

- Crear un proyecto de Python.
- Desarrollar y ejecutar tu código de forma aislada en una máquina.
- Usar bibliotecas que otra persona haya escrito.
- Restaurar un proyecto a partir de una lista de dependencias.

### **¿Cuál es el objetivo principal?**

Utilizar bibliotecas y planificar tu proyecto para crear programas Python que sean más avanzados.

### **Crear un entorno virtual**

Para crear un entorno virtual, llama al módulo. El módulo espera un nombre como argumento venv.

### **Sigue estos pasos:**

Desde tu terminal ve al directorio donde deseas guardar tu proyecto.  
(Documentos/TuFolderPreferido) Ejemplo en windows: cd Documents/TuFolderPreferido

Utiliza el siguiente comando para llamar al módulo venv. El comando difiere ligeramente dependiendo de tu sistema operativo.

En consola: `python3 -m venv env`

Donde:

python3: La versión de python a utilizar.

venv: Llamada al módulo venv conocido como virtual environment.

env: Nombre de nuestro entorno virtual.

En este punto, se crean algunos directorios:

/env

/bin

/include

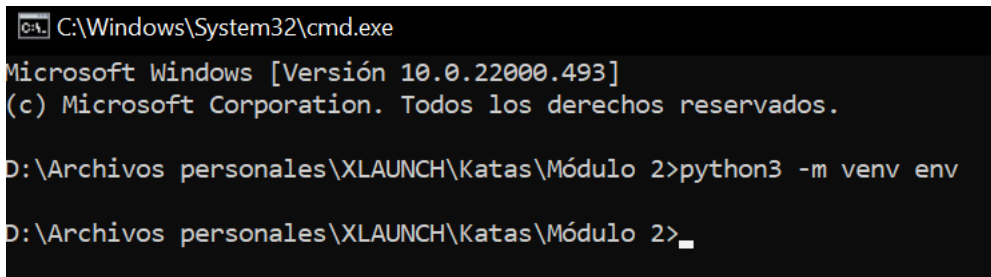
Lib

Tu entorno necesita el directorio venv para realizar un seguimiento de detalles como qué versión de Python y qué bibliotecas está utilizando. No coloque los archivos de programa en el directorio venv. Te sugerimos que coloques tus archivos en el directorio src algo similar. La estructura del proyecto podría verse así:

/env

/src

program.py

A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\System32\cmd.exe'. The window content displays the following text: 'Microsoft Windows [Versión 10.0.22000.493]', '(c) Microsoft Corporation. Todos los derechos reservados.', 'D:\Archivos personales\XLAUNCH\Katas\Módulo 2>python3 -m venv env', and 'D:\Archivos personales\XLAUNCH\Katas\Módulo 2>\_'.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.22000.493]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Archivos personales\XLAUNCH\Katas\Módulo 2>python3 -m venv env
D:\Archivos personales\XLAUNCH\Katas\Módulo 2>_
```

## Activar el entorno virtual

En este punto, tienes un entorno virtual, pero no has comenzado a usarlo. Para usarlo, debes activarlo llamando activate a un script en tu directorio env.

Así es como puede verse la activación en distintos sistemas operativos.

# Bash | Consola

# Windows

env\bin\activate

# Linux, WSL o macOS

source env/bin/activate

Llamar al comando activate cambia el mensaje de salida de la consola. Ahora está precedido con (env) y se parece a este ejemplo:

# Bash | Consola

(env) -> path/to/project

Ahora, ya estás dentro de tu entorno virtual. Cualquier cosa que hagas sucede de forma aislada.

