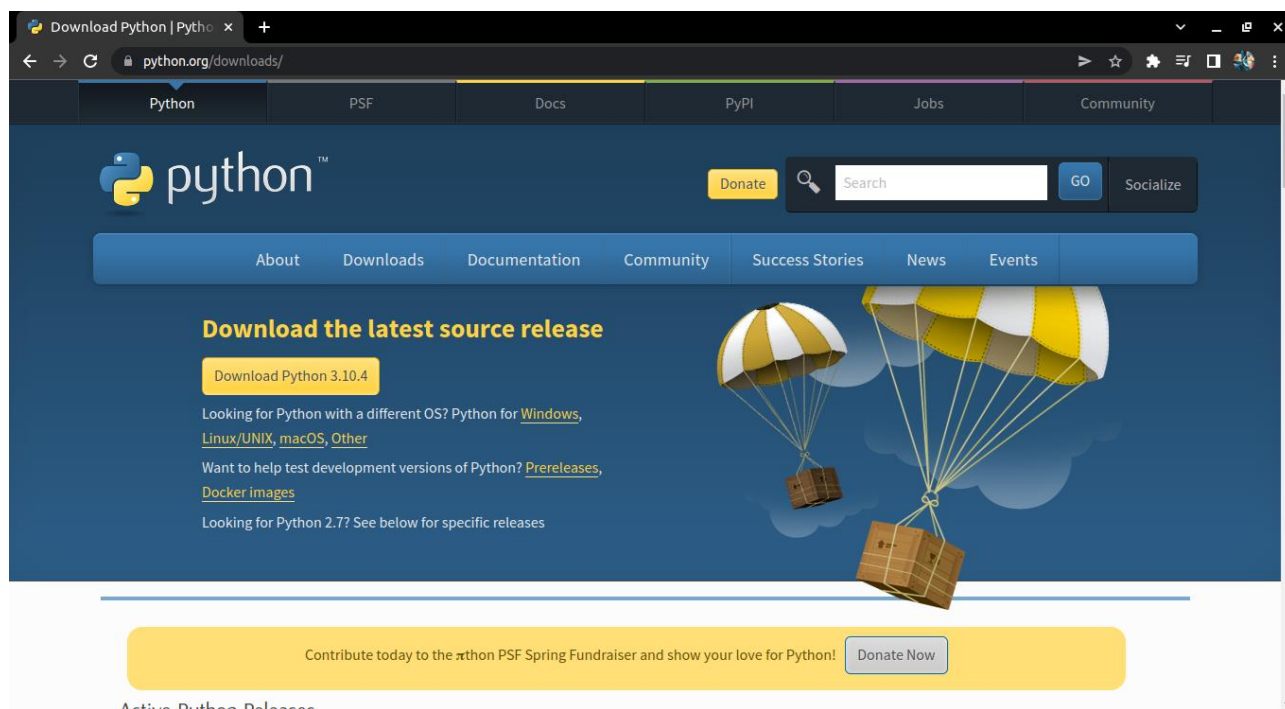


Primeros pasos en Python

.1 Instalar Python en Windows y MacOS

En ambos sistemas operativos la instalación es muy similar.



1. Ir a la página oficial de python <https://www.python.org/downloads/> y descargar la última versión dándole click al botón amarillo.
2. Una vez descargado el archivo, ejecutarlo (con permisos de administrador)
3. Seleccionar Install launcher for all users y Add Python to PATH (importante!)
4. Iniciar la instalación
5. Una vez terminado el proceso, cerrar el mensaje de notificación.

Probando la instalación (Windows)

1. En el botón de inicio, Ejecutar, escribir cmd. Se abrirá la línea de comandos en una ventana negra.
Allí tipear

```
python --version
```


Si todo está correcto, aparecerá la versión de python que hemos instalado.
2. Si escribimos

```
python
```


se nos abrirá la consola/intérprete de python, que podremos distinguir por el prompt `>>>`.
Allí podemos tipear

```
print( 'Hola mundo!' )
```


y habremos escrito nuestra primera línea en python.

.2 El intérprete de Python

El intérprete funciona de manera similar al shell de Unix: cuando se le llama con una entrada estándar conectada a un terminal, lee y ejecuta comandos de manera interactiva; cuando se le llama con un argumento de nombre de archivo o con un archivo como entrada estándar, lee y ejecuta un script desde ese archivo.

Podemos salir del intérprete, tipeando `quit()`

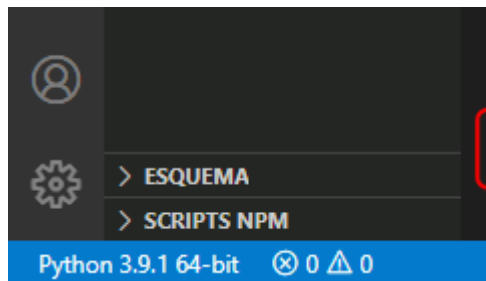
```
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

.3 Instalación en Visual Studio Code

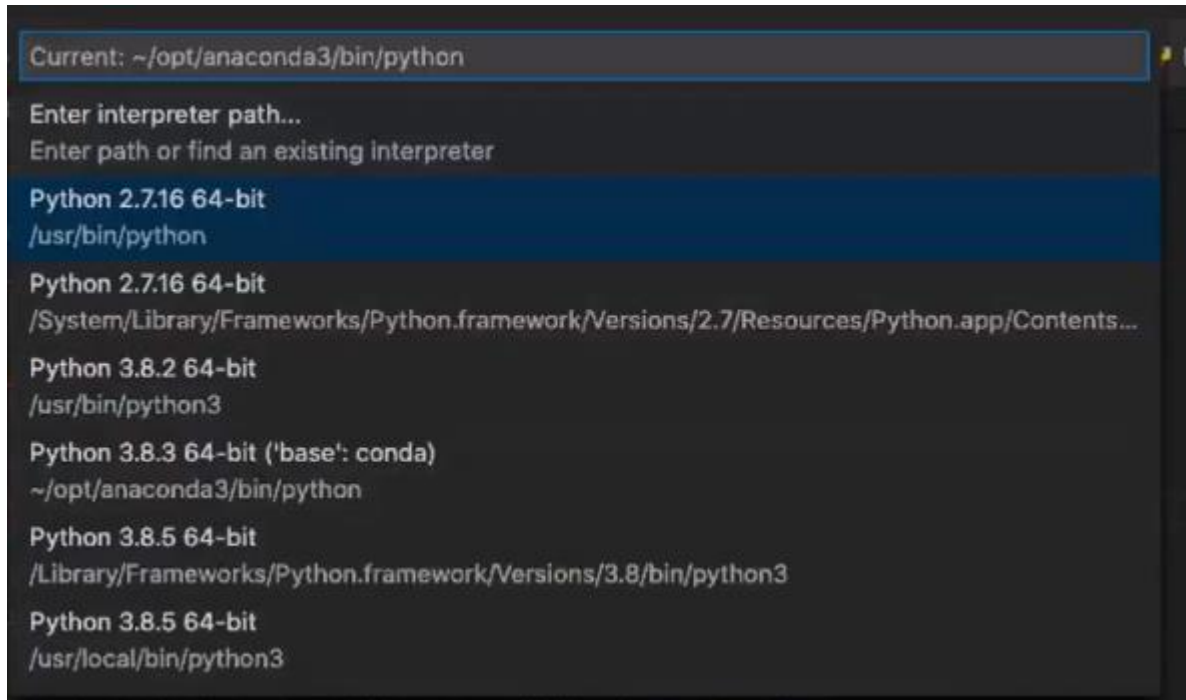
En la pestaña de extensiones, buscar Python. Instalamos la extensión de Microsoft.



Si tenemos varias versiones del intérprete Python, podemos seleccionar cuál queremos usar, pulsando en la barra azul, abajo a la izquierda.



En la parte superior se nos va a desplazar un menú en el que podremos elegir la versión que queremos utilizar.



.4 PIP

Es el instalador de paquetes de Python. Es una utilidad que se ejecuta desde la línea de comandos, que permite instalar, reinstalar o desinstalar paquetes de python.

En versiones superiores s 2.7.9. o 3.4, PIP ya viene instalado con Python por defecto.

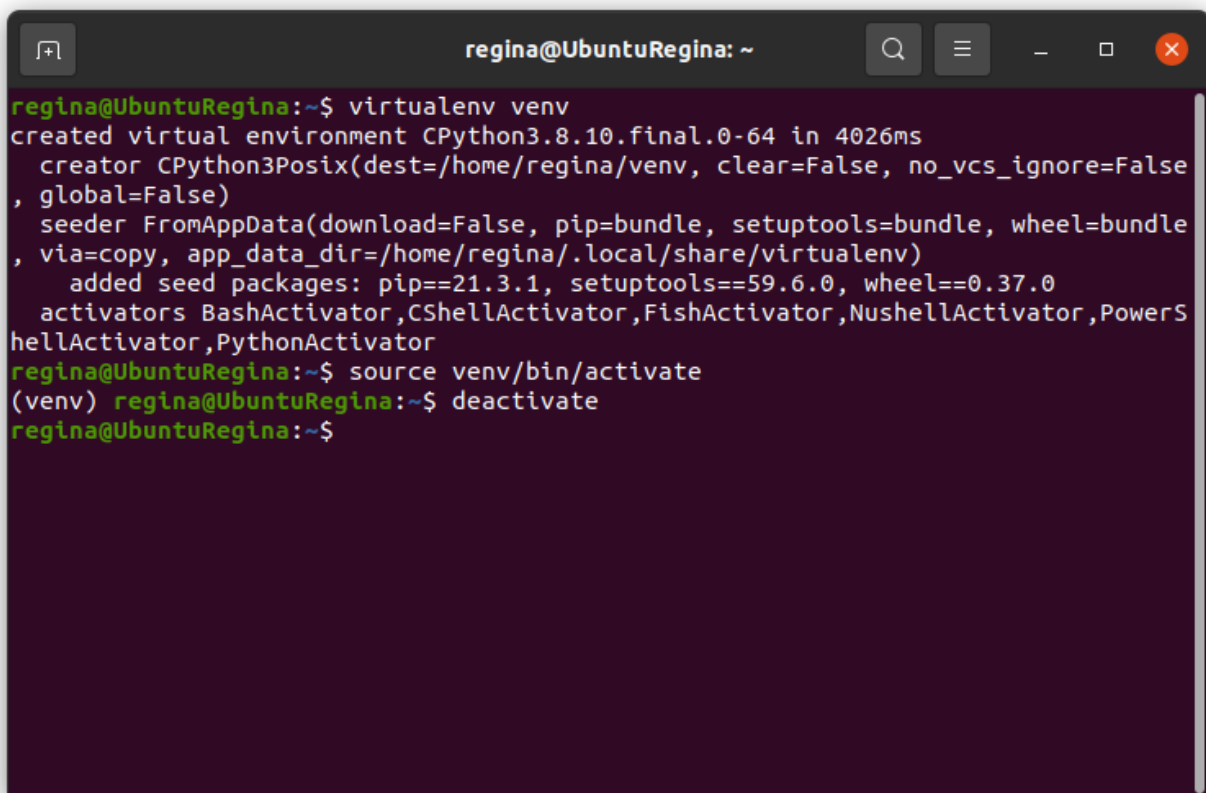
.5 Ambientes virtuales: Virtualenv

Los ambientes virtuales nos permiten encapsular un proyecto para poder instalar versiones de los paquetes que se requieran sin tener que instalarlos en todo el sistema operativo. Esto evita que se produzcan conflictos de paquetes en el intérprete principal.

