



CURSO DE INTELIGENCIA ARTIFICIAL

MÁLAGA, 30 mayo a 20 julio 2021

Inteligencia Artificial con Python

16 junio - 16:00-20:00

Oscar Ramirez Jimenez



Sobre el profesor

Oscar Ramirez Jimenez

Ingeniero técnico e ingeniero en Informática (UMA)

Pythonista desde 2013

Autor del libro: Python a fondo

Autor de la web: elpythonista.com



Conceptos IA con Python

Programación de algoritmos

Fundamentos de programación en Python

Herramientas para desarrollo de Python

Scraping con Python

Manipulación y presentación de datos (Pandas)

Librerías y herramientas IA con Python

Pandas

Jupyter-notebook

Requests

Dataframes

Matplotlib

Tipos de datos

build-in

PyPI

json, csv, sql

Anaconda

Puntos a cubrir hoy

Horas	Concepto
16:00 18:00	<ul style="list-style-type: none">• Programación de algoritmos, ejemplos, pseudo-código y diag. de flujo• Introducción a Python• Instalación de Python y arquitectura de ejecuciones• Ejemplos de código Python• Buenas prácticas en Python (PEP 8 y PEP 20)• Librerías y entornos virtuales
18:00 18:05	Descanso
18:05 20:00	<ul style="list-style-type: none">• IDEs y distribuciones• Instalación de Anaconda y Jupyter-notebook• Conda y gestión de dependencias• Tutorial de Jupyter notebook• Tutorial de markdown en Jupyter• Ejemplos de Python en Jupyter• Breve explicación de Google Colab

Descarga de anaconda

<https://www.anaconda.com/products/individual#Downloads>

Anaconda Installers

Windows

Python 3.8

64-Bit Graphical Installer (477 MB)

32-Bit Graphical Installer (409 MB)

MacOS

Python 3.8

☒ [64-Bit Graphical Installer \(440 MB\)](#)

64-Bit Command Line Installer (433 MB)

Linux

Python 3.8

64-Bit (x86) Installer (544 MB)

64-Bit (Power8 and Power9) Installer (285 MB)

64-Bit (AWS Graviton2 / ARM64) Installer (413 M)

64-bit (Linux on IBM Z & LinuxONE) Installer (292 M)

Programación de algoritmos

Los algoritmos definen ideas en forma de instrucciones con la finalidad de realizar una tarea

Ejemplo de algoritmo 1

¿Cómo preparar salmón al horno?



Ejemplo de algoritmo 1

1. Pelar las patatas
2. Cortar las patatas y verduras
3. Hornear durante 20 minutos (190°)
4. Añadir Salmón sazonado
5. Hornear por 10 minutos más
6. Emplatar



Ejemplo de algoritmo 2

¿Cómo poner una lavadora?



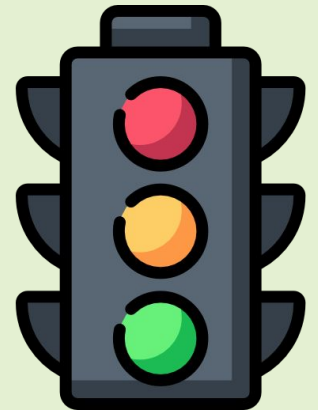
Ejemplo de algoritmo 2

1. Añadir ropa al tambor
2. Seleccionar tipo de lavado
3. Esperar a que termine el lavado
4. Tender



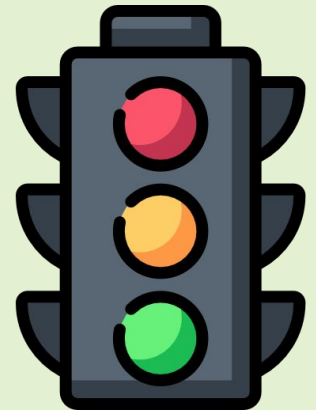
Ejemplo de algoritmo 3

¿Secuencia de un semáforo?



Ejemplo de algoritmo 3

1. Poner luz en color rojo
2. Esperar 30 segundos
3. Poner luz en verde
4. Esperar 40 segundos
5. Poner luz en ambar
6. Esperar 5 segundos
7. Volver al paso 1

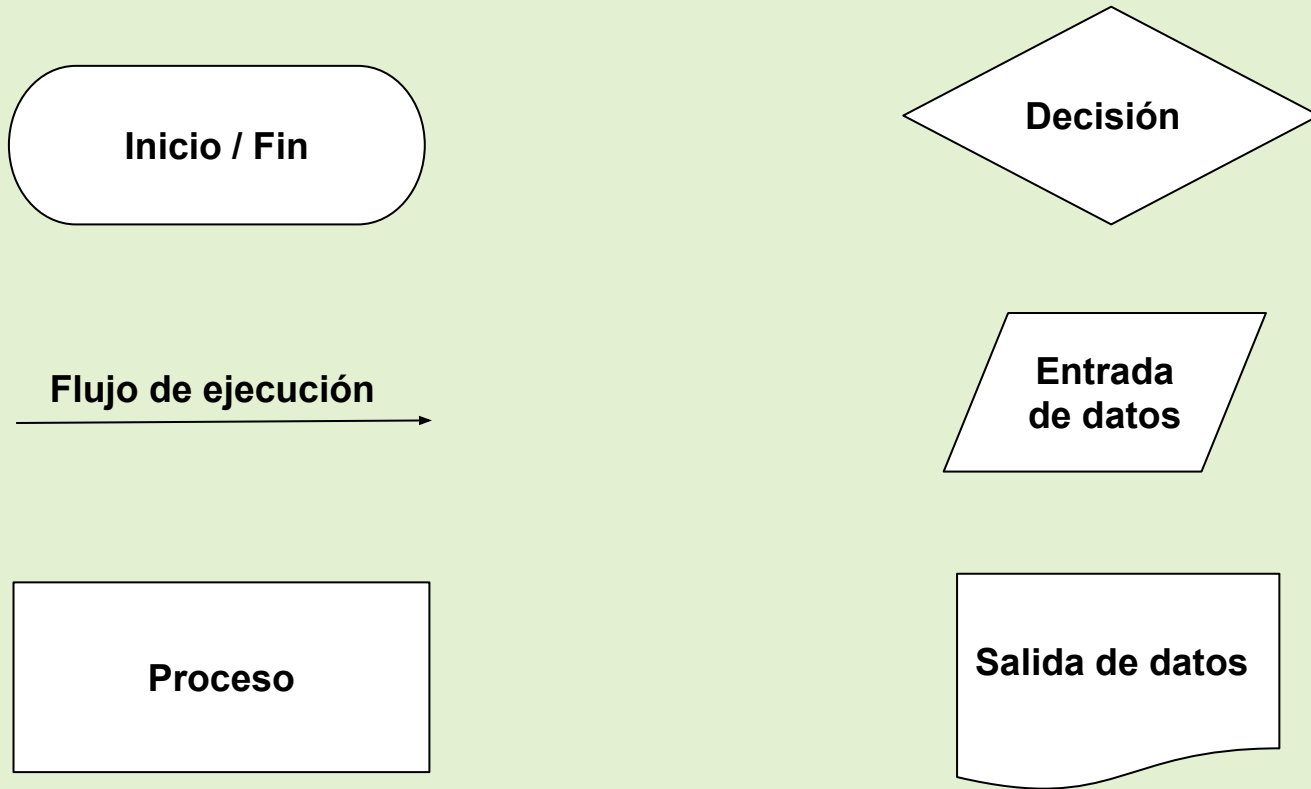


Qué es el pseudo código

Es una representación laxa de un algoritmo sin seguir patrones de ningún lenguaje de programación

“Sirve para hacer bocetos de algoritmos”

Diagrama de flujo



Pseudo código ej.1

1. Pelar las patatas
2. Cortar las patatas y verduras
3. Hornear durante 20 minutos (190°)
4. Añadir Salmón sazonado
5. Hornear por 10 minutos más
6. Emplatar

1- pelar(patatas)

2- cortar(patatas)

2- cortar(verduras)

3- hornear([patatas, verduras], 190, 20)

4- sazonar(salmon)

5- hornear(salmon, 190, 10)

6- emplatar([patatas, verduras, salmon])



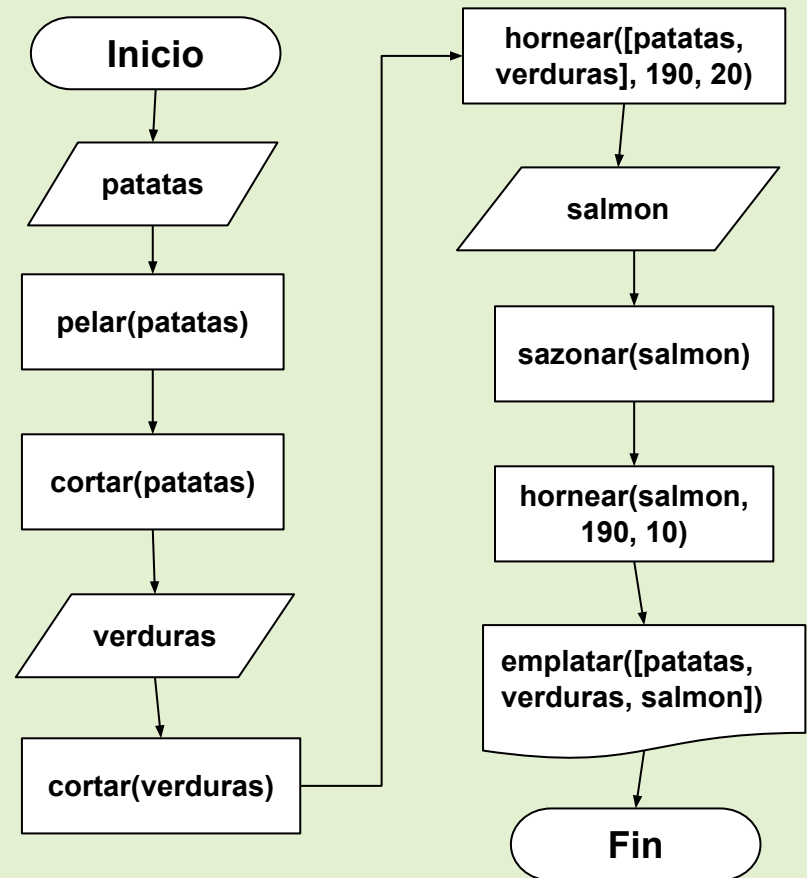
Diagrama de flujo ej.1

Pseudo código

- 1- pelar(patatas)
- 2- cortar(patatas)
- 2- cortar(verduras)
- 3- hornear([patatas, verduras], 190, 20)
- 4- sazonar(salmon)
- 5- hornear(salmon, 190, 10)
- 6- emplatar([patatas, verduras, salmon])