```
D:\Archivos personales\XLAUNCH\Katas\Módulo 2>env\Scripts\activate  
(env) D:\Archivos personales\XLAUNCH\Katas\Módulo 2>
```

### ¿Qué es un paquete?

Una de las principales ventajas de utilizar bibliotecas externas es acelerar el tiempo de desarrollo de tu programa. Puedes obtener una biblioteca de este tipo en Internet. Pero al buscar e instalar estas bibliotecas a través de un entorno virtual, te aseguras de instalar estas bibliotecas solo para el entorno virtual y no globalmente para toda la máquina.

### Instalar un paquete

Instalar un paquete mediante pip. El comando pip utiliza el Python Package Index, o PyPi para abreviar, para saber dónde obtener los paquetes. Puedes visitar el sitio web de PyPi para conocer qué paquetes están disponibles.

### Para instalar un paquete, ejecute, como en este ejemplo: `pip install`

Dato curioso: Si estás desde un notebook se ejecuta así: `!pip install python-dateutil` Con signo de admiración al inicio. Sin embargo, no estamos trabajando con notebooks ahorita, estamos ejecutando todo por terminal (consola, bash, cli, cmd, como sea que le digas).

# Bash | Consola

```
pip install python-dateutil
```

Si ejecutas el comando anterior, descargará e instalará dateutil, un paquete para analizar el formato de archivo .yaml. Después de instalar el paquete, puedes verlo en la lista si expande el directorio lib en env, así:

```
(env) D:\Archivos personales\XLAUNCH\Katas\Módulo 2>pip install python-dateutil
Collecting python-dateutil
  Using cached python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Collecting six>=1.5
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, python-dateutil
Successfully installed python-dateutil-2.8.2 six-1.16.0
WARNING: You are using pip version 21.2.4; however, version 22.0.3 is available.
You should consider upgrading via the 'D:\Archivos personales\XLAUNCH\Katas\Módulo 2\env\Scripts\python.exe -m pip install --upgrade pip' command.

(env) D:\Archivos personales\XLAUNCH\Katas\Módulo 2>
```

# Mensaje de salida en consola

/env

/lib

/dateutil

Para ver qué paquetes están ahora instalados en tu entorno virtual, puedes ejecutar pip freeze. Este comando produce una lista de paquetes instalados en el terminal:

Recuerda: Si deseas instalar un paquete desde un notebook se ejecuta con signo de admiración al inicio. !pip freeze Sin embargo, no estamos trabajando con notebooks ahorita, estamos ejecutando todo por terminal (consola, bash, cli, cmd, como sea que le digas).

# Mensaje de salida en consola

python-dateutil==2.8.2

six==1.16.0

```
(env) D:\Archivos personales\XLAUNCH\Katas\Módulo 2>pip freeze
python-dateutil==2.8.2
six==1.16.0

(env) D:\Archivos personales\XLAUNCH\Katas\Módulo 2>
```

Contiene algo más que sólo pipdate por que en sí misma se basan otras bibliotecas.

Para asegurarte de que estos paquetes solo existen en tu entorno virtual, intenta salir de ese entorno llamando al comando deactivate:

# Bash | Consola

deactivate

Observa cómo cambia el mensaje de la terminal. Ya no está precedido por (env) y ha regresado a su estado anterior:

```
(env) D:\Archivos personales\XLAUNCH\Katas\Módulo 2> deactivate
D:\Archivos personales\XLAUNCH\Katas\Módulo 2>
```

# Bash | Consola

path/to/project

Si ejecutas el comando `pip freeze`, verás una lista mucho más larga de dependencias. Esta lista indica que verás todos los paquetes instalados en tu máquina en lugar de solo lo que está instalado en tu entorno virtual.

Más formas de instalar un paquete

También puedes utilizar los siguientes comandos para instalar un paquete:

Teniendo un conjunto de archivos en tu máquina e instalándolos desde esa fuente:

# Bash | Consola

`cd <to where the package is on your machine>`

`python3 -m pip install .`

Instalar desde un repositorio de GitHub que nos proporciona el control de versiones:

`git+https://github.com/your-repo.git`

Instalar desde un archivo comprimido:

`python3 -m pip install package.tar.gz`

Usar un paquete instalado

Ahora tienes un paquete instalado. ¿Cómo se usa en el código?

Asegúrate de tener un directorio para tus archivos. Te sugerimos que llames al directorio (folder) `src` y agregues un archivo Python llamado `app.py`. Ahora agrega un poco de código para llamar al comando `pipdate`:

```
from datetime import *  
from dateutil.relativedelta import *  
now = datetime.now()  
print(now)  
  
now = now + relativedelta(months=1, weeks=1, hour=10)  
  
print(now)
```