Para trabajar en entornos virtuales:

1. Instalar el paquete de virtualenv de forma global, ejecutando
`pip install virtualenv`
Para verificar que se instaló correctamente podemos ejecutar `which virtualenv`
2. Crear o seleccionar la carpeta en donde tendremos nuestro entorno virtual, estando ahí ejecutamos lo siguiente para crear el entorno virtual:
`virtualenv nombre_entorno`

- por convención es recomendable llamarlo venv
- Después de crear el entorno virtual debemos activarlo, para ello se ejecuta el siguiente comando:
`source /venv/Scripts/activate` (para windows)
`source /venv/bin/activate` (en linux)
con esto quedará activado y nos aparecerá el nombre del entorno virtual al inicio de la línea en la terminal de comandos, (venv en este caso).
Para desactivarlo sería lo mismo pero al final se coloca
`deactivate`
 - Para ver los paquetes que tenemos instalados en nuestro entorno virtual ejecutamos el siguiente comando:
`pip freeze`
Esto nos mostrará el listado de los paquetes, si no aparece nada es porque no se ha instalado ningún paquete aún.
 - Si queremos tener todos los paquetes agrupados y con su versión correspondiente, podemos hacer uso del archivo requirements.txt en donde colocaremos cada uno de los paquetes
`pip freeze > requirements.txt`
Luego podremos distribuir nuestro trabajo y quienes quieran trabajar sobre él, podrán instalar los paquetes requeridos usando el siguiente comando:
`pip install -r requirements.txt`