Diagrama de flujo



Pseudo código ej.2

1. Añadir ropa al tambor
2. Seleccionar tipo de lavado
3. Esperar a que termine el lavado
4. Tender

1 - añadir(ropa)

2 - seleccionarlavado(tipo)

3 - mientras no lavado_acabado

3 bis - esperar(10 segundos)

4 - tender



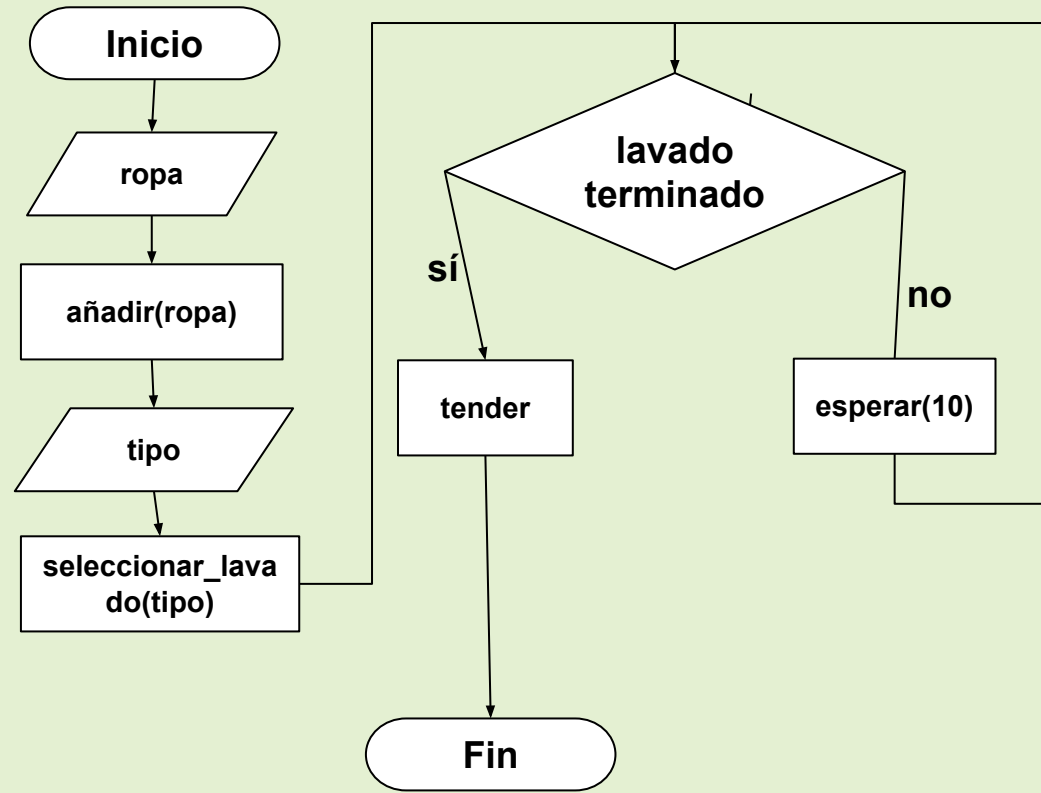
Diagrama de flujo ej.2

Pseudo código

- 1- añadir(ropa)
- 2 - seleccionarlavado(tipo)
- 3 - mientras no lavado_acabado
- 3 bis - esperar
- 4 - tender



Diagrama de flujo



Pseudo código ej.3

1. Poner luz en color rojo
2. Esperar 30 segundos
3. Poner luz en verde
4. Esperar 40 segundos
5. Poner luz en ambar
6. Esperar 5 segundos
7. Volver al paso 1

1 - cambiarcolor('rojo')

2 - esperar(30)

3 - cambiarcolor('verde')

4 - esperar(40)

5 - cambiarcolor('ambar')

6 - esperar(5)

7 - paso-1

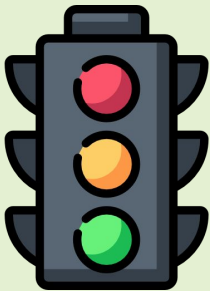


Diagrama de flujo ej.3

Pseudo código

1 - cambiarcolor('rojo')
2 - esperar(30)
3 - cambiarcolor('verde')
4 - esperar(40)
5 - cambiarcolor('ambar')
6 - esperar(5)
7 - paso-1

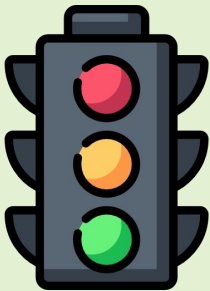
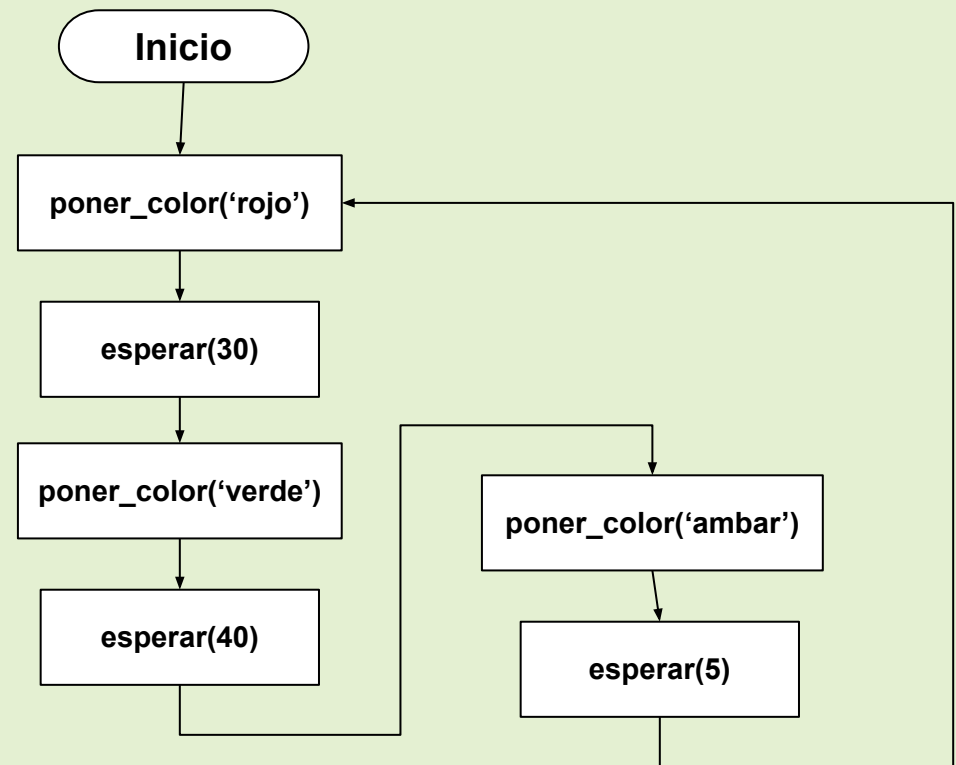


