Práctica 3. Gestión de dependencias en Python (y entornos virtuales)

Disponible: 25 de septiembre 13:00h

Entrega: 2 de octubre. 23:55h

1 Introducción

IMPORTANTE: la práctica comienza en la Sección 2. Lo que se explica hasta esa sección es de repaso, podéis pasar directamente al enunciado de la práctica.

Para este pequeño tutorial vamos a realizar todos los comandos para crear los entornos virtuales en una consola de la máquina virtual. **Por lo tanto, se va a hacer uso de los comandos explicados a continuación y de los que se encuentran en el pdf de entornos virtuales, disponible en la pestaña de manuales de instalación y uso del software. Se asume que estamos en un entorno Linux (máquina virtual) o cluster.**

1.1 Pip

Pip es el instalador de paquetes de python por defecto. Lo utilizaremos para instalar cualquier paquete que necesite nuestro proyecto. Su uso es muy sencillo, si por ejemplo queremos instalar numpy, ejecutamos:

```
pip install numpy
```

Igualmente, podemos indicar exactamente qué versión del paquete queremos instalar con == (¡cuidado con los espacios!):

```
pip install numpy==1.26.4
```

Hay que tener en cuenta que normalmente los paquetes/librerías en Python no son normalmente independientes y cada paquete requerirá normalmente otros. Al instalar un paquete con pip se instalarán también sus dependencias, es importante comprobar que las dependencias de los distintos paquetes en un proyecto no son incompatibles entre sí.

Hay muchas opciones disponibles en pip, algunas de las más interesantes son:

1. **install/uninstall**: instala/desinstala un paquete
2. **list** y **freeze**: muestran todos los paquetes instalados con sus respectivas versiones.
3. **show**: seguido del nombre de un paquete, nos dará información sobre este, versión, dependencias, etc.

Igualmente, dentro de la instrucción install, existen muchas opciones:

1. **--upgrade**: actualiza un paquete a su última versión
2. **--force-reinstall**: instala el paquete aunque ya esté instalado (es equivalente a ejecutar primero uninstall y después install)

3. **--ignore-installed**: instala ignorando los paquetes ya instalados, puede dar problemas con las dependencias de cada paquete.

4. **--no-deps**: no instala las dependencias del paquete.

Como en cualquier otro módulo de Python, podéis ver el resto de opciones ejecutando **pip -h** o **pip install -h**

1.2 Entornos virtuales en Python

Un entorno virtual de Python es una herramienta que nos permite desarrollar nuestro proyecto de forma autocontenida sin interferencia entre paquetes y versiones de los distintos proyectos. No hay límite al número de entornos virtuales que podemos crear, **siendo aconsejable que cada proyecto tenga su propio entorno**. En concreto, un entorno virtual nos permite:

1. Evitar problemas con las dependencias de los distintos paquetes (un proyecto utiliza un paquete A que es incompatible con el paquete B que utiliza otro proyecto), o distintas versiones de un mismo paquete.

2. Administrar todas las dependencias en un fichero de *requirements* que nos permita desplegar nuestro proyecto de forma rápida en distintas máquinas

3. Poder instalar paquetes en máquinas en las que no tenemos permisos de administrador

Crear un entorno virtual en Python es muy sencillo y existen varias formas para hacerlo, vamos a ver dos de las más utilizadas:

1.2.1 Venv

Venv es un módulo que es parte de la distribución estándar de Python, no requiere por lo tanto de ninguna instalación extra. Para utilizarlo, simplemente hay que ejecutar este comando (si venv no está instalado y nuestra versión de python es la 3.12. Si es distinta a la 3.12, habrá que usar otra):

```
sudo apt update
sudo apt upgrade
sudo apt install python3.12-venv
sudo apt install python3-virtualenv
```

Y después:

```
python3 -m venv ruta_venv/
```

donde *ruta_venv/* será el directorio donde queremos que se cree el entorno virtual. Hay veces que dependiendo de la versión de Python que esté instalada, no es necesario poner python3, sino solamente python.

1.2.2 Virtualenv

Virtualenv es un paquete que amplía la funcionalidad de venv. En este caso no se incluye en la distribución de Python y es necesario instalarlo a través de pip o también mediante el comando:

```
sudo apt install python3-virtualenv
```

En particular, es interesante cuando tenemos varias versiones de Python en nuestra máquina y queremos seleccionar qué versión utilizar en el entorno virtual. En este caso podemos lanzar:

```
virtualenv -p ruta_python/ ruta_venv/
```

con *ruta_python* la ruta de la versión de Python que queremos utilizar.