```
regina@UbuntuRegina: ~  
regina@UbuntuRegina:~$ virtualenv venv  
created virtual environment CPython3.8.10.final.0-64 in 4026ms  
  creator CPython3Posix(dest=/home/regina/venv, clear=False, no_vcs_ignore=False, global=False)  
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/regina/.local/share/virtualenv)  
    added seed packages: pip==21.3.1, setuptools==59.6.0, wheel==0.37.0  
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator  
regina@UbuntuRegina:~$ source venv/bin/activate  
(venv) regina@UbuntuRegina:~$ deactivate  
regina@UbuntuRegina:~$
```

Tipos de datos en Python

Los tipos de datos se pueden clasificar en:

Mutable: su valor puede cambiarse en tiempo de ejecución.

Immutable: su valor no puede cambiarse en tiempo de ejecución.

Enteros (int)

Representan todos los números enteros (positivos, negativos y 0), sin parte decimal.

Reales (float)

Representa los números reales, tienen una parte entera y otra decimal.

Complejos (complex)

Representan números complejos, con una parte real y otra imaginaria. Son tipos de datos inmutables.

Booleanos

Pueden tener únicamente dos tipos de valores: "True" o "False". Estos sirven para agregar atributos a la variable si es verdadero o falso.

Cadena (str)

Permite almacenar cadenas de caracteres. Es de tipo inmutable.

Lista (list)

Variable que almacenan arrays, internamente cada posición puede ser un tipo de datos distinto. Se escribe entre corchetes, y sus elementos se separan con comas.

Ejemplo: ingredientes = ['huevos', 'harina', 'leche', 'azúcar']

Tupla

Las tuplas son objetos de tipo secuencia, específicamente es un tipo de dato lista inmutable. Esta no puede modificarse de ningún modo después de su creación.

Ejemplo: mi_tupla = ('Python', True, 'ananá', 34)

Diccionario (dict)

Pueden ser creados colocando una lista separada por coma de pares "key:value" entre {}.

Único tipo de mapeo estándar actual.

Ejemplo: mi_dict = {'Ana': 27, 'Juan': 51}

Conjuntos

Un conjunto, es una colección no ordenada y sin elementos repetidos. Los usos básicos de éstos incluyen verificación de pertenencia y eliminación de entradas duplicadas.

set: Mutable, sin orden, no contiene duplicados.

Ejemplo: `my_set = set([3.6, 'Entre Ríos', False])`

frozenset: Inmutable, sin orden, no contiene duplicados.

Ejemplo: `my_fs = frozenset([3.6, 'Entre Ríos', False])`

Enlaces externos

Página oficial de descargas Python: <https://www.python.org/downloads/>
Documentación oficial python en español: <https://docs.python.org/es/3/>