Diagrama de flujo



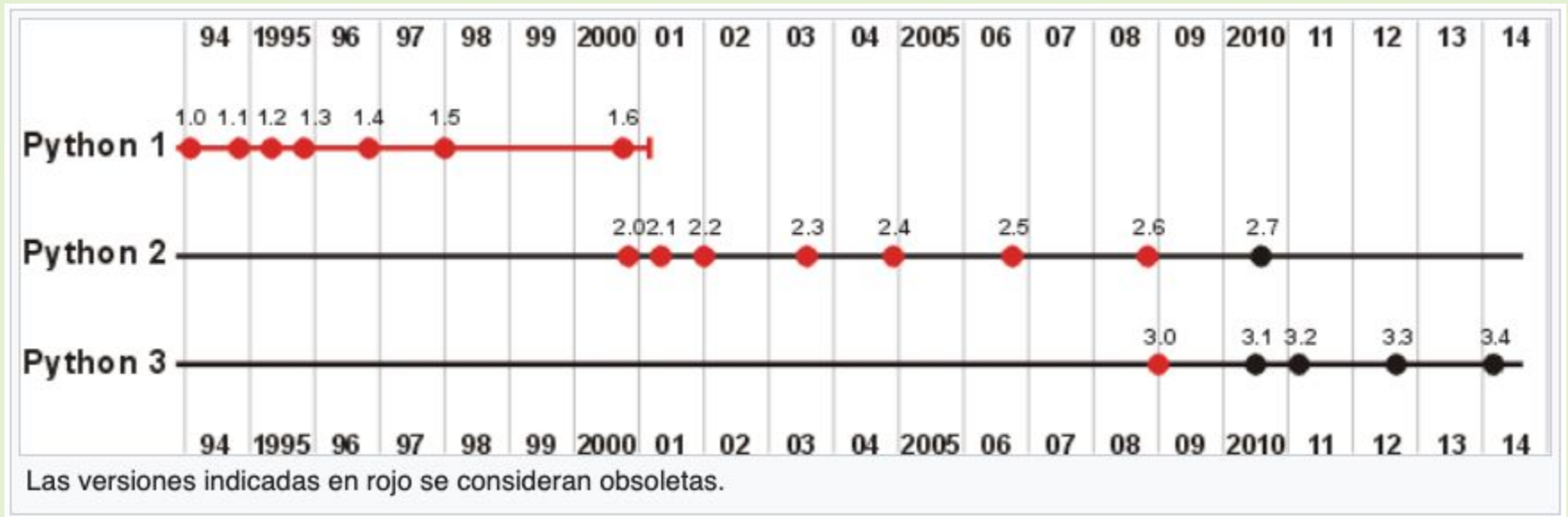
Qué es Python

Python es un **lenguaje de programación**

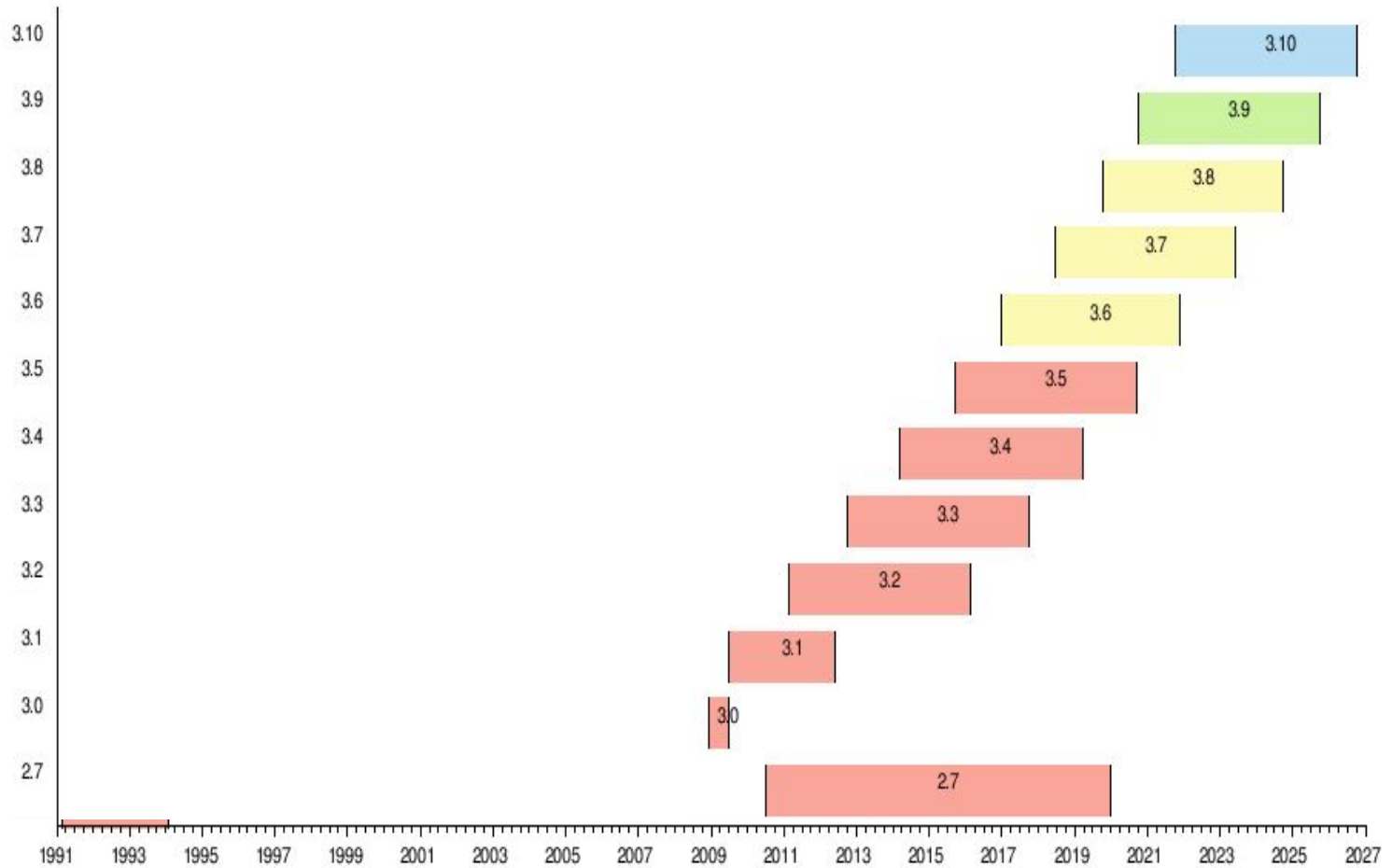
- Software libre
- Multiparadigma
- Multiplataforma
- Interpretado
- Elegante y simple, pero extremadamente potente
- De tipado dinámico y fuertemente tipado



Historia de Python



Historia de Python

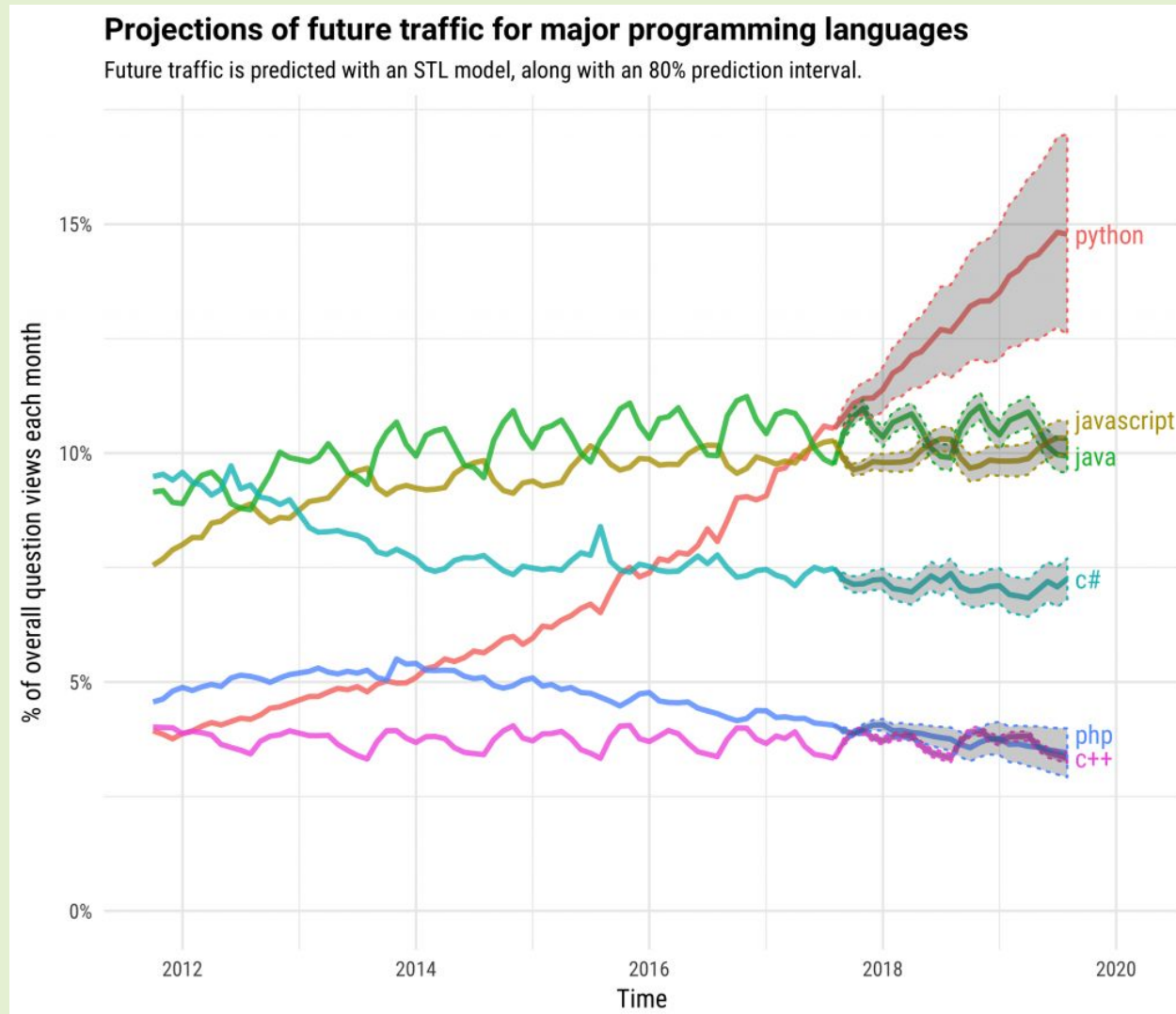


Usos de Python

Muy utilizado en diferentes áreas como:

- Desarrollo de apps de escritorio
- Desarrollo web
- Testeo y ciberseguridad
- Administración de sistemas y Scripting
- Ciencia de datos e inteligencia artificial

Predicciones de uso



<https://stackoverflow.blog/2017/09/06/incredible-growth-python/>

Componentes de Python

Máquina virtual de Python

Intérprete de Python

Ficheros de python

Librerías

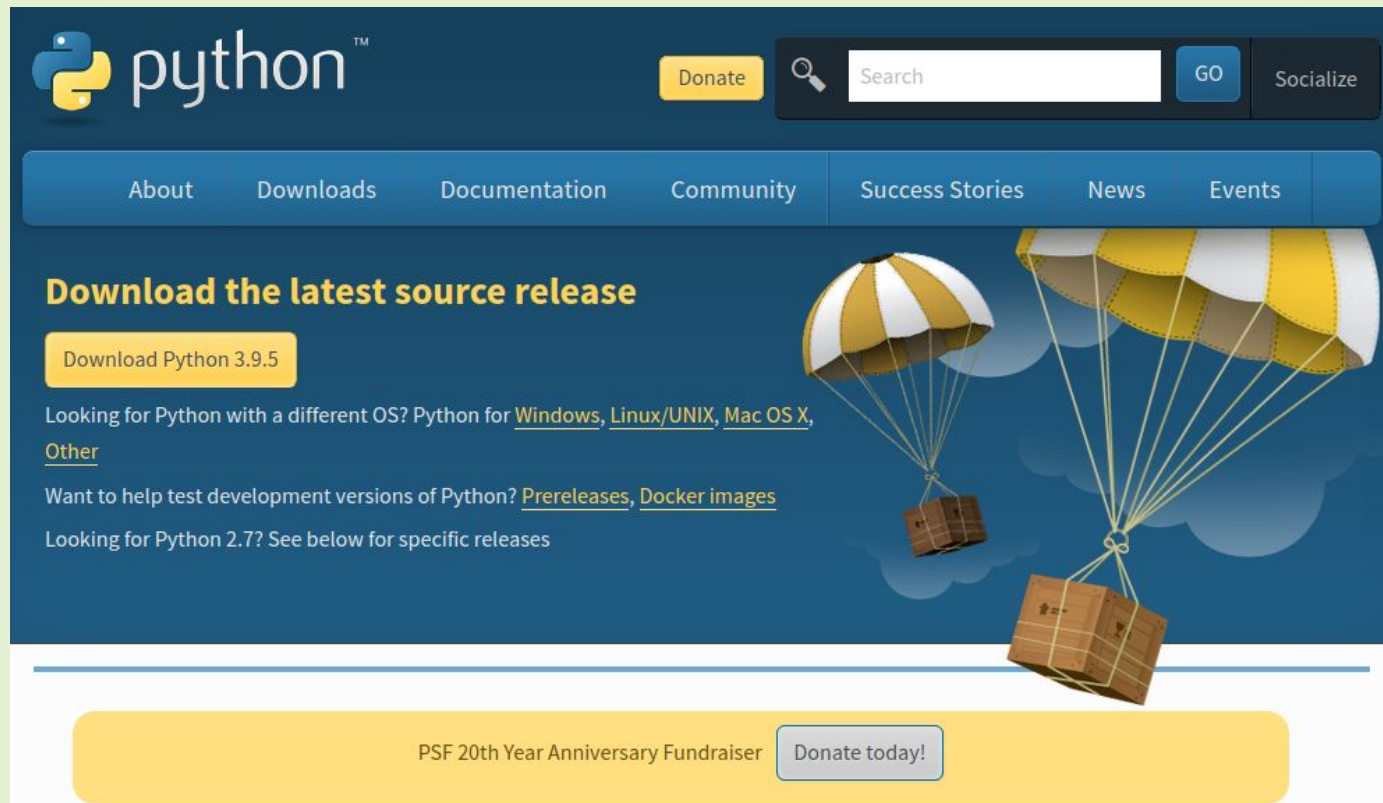
Tipos de ficheros Python

Diferentes extensiones:

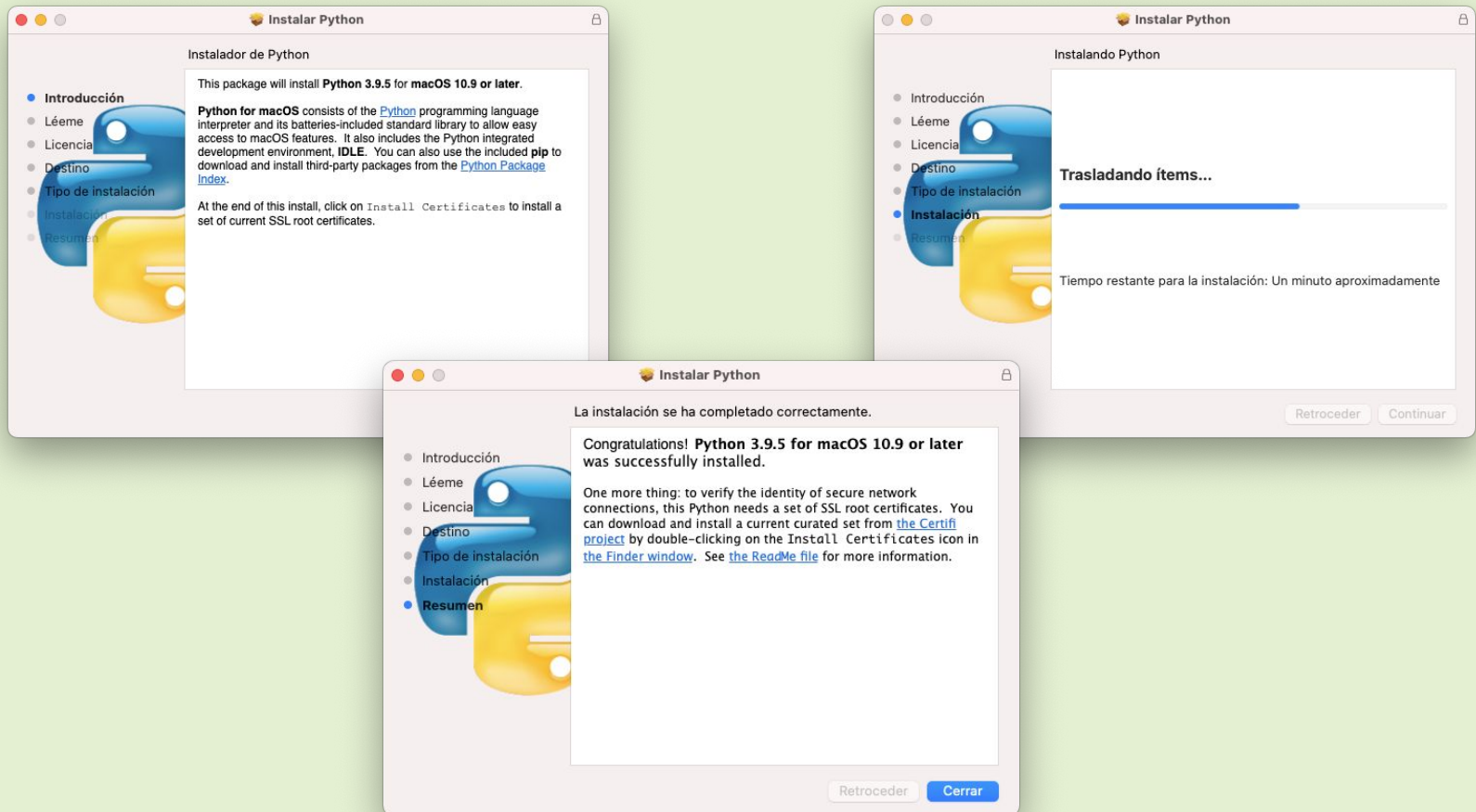
- * `.py`: ficheros de código fuente
- * `.pyc`: código fuente compilado
- * `.pyw`: ficheros para interfaz gráfica

Cómo instalar de Python

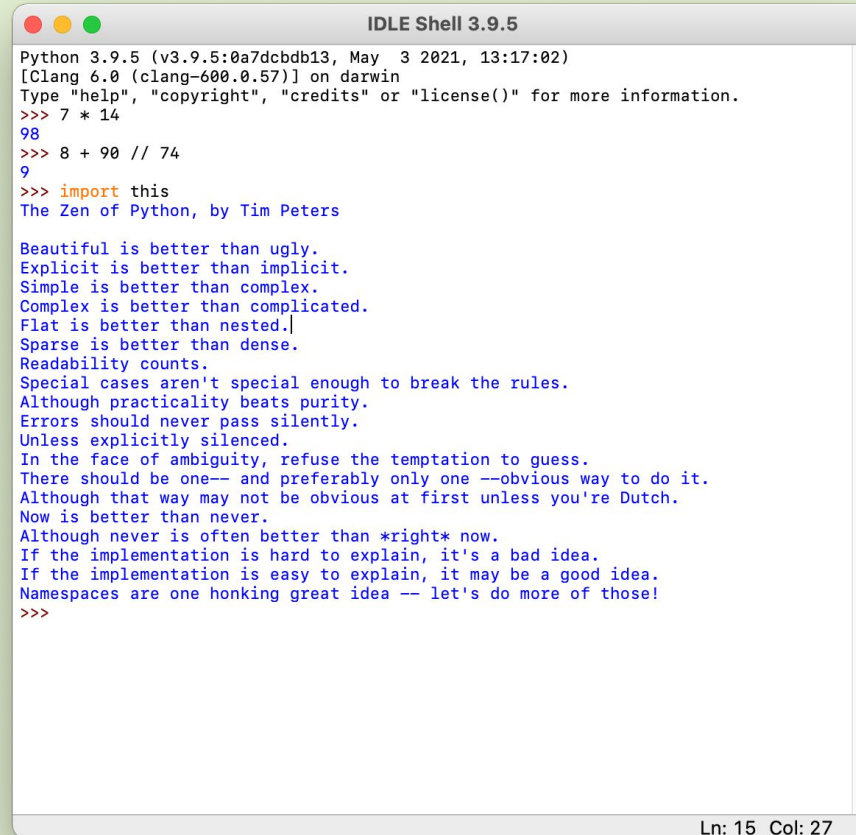
Página oficial: <https://www.python.org/downloads/>



Cómo instalar de Python



IDLE e intérprete



```
Python 3.9.5 (v3.9.5:0a7dcdbd13, May 3 2021, 13:17:02)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> 7 * 14
98
>>> 8 + 90 // 74
9
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Ln: 15 Col: 27

Ejecutar el primer programa Python

```
if __name__ == "__main__":  
    print('Hola mundo')
```

Guardar este código en un archivo llamado:

```
mi_primer_programa.py
```


Ejecutar el primer programa Python

Abrir la terminal de comandos para ejecutar python y
ejecutar el programa

```
> python mi_primer_programa.py
```

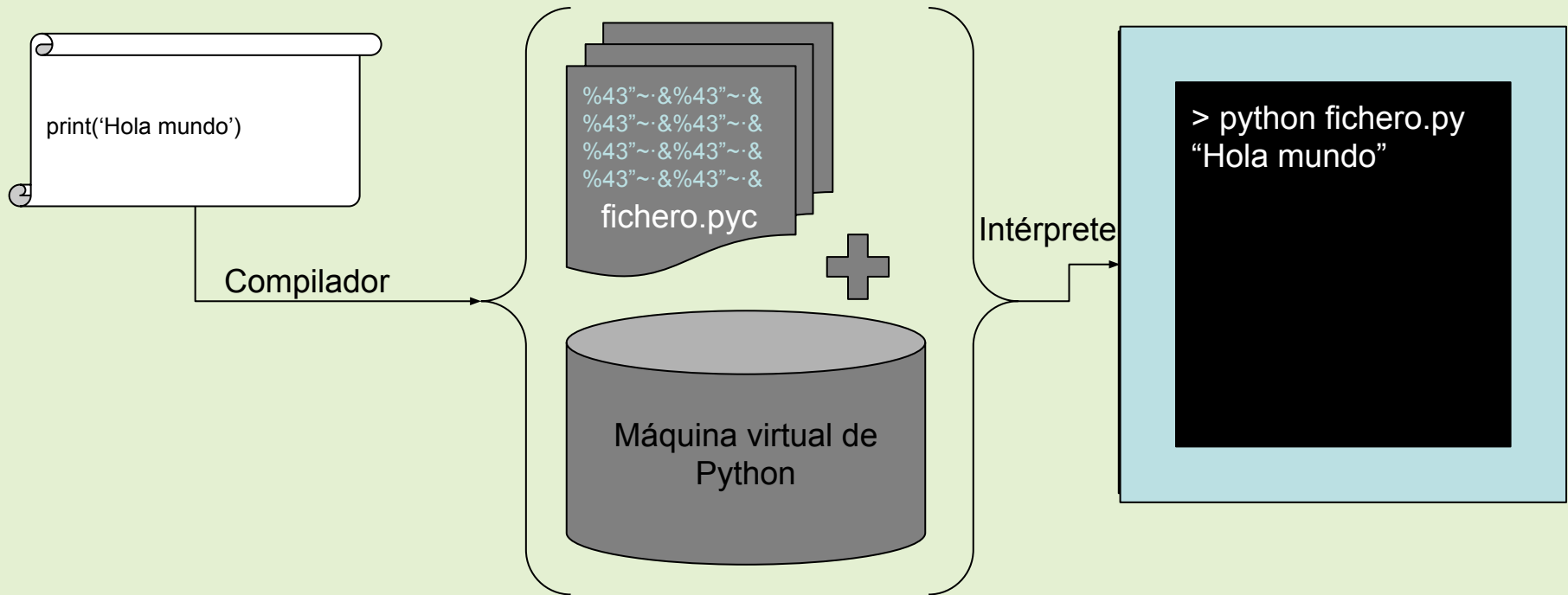
Ahora usando **IDLE**

Código fuente ejecutable

Cuando se lanza el intérprete sobre un fichero de python, si el fichero contiene una sentencia de ejecución, ejecutará el código asociado:

```
if __name__ == "__main__":  
    print("Hola mundo")
```

Ejecución de Python



Estructura de código fuente

Pueden contener:

- **Sentencias de control:** `if, for, while, else`
- **Funciones:** `def mi_funcion()`
- **Clases:** `class mi_clase()`
- **Variables y literales:** `a = 'juan'`
- **Importaciones:** `from json import dump`

Sintaxis de Python

- Indentación de bloques de ejecución
- Sentencias separadas por saltos de línea
- Sintaxis clara y concisa en inglés

Sintaxis de Python

```
>>> def es_par(num):  
...     return not num % 2  
...  
>>> es_par(5)  
False  
>>> es_par(4)  
True
```

Sintaxis de Python

```
>>> names = ['pepe lopez', 'juan perez', 'maría cámara']
>>> for name in names:
...     print(name.title())
...
Pepe Lopez
Juan Perez
María Cámara
```

Sintaxis de Python

```
>>> nums = [3, 4, 5, 61, 2, 5]
>>> for idx, num in enumerate(nums):
...     print(f'Posición [{idx}] valor {num}')
...
Posición [0] valor 3
Posición [1] valor 4
Posición [2] valor 5
Posición [3] valor 61
Posición [4] valor 2
Posición [5] valor 5
```


Sintaxis de Python

```
>>> edades = {'juan': 34, 'maria': 26, 'pedro': 44}
>>> for nombre, edad in edades.items():
...     print(f'Nombre: {nombre} - edad: {edad}')
...
Nombre: juan - edad: 34
Nombre: maria - edad: 26
Nombre: pedro - edad: 44
```

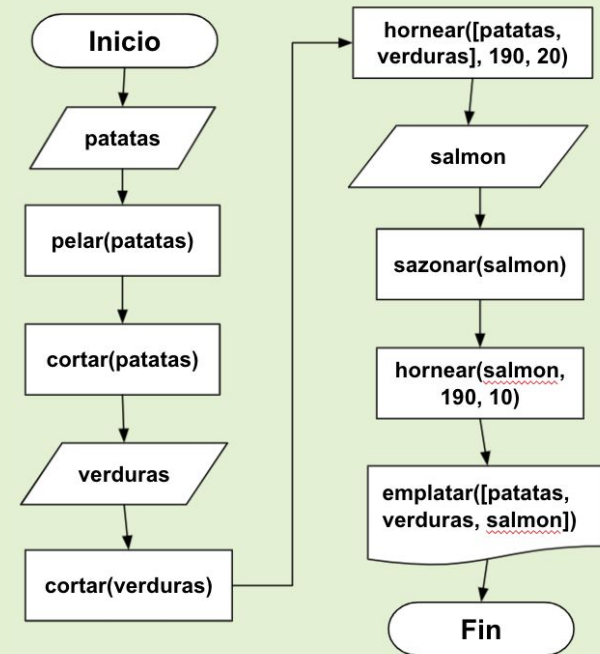
Código Python ej.1

Código Python

```
def preparar_salom(patatas, verduras, salmon):  
    patatas_peladas = pelar(patatas)  
    p_cortadas = cortar(patatas_peladas)  
    v_cortadas = cortar(verduras)  
    guarnicion = hornear([p_cortadas, v_cortadas], 190, 20)  
    s_sazonado = sazonar(salmon)  
    s_horneado = hornear([s_sazonado], 190, 10)  
    return emplatar([guarnicion, s_horneado])
```



Diagrama de flujo



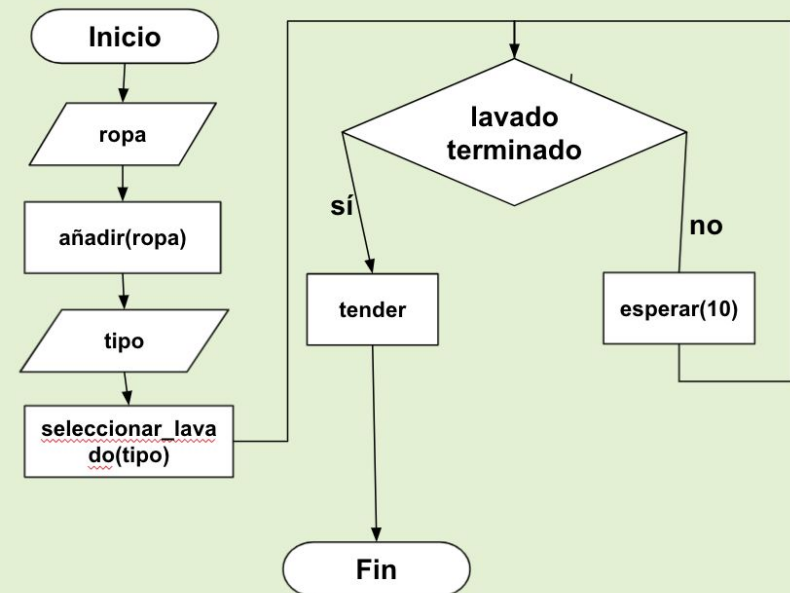
Código Python ej.2

Código Python

```
def lavar_ropa(lavadora, ropa, tipo):  
    lavadora.agregar(ropa)  
    lavadora.lavado_tipo(tipo)  
    estado = lavadora.estado  
    while estado != 'terminado':  
        sleep(10)  
    ropa = lavadora.ropa  
    tender(ropa)
```



Diagrama de flujo



Código Python ej.3

Código Python

```
def secuencia_sem(semaforo):  
    while True:  
        semaforo.color('rojo')  
        sleep(30)  
        semaforo.color('verde')  
        sleep(40)  
        semaforo.color('ambar')  
        sleep(5)
```

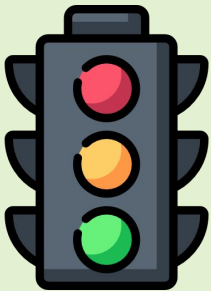
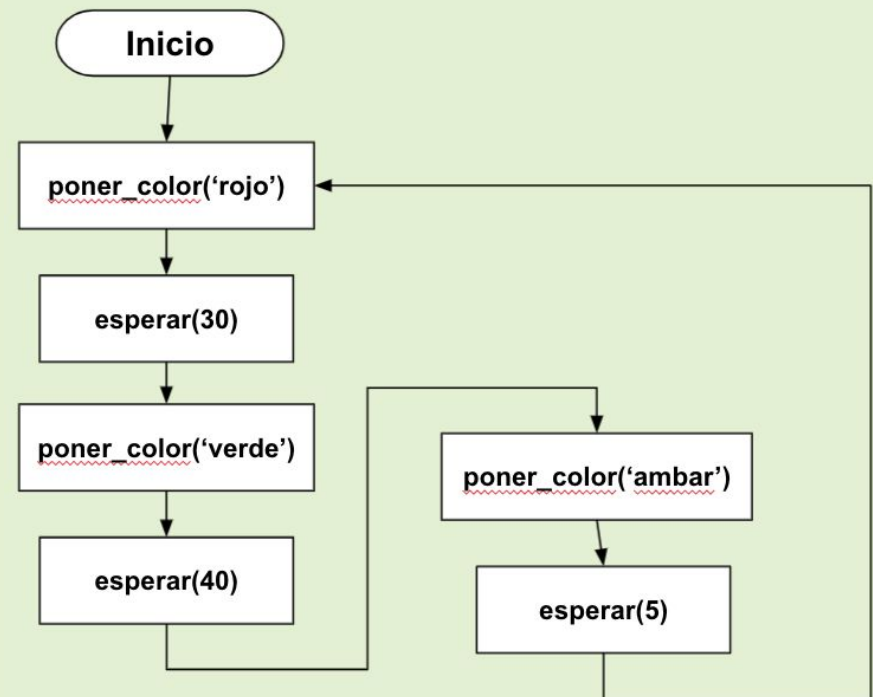


Diagrama de flujo



Estilos en Python (PEP 8)

Ver en: <https://elpythonista.com/pep-8>

Zen de Python (PEP 20)

Ver en: <https://elpythonista.com/zen-de-python>

Librerías en Python

Son código libre

Github + python

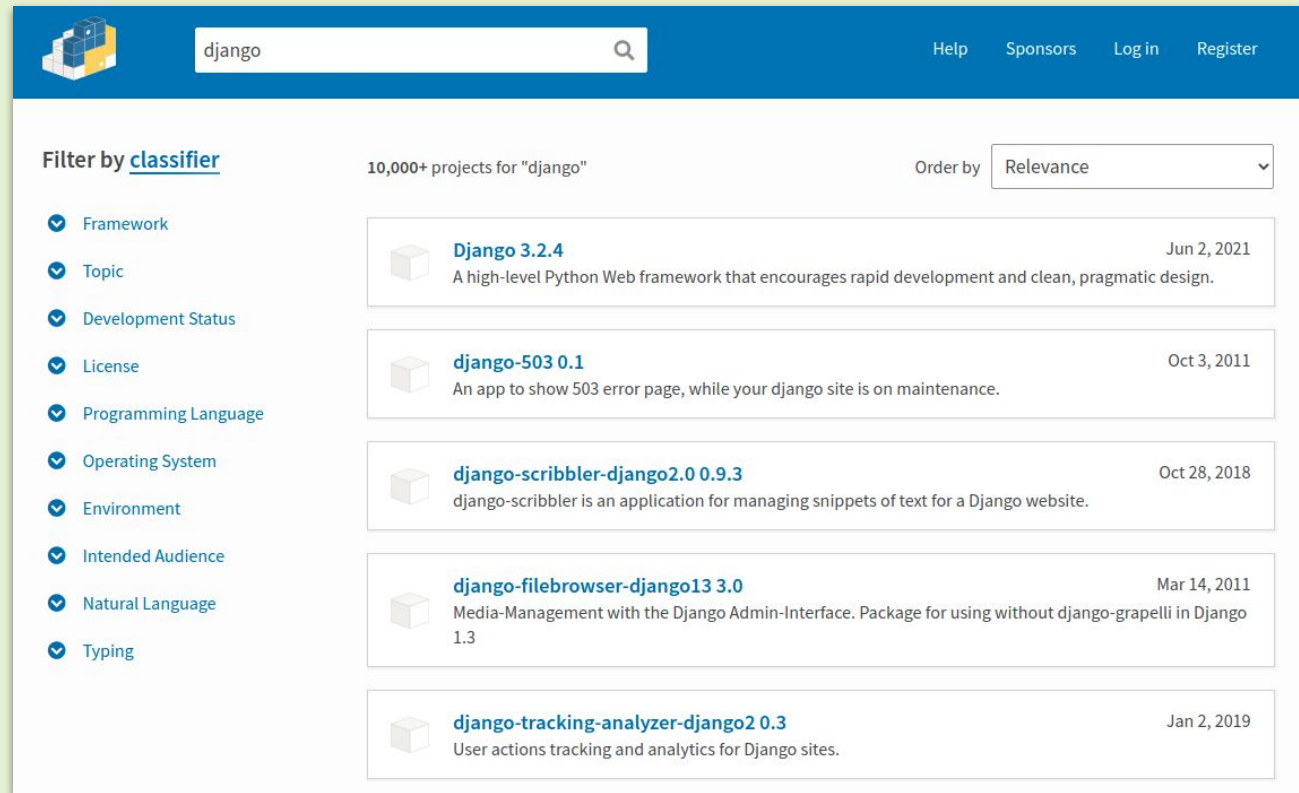
El repositorio público principal es PyPI

Repositorios privados

El administrador paquetes principal es pip

PyPI

<https://pypi.org/>



The screenshot shows the PyPI search results for the package 'django'. The search bar at the top contains 'django' and a magnifying glass icon. To the right of the search bar are links for 'Help', 'Sponsors', 'Log in', and 'Register'. Below the search bar, the text '10,000+ projects for "django"' is displayed. On the left side, there is a 'Filter by classifier' section with a list of classifiers, each with a checkmark icon: Framework, Topic, Development Status, License, Programming Language, Operating System, Environment, Intended Audience, Natural Language, and Typing. On the right side, there is a 'Order by' dropdown menu set to 'Relevance'. The search results are listed in a table-like format with five entries, each showing a package icon, the package name and version, a brief description, and the last update date.

Package Name	Version	Description	Last Update
Django	3.2.4	A high-level Python Web framework that encourages rapid development and clean, pragmatic design.	Jun 2, 2021
django-503	0.1	An app to show 503 error page, while your django site is on maintenance.	Oct 3, 2011
django-scribbler-django2.0	0.9.3	django-scribbler is an application for managing snippets of text for a Django website.	Oct 28, 2018
django-filebrowser-django13	3.0	Media-Management with the Django Admin-Interface. Package for using without django-grapelli in Django 1.3	Mar 14, 2011
django-tracking-analyzer-django2	0.3	User actions tracking and analytics for Django sites.	Jan 2, 2019

(+308.000 proyectos y +500.000 usuarios)

Entornos virtuales

- Permiten aislar instalaciones de Python
- Permiten el uso de diferentes versiones de python y de las mismas librerías
- Ayudan a separar los entornos de ejecución
- Mejoran la mantenibilidad de los proyectos
- venv es el paquete que se utiliza en Python 3
- Crear un virtual en como: `> python -m venv <nombre_venv>`

Control de paquetes

pip es el administrador de paquetes más popular utilizado:

<https://pypi.org/project/pip/>

- `pip install <nombre-paquete>`
- `pip list <nombre>`
- `pip uninstall <nombre-paquete-a-borrar>`
- `pip freeze > requirements.txt`

IDEs y editores para Python

IDEs especializados:

- PyCharm
- Spyder
- Jupyter-notebook
- VisualStudio Code + Python

Editores:

- SublimeText, Atom
- Vim, Eric, Emacs, etc

Distribuciones de Python

- Anaconda
- WinPython
- Enthought Canopy
- ActivePython

Cómo instalar de Anaconda

<https://www.anaconda.com/products/individual#Downloads>

Anaconda Installers

Windows 	MacOS 	Linux 
<p>Python 3.8</p> <p>64-Bit Graphical Installer (477 MB)</p> <p>32-Bit Graphical Installer (409 MB)</p>	<p>Python 3.8</p> <p><input checked="" type="radio"/> 64-Bit Graphical Installer (440 MB)</p> <p>64-Bit Command Line Installer (433 MB)</p>	<p>Python 3.8</p> <p>64-Bit (x86) Installer (544 MB)</p> <p>64-Bit (Power8 and Power9) Installer (285 MB)</p> <p>64-Bit (AWS Graviton2 / ARM64) Installer (413 M)</p> <p>64-bit (Linux on IBM Z & LinuxONE) Installer (292 M)</p>

Instalar anaconda

Linux:

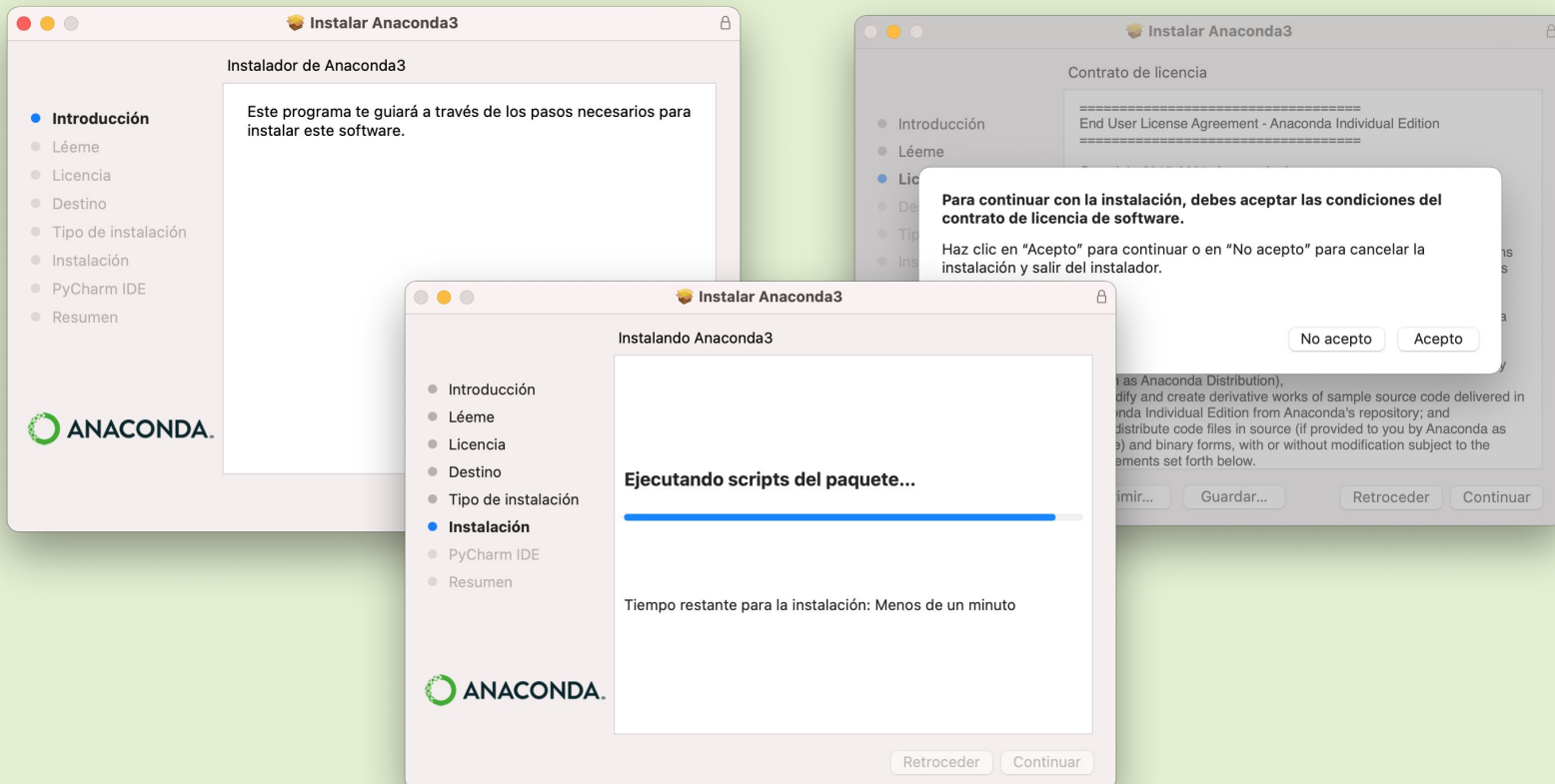
```
> sh ~/Downloads/Anaconda3-2021.05-Linux-x86_64.sh
```

- Aceptamos la licencia
- Confirmamos localización

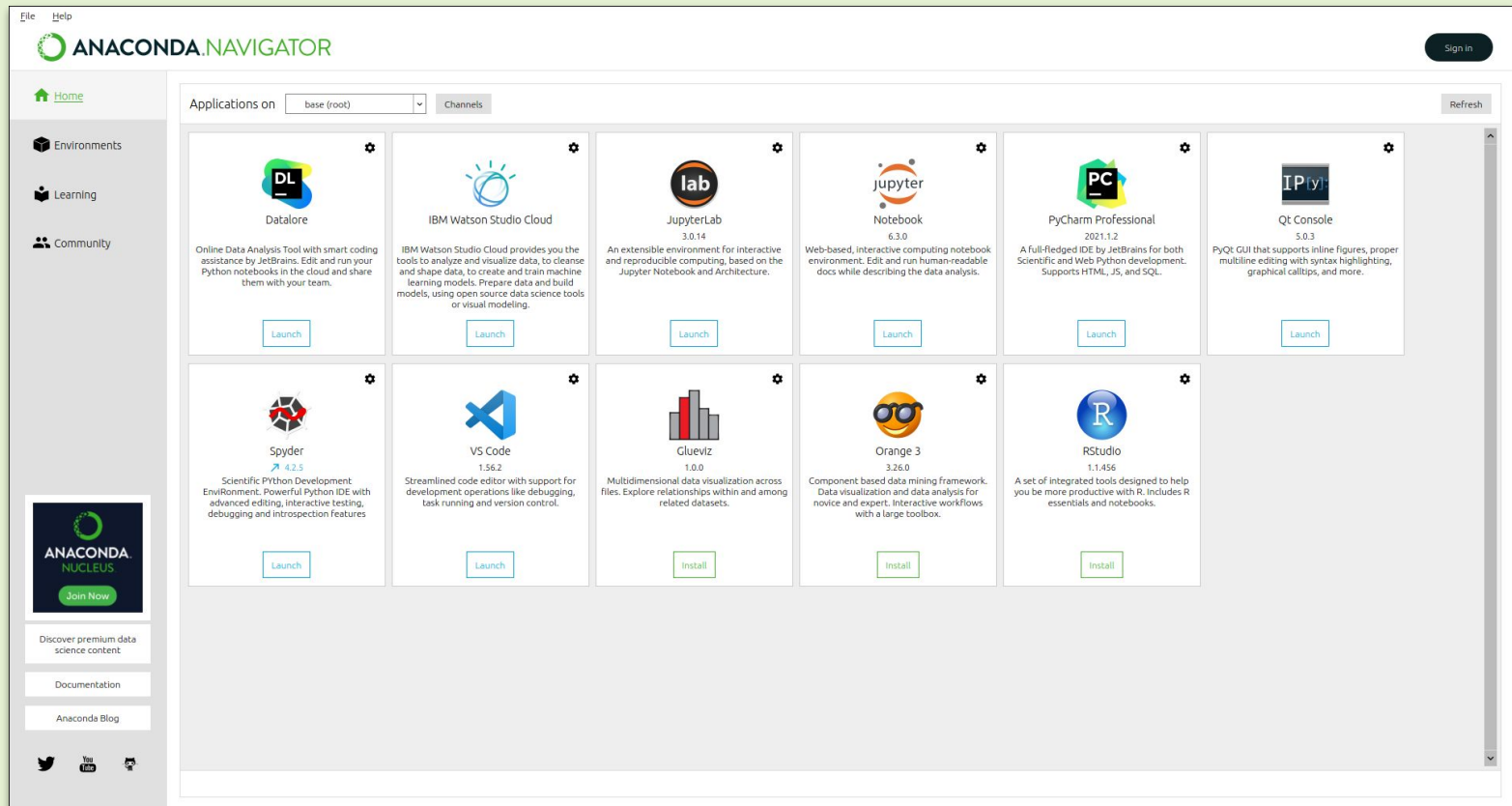
```
> ~/anaconda3/bin/anaconda-navigator
```

Instalar anaconda

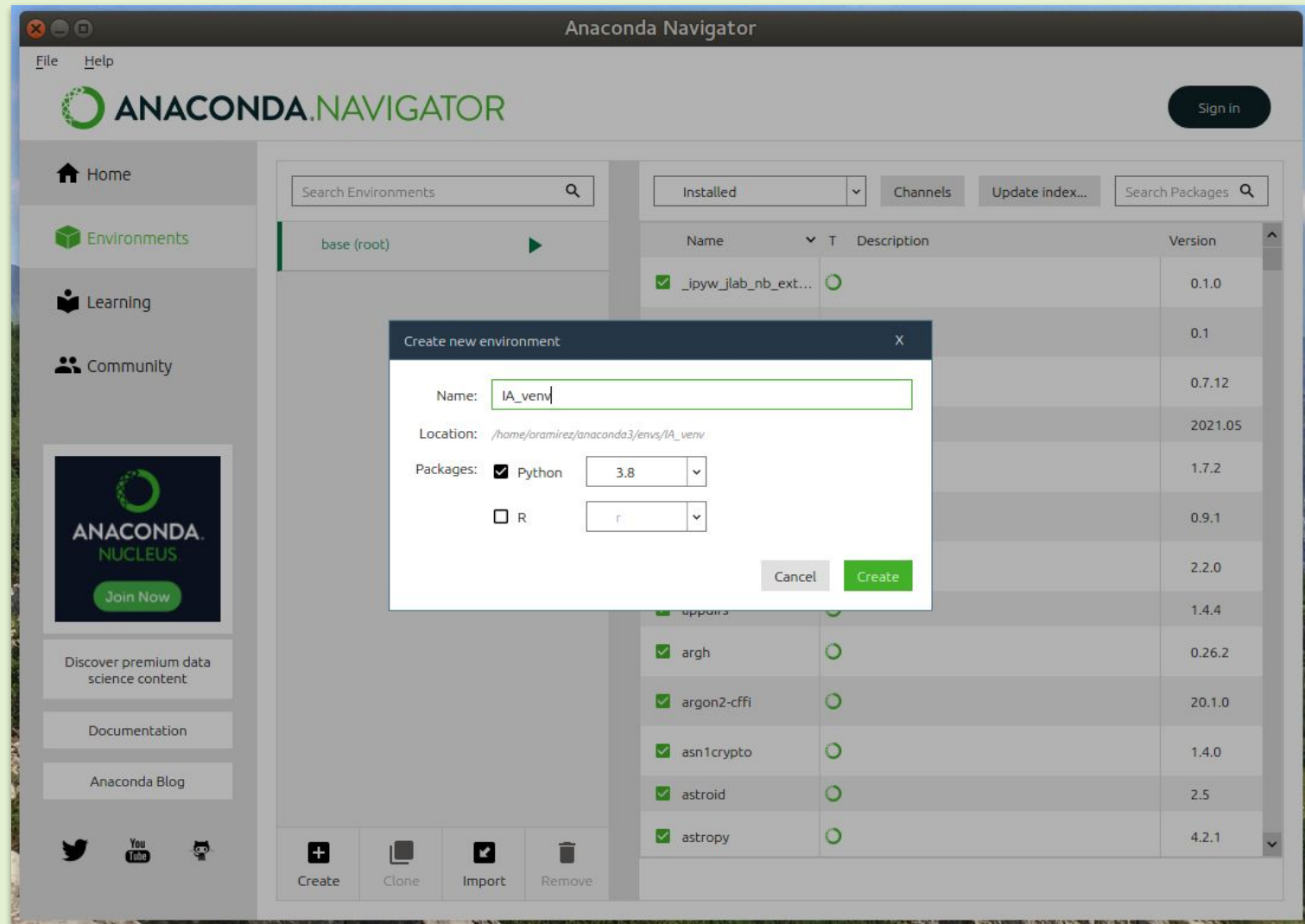
MacOS X y Windows



Explorando Anaconda navigator



Creación de entorno virtual



Búsqueda de paquetes

The screenshot displays the Anaconda Navigator application interface. On the left, a sidebar contains navigation links: Home, Environments, Learning, and Community. Below these is a promotional banner for 'ANACONDA NUCLEUS' with a 'Join Now' button, followed by links to 'Discover premium data science content', 'Documentation', and 'Anaconda Blog'. At the bottom of the sidebar are social media icons for Twitter, YouTube, and GitHub.

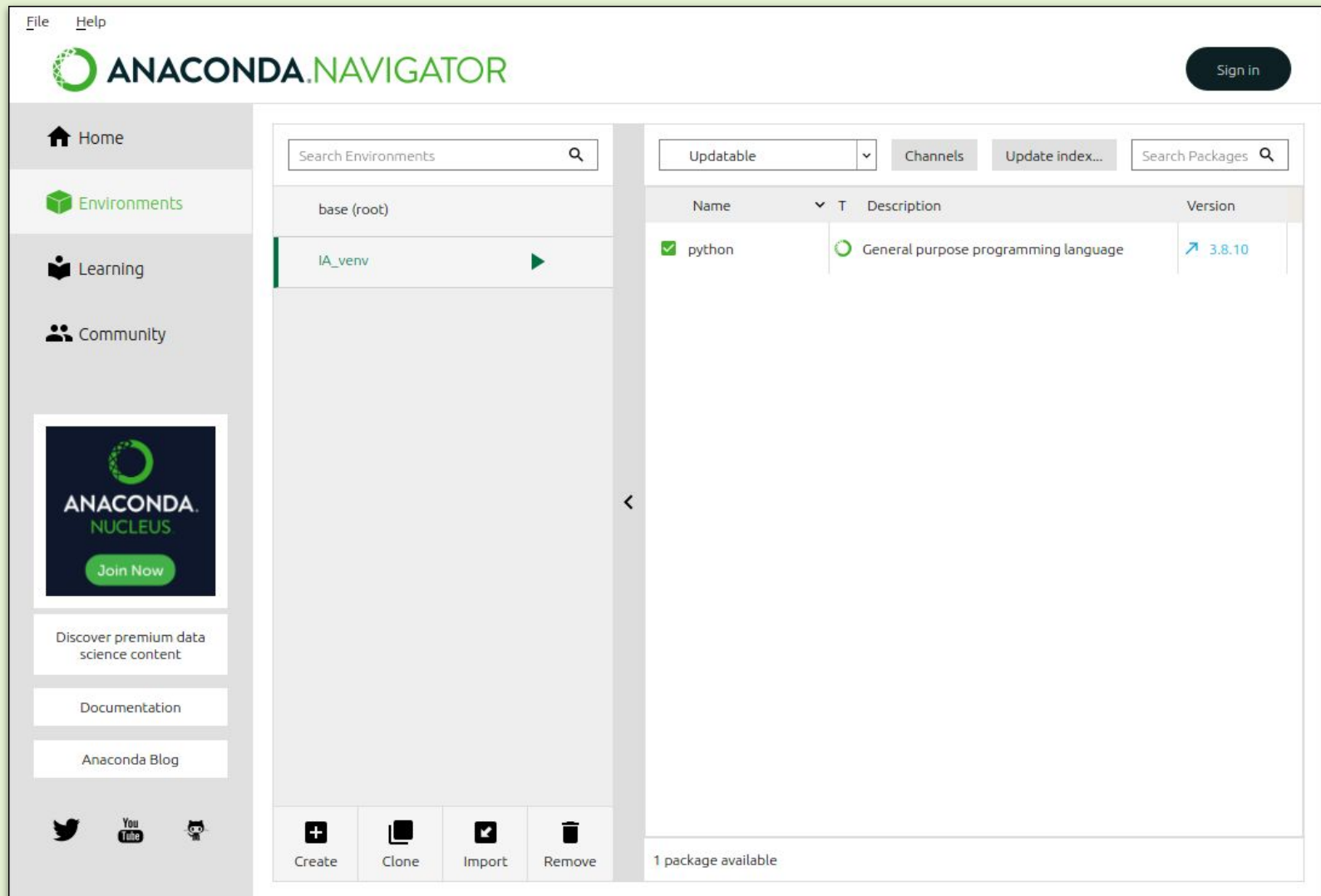
The main area is divided into two panels. The left panel, titled 'Search Environments', shows a search bar with the text 'Search Environments' and a magnifying glass icon. Below the search bar, two environments are listed: 'base (root)' and 'IA_venv', with a green play button next to 'IA_venv'. At the bottom of this panel are four buttons: 'Create', 'Clone', 'Import', and 'Remove'.

The right panel displays the search results for the package 'bokeh'. It features a dropdown menu set to 'Not installed', a 'Channels' button, an 'Update index...' button, and a search input field containing 'bokeh'. Below this is a table with the following data:

Name	T	Description	Version
<input type="checkbox"/> bkcharts	High level chart types built on top of bokeh	0.2	
<input type="checkbox"/> bokeh	Statistical and novel interactive html plots for python	2.3.2	
<input type="checkbox"/> r-bokeh		0.6.3	

At the bottom of the right panel, a status bar indicates '3 packages available matching "bokeh"'.

Actualizaciones de paquetes



The screenshot displays the Anaconda Navigator application interface. The left sidebar contains navigation links for Home, Environments, Learning, and Community. The main panel is divided into two sections: 'Environments' on the left and 'Channels' on the right. The 'Environments' section shows a list of environments, including 'base (root)' and 'IA_venv'. The 'Channels' section displays a table of available packages, with 'python' highlighted as the only package available for update.

Environments

Search Environments
base (root)
IA_venv

Channels

Updatable	Channels	Update index...	Search Packages
Name	Description	Version	
python	General purpose programming language	3.8.10	

1 package available

Enlaces de interés

File Help

ANACONDA.NAVIGATOR [Sign in](#)

Home
Environments
Learning
Community

ANACONDA NUCLEUS
Join Now











Discover premium data science content

Documentation
Anaconda Blog

[Twitter](#) [YouTube](#) [GitHub](#)

Documentation (26) Training (1) Video (21) Webinar (20)

Search

 Python Tutorial Read	 Python Reference Read	 ANACONDA. Anaconda Package List Read	 pandas Pandas Documentation Read
 Numpy Documentation Read	 Scipy Documentation Read	 Matplotlib Documentation Read	 Bokeh User Guide Read
 ANACONDA. Anaconda Cloud Documentation	 ANACONDA. Anaconda Documentation	 ANACONDA. Anaconda Navigator	 The Comprehensive R Archive

Conda

- Gestor de dependencias integrado en anaconda
- Se puede usar por comandos
- Permite el uso de entornos virtuales
- Paquetes gestionados por distribución anaconda

Comandos de conda

Crear entorno

```
conda create --name entorno
```

Activar el entorno en osX

```
source activate nombre_del_entorno
```

Activar el entorno en Windows

```
conda activate nombre_del_entorno
```

https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

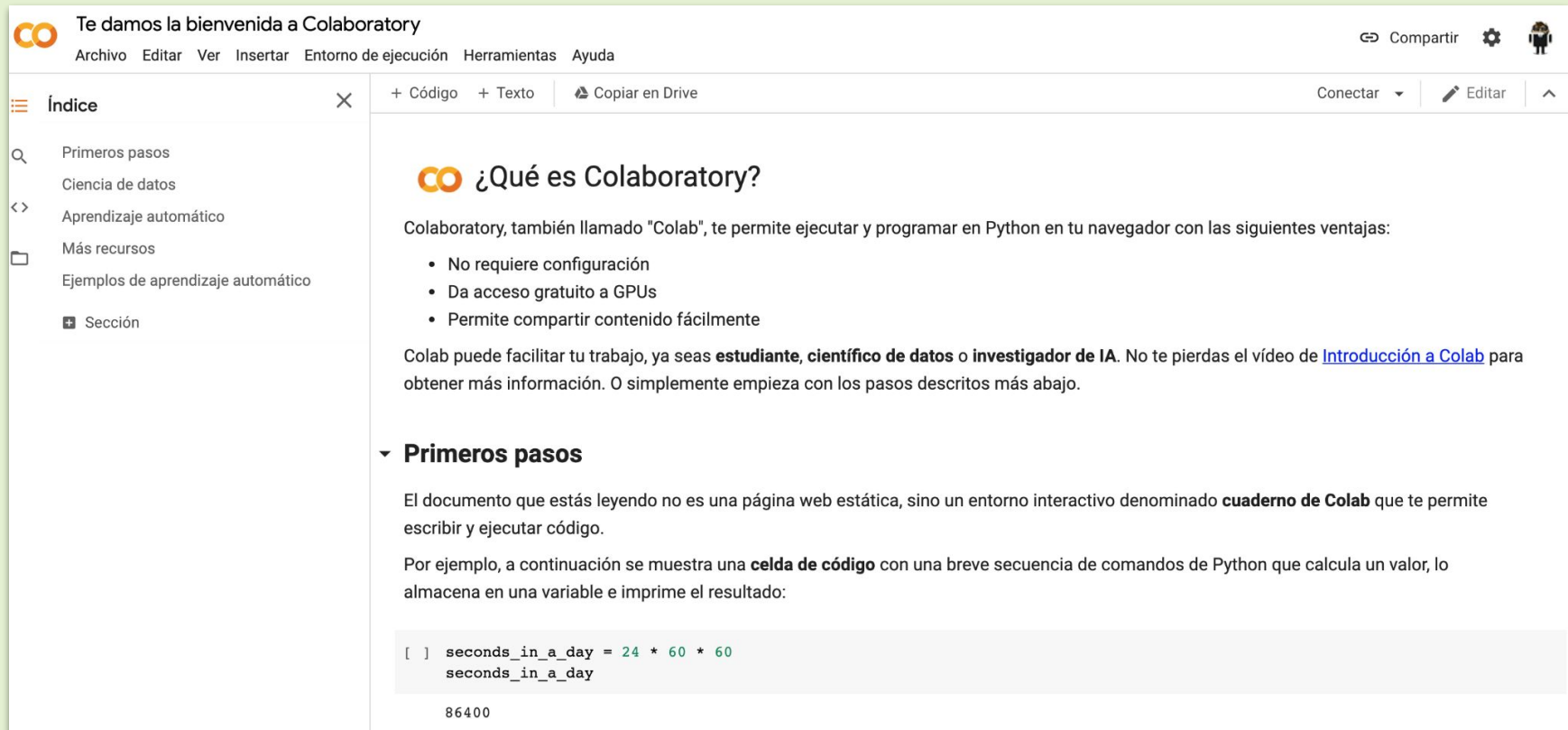
Jupyter notebook

Jupyter es una aplicación web que permite crear estudios bonitos con ecuaciones y gráficos que pueden ser ejecutados y compartidos fácilmente.



Tutorial de Jupyter y de markdown

Google Colab



The screenshot displays the Google Colaboratory (Colab) web interface. At the top, a header bar includes the Colab logo, a welcome message "Te damos la bienvenida a Colaboratory", and navigation links: Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, and Ayuda. On the right side of the header are links for Compartir, settings, and a user profile icon.

Below the header, a sidebar on the left contains an "Índice" (Index) section with a search icon and a list of links: "Primeros pasos", "Ciencia de datos", "Aprendizaje automático", "Más recursos", "Ejemplos de aprendizaje automático", and "Sección".

The main content area features a title "¿Qué es Colaboratory?" with the Colab logo. Below the title, a paragraph explains that Colaboratory (also called "Colab") allows users to execute and program in Python within their browser, listing three advantages:

- No requiere configuración
- Da acceso gratuito a GPUs
- Permite compartir contenido fácilmente

Following this, a paragraph states that Colab can facilitate work for students, data scientists, or AI researchers, and suggests watching a video titled "Introducción a Colab" for more information.

A section titled "Primeros pasos" (First steps) follows, explaining that the document is an interactive "cuaderno de Colab" (Colab notebook) where users can write and execute code. It provides an example of a code cell:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

The output of the code cell is displayed as "86400".

<https://colab.research.google.com/notebooks/intro.ipynb>

Hasta mañana!