1.2.3 Uso de los entornos virtuales

Una vez hemos creado el entorno virtual con alguno de los métodos anteriores, necesitamos activarlo. Para ello, si lo hemos instalado en ruta_venv/ ejecutaremos:

```
source ruta_venv/bin/activate si estamos en Linux

source ruta_venv/Scripts/activate en Windows\newline
```

Una vez hecho esto, todo los paquetes que instalemos con pip se instalarán en este entorno virtual. Para volver a la distribución global de Python en nuestra máquina ejecutaremos:

deactivate

1.3 Requirements.txt

Es una práctica habitual en todos los proyectos crear un fichero en el que tengamos todos los paquetes y versiones utilizadas. De esta forma, si se trabaja en equipo o en distintas máquinas podemos utilizar exactamente la misma distribución y levantar el entorno de forma rápida. A este fichero se le suele llamar por convenio *requirements.txt* y en cada linea contiene el nombre del paquete con su versión **exactamente igual que si las ejecutásemos directamente con pip install desde la terminal**.

```
anyio==3.1.0
appnope==0.1.2
argon2-cffi==20.1.0
astropy==5.0
async-generator==1.10
attrs==21.2.0
Babel==2.9.1
backcall==0.2.0
bleach==3.3.0
```

La forma de instalar el fichero de requirements es con el flag **-r** en la instrucción de pip install, y *suele ser lo primero que se hace al crear un nuevo entorno.*

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

2 Ejercicios de la práctica

Para la realización de esta práctica, se hará uso del cluster o de la máquina virtual como se explicó en clase, aunque se pedirá ejecutar algo en el cluster. Deberás entregar todos los comandos utilizados (de la misma manera que hiciste en la práctica 1). Si en la máquina virtual no tienes instalado python3.12, deberás realizar los siguientes comandos. **En principio no tendrás que hacer nada, ya que está todo configurado.**

```
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
sudo apt install python3.12
sudo apt install python3.12-venv
sudo apt install python3-virtualenv
```

Recuerda que una forma de comprobar que tienes python3.12 instalado es ver si te reconoce el comando python3.12, o si al ver la versión de python3 te devuelve python3.12.

Además, en algunos apartados hay preguntas que deberás contestar, y en otros se especificará que deberás aportar código extra en otros ficheros. Por lo tanto, cada alumno deberá entregar un fichero pdf siguiendo el siguiente formato (sin tildes, espacios ni eñes):

```
<nombreAlumno>_<Apellido1>_<Apellido2>_Gestion_Dependencias.pdf
```

Por ejemplo:

```
Pablo_DelasHeras_Fernandez_Gestion_Dependencias.pdf
```

Tanto el pdf como el resto de ficheros que se vayan pidiendo a lo largo de la práctica deberán entregarse en un zip siguiendo este formato:

```
<nombreAlumno>_<Apellido1>_<Apellido2>_Gestion_Dependencias.zip
```

```
Pablo_DelasHeras_Fernandez_Gestion_Dependencias.zip
```

En **imprescindible** que en el pdf se responda a cada apartado de manera ordenada, con el/los comandos necesarios que se puedan copiar y pegar. Recuerda que en cada apartado, habrá que poner los comandos que has empleado. En algunos casos se solicitarán pantallazos. Es decir, se debe indicar el **nombre** del **ejercicio**, el apartado en **cuestión** y su **respuesta** (no es válido indicar todos los comandos sin ningún tipo de orden). **Por otro lado, estará prohibido entregar en moodle los directorios de los entornos virtuales creados (no se evaluará la práctica). Es decir, crearás los entornos virtuales, trabajarás con ellos, pero no los entregarás, solo entregarás el pdf requerido y los otros ficheros solicitados.**

2.1 Ejercicio 1

1. Crea un directorio evitando los posibles errores que puedan surgir (p.e. si ya existe) siguiendo este formato: Practica3-<clave_alumno>, sin espacios, ni tildes y sin los símbolos de < y >. Realiza todos los apartados en ese directorio, por lo que deberás situarte en él.

2. Comprueba la versión de Python3 que tienes instalada en el sistema (puede que tengas que hacer uso de python3 en lugar de python).

3. Crea un entorno virtual de Python con la versión python3.12 en el directorio indicado en el apartado 1, en un subdirectorio llamado "entorno312-<nombre_alumno>" (es decir, que "entorno312-<nombre_alumno>' esté dentro del primer directorio creado y nombre_alumno sea el nombre de pila del alumno y sin contener los símbolos de < y >). Ya sabéis que si os devuelve un error venv, tendréis que ejecutar:

   ```
   sudo apt install python3.12-venv
   ```

   Recordad que debe ser la versión 3.12 de python, por lo que es posible que para crear el entorno virtual, en lugar de hacerlo con python3 tengáis que hacerlo con python3.12

4. Activa el entorno virtual que acabas de crear (todo el resto de comandos, salvo que se indique lo contrario, tienes que realizarlos con el entorno virtual activado). Adjunta un pantallazo en el pdf de la terminal mostrando el entorno virtual activado.

5. Comprueba los paquetes que hay instalados en el entorno virtual nada más activarlo (muestra un pantallazo de los paquetes instalados) y actualiza pip a su última versión.

6. Muestra los paquetes que hay instalados en el entorno virtual después de actualizar pip. Indica si ha cambiado algo. Desactiva dicho entorno y muestra los paquetes que hay instalados en la versión de Python3.12 del sistema. Después de compararlos, vuelve a dejar activo el entorno virtual (los siguientes apartados, salvo que se diga lo contrario, deberán realizarse con el entorno virtual activo).

7. Instala la versión 2.3.2 de numpy, y la 1.13.0 de scipy (con el entorno virtual activo). Intenta hacer el comando en una línea. Describe qué observas adjuntando un pantallazo.

8. Si se ha instalado alguno de los paquetes, desinstálalos. Analizando la información de los resultados de la terminal, intenta instalar de nuevo los paquetes anteriores encontrando versiones compatibles manteniendo fija la versión de numpy o de scipy estipulada en el apartado anterior (la que prefieras). Adjunta un pantallazo.

9. Instala la versión de pandas 2.3.0. ¿Cuáles son las dependencias de pandas? Deberás entregar el comando utilizado para verlo y especificar cuales son los paquetes de los que depende.

10. Desinstala los paquetes que has instalado (intenta hacerlo en una sola línea). Comprueba luego que efectivamente se han desinstalado los paquetes. Adjunta un pantallazo.

2.2 Ejercicio 2 (sigue con el entorno virtual anterior)

1. Crea un fichero requirements.txt con las versiones 2.3.2 de numpy, la versión 2.3.0 de pandas y la 1.13.0 de scipy e instálalo. ¿Hay algún error? (deberás aportar una captura del fichero requirements.txt así como el comando para instalarlo).

2. ¿Puedes utilizar la información del error para crear un requirements válido? mantén las versiones de scipy y pandas fija.

3. ¿Crees que la solución adoptada es adecuada para la distribución del paquete/proyecto en el que estés trabajando?

4. Aunque suele ser suficiente con escribir únicamente los paquetes *extra* que instalamos en los requirements, en ocasiones puede ser recomendable que este contenga todos los paquetes que contenga nuestro entorno de Python. Con lo que hemos visto, ¿se te ocurre cómo hacerlo?. Muestra el comando empleado para mostrar el contenido del requirements.txt y muestra un pantallazo de dicho contenido.

5. Desinstala todos los paquetes instalados, comprueba que se hayan desinstalado completamente, desactiva el entorno y elimina completamente el directorio de entorno312-<nombre_alumno>.

2.3 Ejercicio 3

1. Crea un entorno virtual **con virtualenv** en una carpeta que se llame venv-<nombre_alumno>312 con la versión de Python 3.12 y actívalo (es posible que necesites permisos de superusuario). Recuerda indicar solo el nombre de pila y sin los símbolos de < y >. **Si te falla la creación con virtualenv, hazlo con venv, pero deberás adjuntar un pantallazo de que ha dado error virtualenv**. **Es posible que en el cluster no funcione virtualenv. En ese caso, adjunta pantallazo del error y hazlo con venv**.

2. Usa ahora el código llamado "plot_color_quantization.py" que se encuentra en el material de la P3. Indica el comando empleado para cambiar el nombre del fichero a Ejercicio3<nombre_alumno>.py (con tu nombre de pila y sin los símbolos de < y >). Además, después de los plt.imshow y plt.show, añade una linea con plt.savefig ('nombre_figura'.png) para guardar las figuras creadas. Adjunta también el fichero .py en la entrega.

3. Crea un fichero de requirements312.txt con todos los paquetes que necesites instalar para ejecutar el script. Instala la versión más actualizada posible de cada paquete. Comprueba que puedes ejecutar el script generando las imágenes solicitadas.

4. Como habréis visto, cada vez que instaláis una librería, ésta instala a su vez otras que necesita para su funcionamiento (siempre que no estén ya instaladas). Esto implica que en los requirements.txt no se encuentran todos los paquetes realmente instalados y en proyectos más complejos puede haber problemas con las dependencias. Existen algunas librerías como poetry o pip-tools que aportan funcionalidades extra a la hora de manjear las dependencias de un proyecto.

   Crea un nuevo fichero requirements312_final.txt (deberás adjuntarlo o una captura del mismo) utilizando pip-tools (es posible que tengas que ejecutar antes de nada pip install pip-tools). Lee la documentación del paquete para ver cómo hacerlo. **Tendrás que crear un requirements.in y luego compilarlo**. Deberás copiar todos los comandos utilizados, además de otros ficheros que necesites crear para ejecutar este apartado. Desactiva el entorno virtual al final y elimina el directorio.

2.4 Ejercicio 4

Aunque estamos ejecutando todas estas instrucciones una a una desde la terminal, también podemos implementarlo en un único fichero bash que pueda

ejecutar todo a la vez. Crea ahora un fichero bash (que deberás añadir a la entrega) con el nombre:

```
<nombreAlumno>_<Apellido1>_<Apellido2>
    _Gestion_Dependencias.sh
```

y que realice lo siguiente:

- Crear un entorno virtual nuevo con la versión de python 3.12 y activar dicho entorno. Utiliza venv.
- Instalar los paquetes del requirements (crea un nuevo requirements llamado requirements4_1.txt **(puedes crearlo a mano)** y entrégalo también) empleados para hacer funcionar el fichero de plot_color_quantization.py. **Recuerda que para instalar sklearn, hay que instalar scikit-learn**.
- Ejecutar dicho fichero y redirigir la salida a un fichero llamado logColor.txt
- Desactivar el entorno.
- Justo después, repetir los mismos pasos pero con la versión de python 3.11 para poder ejecutar el fichero .py de plot_face_recognition (disponible también en el código de moodle). Tendrás que crear un nuevo fichero de requirements (crea un nuevo requirements llamado requirements4_2.txt y entrégalo también) para hacer funcionar dicho fichero en el entorno virtual de python3.11. **Si por algún casual no tienes la versión 3.11 de python instalada, puedes consultar cómo instalarla en este enlace. Tendrás que instalar venv en la versión de python 3.11 como has hecho antes con python3.12.**
- Ejecuta este apartado una vez que todo funcione bien, en el cluster (adjunta un pantallazo). Acuérdate de que una vez comprobado que funcione todo, eliminar los **entornos virtuales** del cluster. **En el cluster, para crear los entornos virtuales con venv, tendréis que llamar a python con python3.11 y python3.12.**

2.5 OPCIONAL: entornos virtuales

1. Crea un entorno virtual con la versión de python (3.12) y actívalo (es posible que necesites permisos de superusuario).
2. Tenéis dos scripts, en el resto de material de moodle, utils.py y model_training.py. Deberéis rellenar el segundo en los sitios indicados siguiendo los siguientes pasos y adjuntarlo en la entrega. El objetivo es entrenar un modelo que realice una regresión lineal (En este caso simple, buscaremos una función del tipo $y = \beta x$). La salida del script será un plot con la función encontrada que deberéis añadir en el pdf de entrega. Deberéis entregar también el fichero model_training.py que habéis rellenado. Indica también si has tenido que instalar algún paquete en particular.

- Importa la función *generate_ dataset* del fichero de utils.
- Genera un dataset de 200 muestras con dicha función.
- Divide el dataset en trainset (x_train, y_train), y testset (x_test, y_test) con proporción de 80% - 20%.
- Calcula el valor de y predicho por el modelo que acabas de entrenar.
- Plotea el resultado de la predicción para compararlo con el valor real, deberás entregar también este plot.
- Una vez que veas que todo funciona, crea un scrip de bash (el nombre del script de bash debe contener tu nombre y _opcional.sh). El script debe crear el entorno virtual, instalar las librerías, ejecutar el fichero python y obtener la figura.

2.6 OPCIONAL: AWK

(a) Indica el directorio actual en el que te encuentras y crea un directorio llamado "Practica3". Se pedirá en este caso trabajar con AWK, según el material que hay subido a moodle en manuales y uso del software.

(b) En esta práctica utilizaréis el fichero de 10000 Sales Records.csv (disponible en los ficheros opcionales de la P1).

(c) Utilizando **únicamente** awk, sumar el beneficio total (Total Profit) de las ventas de Italia.

(d) Utilizando **únicamente** awk, mostrar los países que tienen algún valor de Total Revenue menor de 1000$

(e) Utilizando **únicamente** awk, cuenta el número de unos (1) por cada una de las diez primeras filas del fichero. El output que deberá tener en cada linea dos columnas, una con el número de linea y otra con el numero de unos (como se muestra a continuación):

```
1 0
2 9
3 10
4 16
5 11
6 12
7 7
8 5
9 8
10 15
```