Creando un entorno virtual con `conda`

Lo primero de todo es que si tenemos instalado `conda`, el prompt nos indica en qué entorno estamos trabajando. Si no hemos tocado nada, debería ser algo parecido a esto:

```
(base) ➜  ~
```

Comprobemos que no existe ningún otro entorno:

```
conda env list
```

Lo cual indica que estamos en el entorno base de `conda`. Creemos ahora nuestro entorno virtual de trabajo con `create -n`.

```
conda create -n PWC_project
```

`conda` nos dice dónde está metiendo el entorno. En concreto, en:

```
  environment location: /opt/anaconda3/envs/PWC_project
```

Veamos cómo ahora tenemos dos entornos sobre los que escoger. El entorno activo se indica con un asterisco:

```
conda env list
```

Para pasar al nuevo entorno, lo hacemos con la instrucción `activate`:

```
conda activate PWC_project
```

Y el prompt debería cambiar a algo parecido a esto:

```
(PWC_project) ➜  ~
```

Comprobamos que efectivamente es el entorno activo:

```
conda env list
```

Veamos qué tiene instalado el entorno con `list`:

```
conda list
```

Este entorno está totalmente vacío. Vamos a instalar con `install` los mínimos paquetes necesarios para poder trabajar en él:

```
conda install -c conda-forge jupyterhub jupyterlab nodejs nb_conda_kernels notebook
```

Veamos qué hemos instalado con `list`:

```
conda list
```

Lancemos nuestro recién instalado `jupyter-notebook`, *que se ejecutará desde nuestro entorno virtual*:

```
jupyter notebook
```

Y nos ponemos a trabajar en nuestro proyecto, instalando las librerías que sean necesarias para desarrollarlo. Una vez ya hayamos instalado todo lo que necesitamos podemos exportar las dependencias instaladas a través del archivo `environment.yml`:

```
conda env export >> environment.yml
ls
```

Estructura del archivo "environment.yml"

```
nano environment.yml
```

Recomendamos poner una ruta relativa, para que cuando se comparte el entorno se instale siempre en el mismo sitio independientemente de la carpeta de trabajo. