Entornos virtuales

Crear y Usar Entornos Virtuales con Virtualenvwrapper

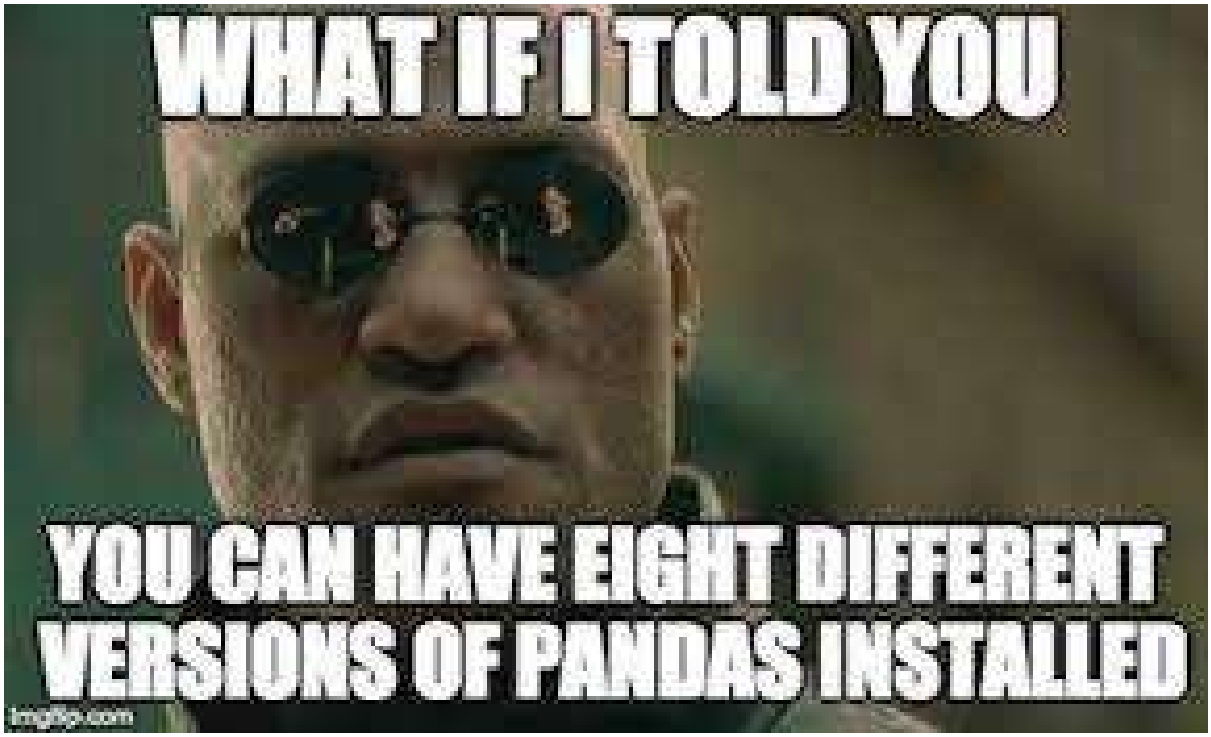
🌐 Introducción

¡Bienvenidos, estudiantes de programación de servicios y procesos! Hoy aprenderemos una práctica esencial en el mundo de la programación: el uso de entornos virtuales. Usaremos la biblioteca **Virtualenvwrapper** en Windows (o **Virtualenvwrapper-win**) y Linux para asegurarnos de que nuestros proyectos sean organizados, libres de conflictos y fáciles de mantener. 🚀

¿Por qué usar Entornos Virtuales? 🤔

Imagina que estás cocinando en una cocina compartida y necesitas varios ingredientes para diferentes recetas. Sería caótico tener todo esparcido por la encimera, ¿verdad? Los entornos virtuales son como estaciones de trabajo independientes en tu cocina, cada una con sus ingredientes propios. Aquí tienes algunas razones para usarlos:

1. **Aislamiento**: Evita conflictos entre diferentes versiones de bibliotecas y paquetes. Un entorno virtual te permite tener una versión de una biblioteca en un proyecto y una versión diferente en otro, sin que choquen.
2. **Portabilidad**: Puedes compartir fácilmente tu proyecto con otros sin preocuparte por las dependencias. ¡Incluso puedes moverlo entre sistemas operativos!
3. **Limpieza**: Mantén tu sistema limpio de paquetes innecesarios. Si algo sale mal en un entorno virtual, puedes desecharlo sin afectar al sistema principal.



Requisitos

Para instalarlo haremos uso de PIP, un manejador de paquetes excepcional para Linux, Windows y MacOS. Para instalarlo debes:

- Tener instalado Python
- Descargar get-pip.py
- Abrir la consola y navegar hasta la ruta donde se descargó el archivo.
- ejecutar desde la consola: python get-pip.py
- Esperar a que termine el proceso.

📜 Configuración Inicial

Antes de empezar, debemos configurar nuestro entorno. En Windows, usaremos **Virtualenvwrapper-win**, mientras que en Linux usaremos **Virtualenvwrapper**. Además, debemos agregar algunas variables al PATH.

Windows:

1. Instala **Virtualenvwrapper-win** usando **pip**:

```
bashCopy code
pip install virtualenvwrapper-win
```

1. Abre el **Explorador de Windows**, haz clic derecho en **Este equipo** y selecciona **Propiedades**. Luego, en la barra lateral izquierda, elige **Configuración avanzada del sistema**.
2. En la pestaña **Opciones avanzadas**, haz clic en el botón **Variables de entorno**.
3. En la sección **Variables del sistema**, busca la variable llamada **Path** y haz clic en **Editar**. Luego, agrega la ruta al directorio de tus entornos virtuales. Por ejemplo, `C:\Users\TuUsuario\Envs`.

Linux:

1. Instala **Virtualenvwrapper** usando **pip**:

```
bashCopy code
pip install virtualenvwrapper
```

1. Agrega las siguientes líneas al final de tu archivo **.bashrc** o **.zshrc**:

```
bashCopy code
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

1. Recarga tu terminal o ejecuta `source ~/.bashrc` (o `source ~/.zshrc` si usas Zsh).

🛠️ Crear un Entorno Virtual

¡Hora de preparar nuestra estación de trabajo! Vamos a crear un entorno virtual para un proyecto imaginario llamado "MiProyecto".

1. Abre tu terminal y ejecuta el siguiente comando:

```
bashCopy code
mkvirtualenv MiProyecto
```

1. ¡Listo! Ahora estás trabajando en tu nuevo entorno virtual. Puedes verificarlo porque verás el nombre del entorno virtual en el indicador de la terminal, por ejemplo: `(MiProyecto)`.

🚀 Activar y Desactivar Entornos Virtuales

Cuando quieras trabajar en un proyecto específico, simplemente activa su entorno virtual:

```
bashCopy code
workon MiProyecto
```

Y cuando hayas terminado, puedes desactivar el entorno virtual:

```
bashCopy code
deactivate
```

🗑️ Eliminar un Entorno Virtual

Si ya no necesitas un entorno virtual, puedes eliminarlo:

```
bashCopy code
rmvirtualenv MiProyecto
```

Problemas con las variables de entorno

Entornos virtuales con Virtualenvwrapper – KoraNet (koranets.net)

Si no funciona la creación de un nuevo entorno, es porque no está configurado correctamente las variables de entorno. Para ello:

- setx WORKON_HOME %USERPROFILE%\Envs
- setx VIRTUALENVWRAPPER_SCRIPT %USERPROFILE%\Envs\Scripts\virtualenvwrapper.sh

Otro modo de configurar la variable de entorno para esta librería es u**sando la interfaz gráfica de Windows**:

- Haz clic derecho en "Este equipo" o "Mi PC" en el escritorio o en el menú Inicio y selecciona "Propiedades".
- En la ventana de "Propiedades del sistema", haz clic en "Configuración avanzada del sistema" en la barra lateral izquierda.
- En la pestaña "Opciones avanzadas", haz clic en el botón "Variables de entorno" en la sección "Variables de sistema".
- En la sección "Variables del sistema", busca la variable **WORKON_HOME**. Si no existe, puedes crearla haciendo clic en "Nueva..." y configurar su valor como **%USERPROFILE%\Envs**.
- A continuación, busca la variable **VIRTUALENVWRAPPER_SCRIPT**. Si no existe, puedes crearla haciendo clic en "Nueva..." y configurar su valor como **%USERPROFILE%\Envs\Scripts\virtualenvwrapper.sh**.

- Haz clic en "Aceptar" para guardar los cambios en las variables de entorno.

¿Por qué hacemos esta configuración?

La variable WORKON_HOME le dice a virtualenvwrapper dónde alojar tus entornos virtuales. Por omisión es $HOME/.virtualenvs. Si el directorio no existe cuando virtualenvwrapper es cargado, éste será creado automáticamente.

Otro tutorial interesante:

Tutorial de Python virtualenvwrapper (rukbottoland.com)

Para que lo pilles si aún no te entra en la cabeza

Imagina que te asignaron una tarea emocionante en el trabajo o en tu proyecto personal: mejorar una función crítica en un repositorio de código existente. El problema es que este repositorio tiene muchas dependencias específicas y tu computadora ya tiene algunas versiones instaladas. Aquí es donde los entornos virtuales se vuelven cruciales.

1. **Creación del Entorno Virtual**:

Primero, creas un entorno virtual llamado "ProyectoGenial" usando Virtualenvwrapper:

```
bashCopy code
mkvirtualenv ProyectoGenial
```

Inmediatamente, tu terminal cambia para indicar que estás en el entorno virtual recién creado: `(ProyectoGenial)`.

1. **Instalación de Dependencias**:

Luego, necesitas instalar todas las bibliotecas específicas del proyecto. Digamos que necesitas las bibliotecas `requests` y `beautifulsoup4`. Utilizas pip para instalarlas:

```
bashCopy code
pip install requests beautifulsoup4
```

Ahora tienes un entorno virtual limpio con las bibliotecas correctas instaladas, sin afectar a las versiones globales de estas bibliotecas en tu sistema.

1. **Trabajo en el Proyecto**:

Trabajas en tu tarea con total tranquilidad, sabiendo que estás en un entorno aislado. Puedes modificar archivos, agregar código y experimentar sin preocuparte por afectar otras partes de tu sistema.

1. **Finalización del Trabajo**:

Cuando terminas tu tarea y quieres liberar recursos en tu computadora, simplemente desactivas el entorno virtual:

```
bashCopy code
deactivate
```

El entorno virtual se cierra, y vuelves al entorno de usuario normal. El espacio y los recursos que utilizaba el entorno virtual se liberan.

1. **Eliminar el Entorno Virtual (Opcional)**:

Si sabes que no volverás a trabajar en ese proyecto en un tiempo, puedes eliminar el entorno virtual para liberar aún más espacio:

```
bashCopy code
rmvirtualenv ProyectoGenial
```

No te preocupes, puedes crearlo nuevamente y reinstalar las dependencias cuando necesites retomar el proyecto.

Esta anécdota ilustra cómo los entornos virtuales te permiten trabajar en proyectos específicos de manera ordenada y segura, sin temor a afectar otros proyectos o el

funcionamiento de tu computadora. ¡Una práctica esencial para cualquier desarrollador! 🚀👩‍💻👨‍💻

Imagina que te asignaron una tarea emocionante en el trabajo o en tu proyecto personal: mejorar una función crítica en un repositorio de código existente. El problema es que este repositorio tiene muchas dependencias específicas y tu computadora ya tiene algunas versiones instaladas. Aquí es donde los entornos virtuales se vuelven cruciales.

1. **Creación del Entorno Virtual**:

Primero, creas un entorno virtual llamado "ProyectoGenial" usando Virtualenvwrapper:

```
bashCopy code
mkvirtualenv ProyectoGenial
```

Inmediatamente, tu terminal cambia para indicar que estás en el entorno virtual recién creado: `(ProyectoGenial)`.

1. **Instalación de Dependencias desde `requirements.txt`**:

En lugar de instalar las dependencias manualmente, puedes utilizar un archivo `requirements.txt` que lista todas las bibliotecas necesarias junto con sus versiones. Esto asegura que todas las dependencias se instalen correctamente y en las versiones correctas.

Primero, asegúrate de que tu proyecto tenga un archivo `requirements.txt` con contenido como este:

```
plaintextCopy code
requests==2.26.0
beautifulsoup4==4.10.0
```

Luego, utiliza el siguiente comando para instalar todas las dependencias desde el archivo `requirements.txt`:

```
bashCopy code
pip install -r requirements.txt
```

Esto instalará todas las bibliotecas y versiones enumeradas en el archivo de requisitos.

1. **Trabajo en el Proyecto**:

Trabajas en tu tarea con total tranquilidad, sabiendo que estás en un entorno aislado. Puedes modificar archivos, agregar código y experimentar sin preocuparte por afectar otras partes de tu sistema.

1. **Finalización del Trabajo**:

Cuando terminas tu tarea y quieres liberar recursos en tu computadora, simplemente desactivas el entorno virtual:

```
bashCopy code
deactivate
```

El entorno virtual se cierra, y vuelves al entorno de usuario normal. El espacio y los recursos que utilizaba el entorno virtual se liberan.

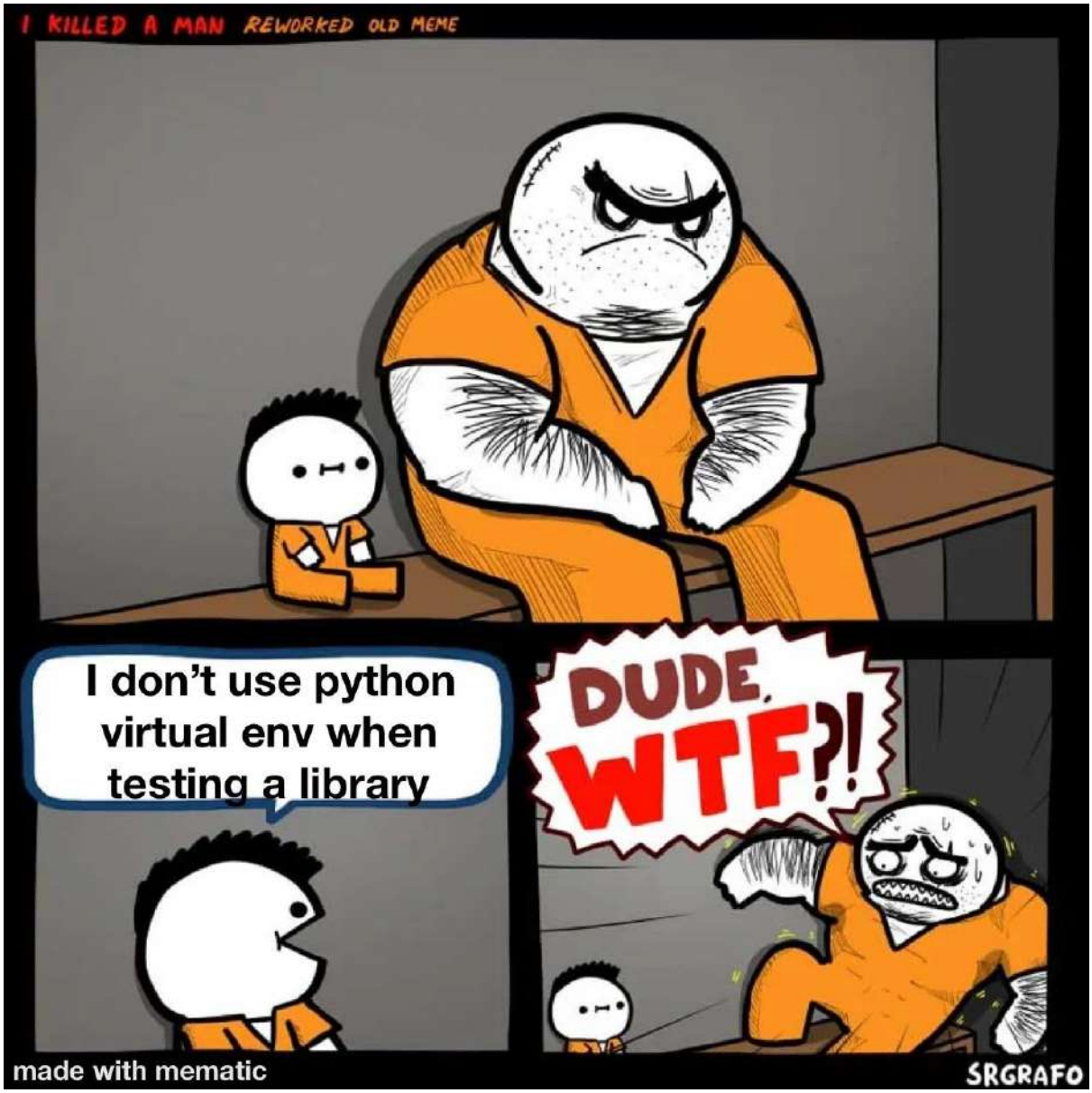
1. **Eliminar el Entorno Virtual (Opcional)**:

Si sabes que no volverás a trabajar en ese proyecto en un tiempo, puedes eliminar el entorno virtual para liberar aún más espacio:

```
bashCopy code
rmvirtualenv ProyectoGenial
```

No te preocupes, puedes crearlo nuevamente y reinstalar las dependencias cuando necesites retomar el proyecto.

Esta anécdota ilustra cómo los entornos virtuales y el uso de un archivo `requirements.txt` simplifican la gestión de dependencias y te permiten trabajar en proyectos específicos de manera ordenada y segura. ¡Una práctica esencial para cualquier desarrollador! 🚀👩‍💻👨‍💻



🧐 Conclusión

¡Felicidades! Ahora sabes cómo usar entornos virtuales con Virtualenvwrapper en Windows y Linux. Esta práctica te ayudará a mantener tu espacio de trabajo ordenado y a evitar conflictos entre proyectos. Recuerda que es esencial en el mundo de la programación. 🌟

Anécdota: Una vez, un desarrollador olvidó usar entornos virtuales y actualizó una biblioteca crucial en su proyecto, ¡causando estragos en todo el sistema! No seas ese desarrollador. 😄

¡A programar sin límites y con entornos virtuales bien organizados! 🚀👩‍💻👨‍💻