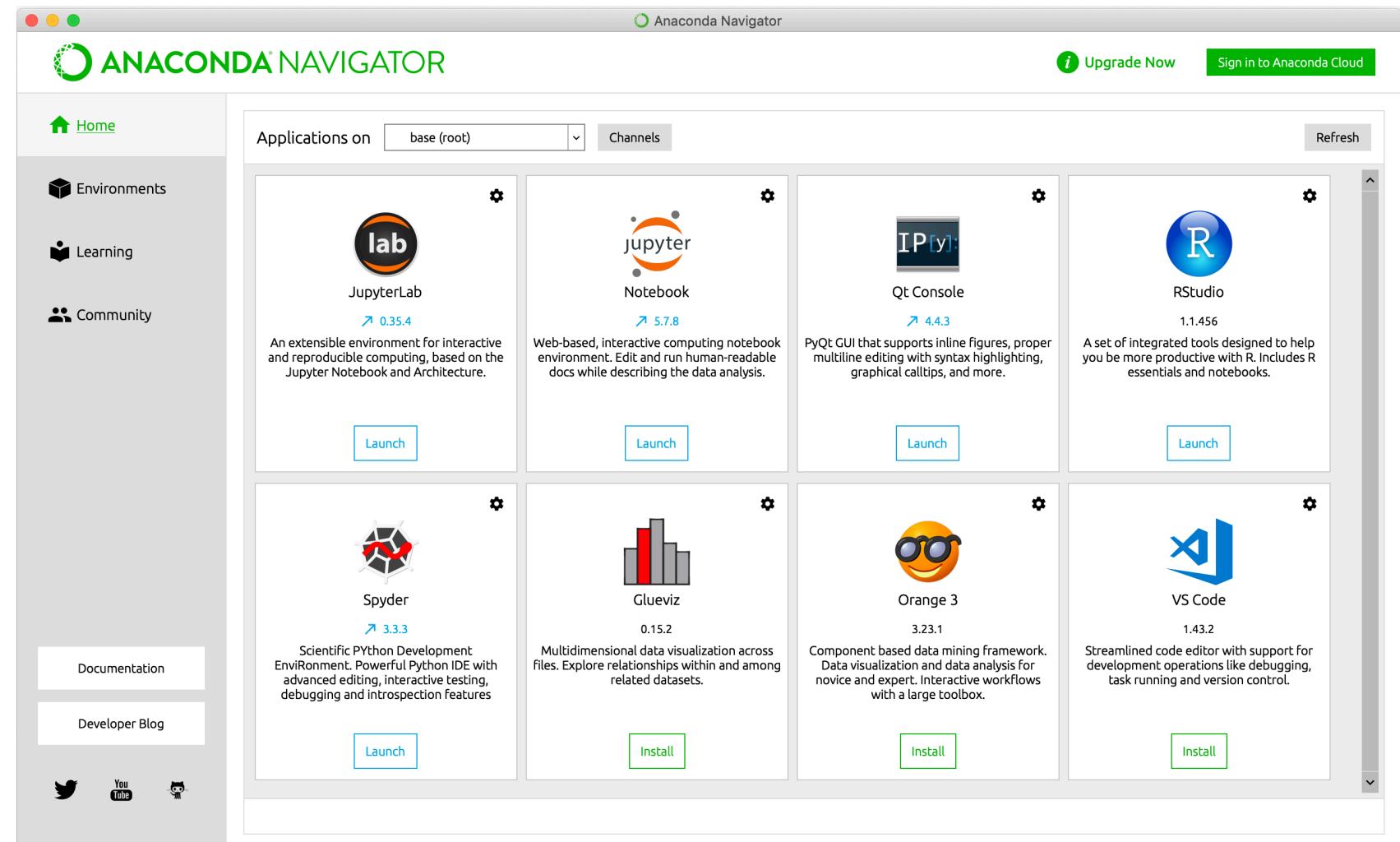
Ambiente de desarrollo  
interactivo para Python

## Rstudio

Ambiente de desarrollo  
interactivo para R

## Spyder

IDE para desarrollo en python



# Comandos básicos de Conda

## Conociendo conda

Como en otros paquetes, la opción --help nos ayudará a entender el funcionamiento de la herramienta y cuales acciones Podemos ejecutar con este software

```
torroledo@x86_64-apple-darwin13: ~
(base) ➔ ~ conda help
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and packages.

Options:

positional arguments:
  command
    clean      Remove unused packages and caches.
    config     Modify configuration values in .condarc. This is modeled
              after the git config command. Writes to the user .condarc
              file (/Users/torroledo/.condarc) by default.
    create     Create a new conda environment from a list of specified
              packages.
    help       Displays a list of available conda commands and their help
              strings.
    info       Display information about current conda install.
    init       Initialize conda for shell interaction. [Experimental]
    install   Installs a list of packages into a specified conda
              environment.
    list      List linked packages in a conda environment.
    package   Low-level conda package utility. (EXPERIMENTAL)
    remove   Remove a list of packages from a specified conda environment.
    uninstall Alias for conda remove.
```

# Creación de un ambiente virtual

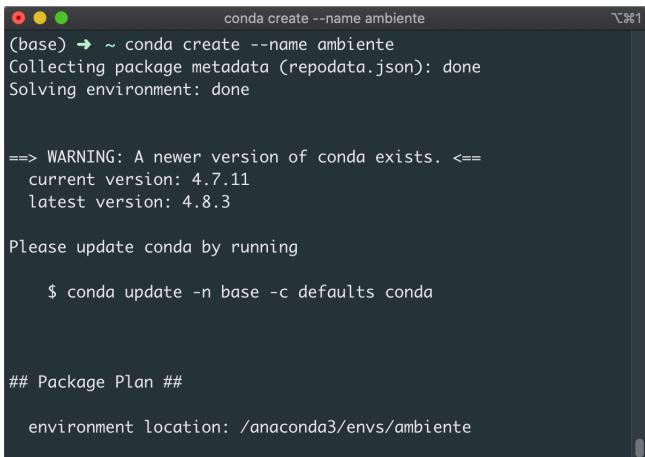
1

Ambiente  
base

A screenshot of a terminal window titled "torroledo@x86\_64-apple-darwin13: ~". The prompt "(base) ~" is visible, indicating the current environment.

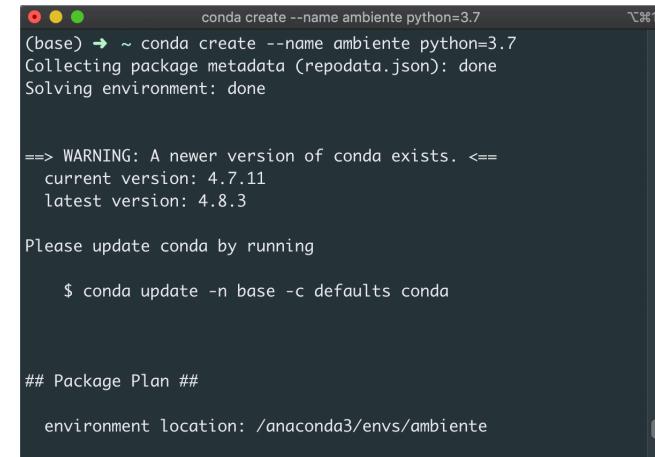
2

Creación de  
ambiente  
vacío

A screenshot of a terminal window titled "conda create --name ambiente". The command "conda create --name ambiente" is run, followed by a warning about an older conda version, a prompt to update, and the confirmation message "## Package Plan ##" and "environment location: /anaconda3/envs/ambiente".

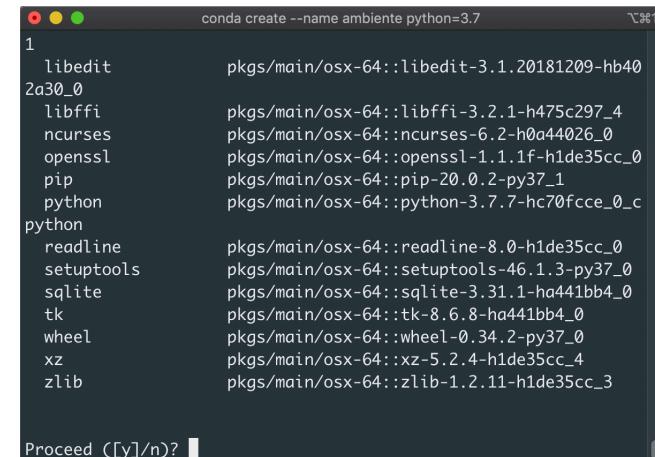
3

Creación de  
ambiente  
personalizado

A screenshot of a terminal window titled "conda create --name ambiente python=3.7". The command "conda create --name ambiente python=3.7" is run, followed by a warning about an older conda version, a prompt to update, and the confirmation message "## Package Plan ##" and "environment location: /anaconda3/envs/ambiente".

4

Autorización  
de la creación

A screenshot of a terminal window titled "conda create --name ambiente python=3.7". The command "conda create --name ambiente python=3.7" is run, followed by a list of packages being installed, and a final prompt "Proceed ([y]/n)?".

# Pip: sistema centralizado para gestión de paquetes en Python



pip es un sistema para la gestión de paquetes y librerías en *python* usando el repositorio centralizado de paquetes *Python Package Index (PyPI)*

## Comparación de conda y pip

	conda	pip
Maneja	Binarios	Wheel or source
¿Requiere compiladores?	no	Si
Tipos de paquetes	Cualquiera	Python
Ambientes virtuales	Inluido	No, require virtualenv o venv
Chequeo dependencias	Si	No
Fuente de paquetes	Repo y cloud de Anaconda	PyPI

# Creación de contenido y referencias

---

## Autores



**Iván Torroledo**

Data science fellow

Ivan\_Torroledo@mckinsey.com

Bogotá



**Simon Tamayo**

Data science specialist

Simon\_Tamayo@mckinsey.com

Bogotá

## Referencias

El contenido ha sido desarrollado gracias a la contribución de los autores y las siguientes fuentes externas:

- Documentación oficial Anaconda. Recuperado en Marzo 2020:  
<https://docs.anaconda.com/anaconda/navigator/tutorials/>
- Documentación oficial PyPI. Python Software Foundation.  
Recuperado en Marzo 2020:  
<https://packaging.python.org/tutorials/installing-packages/>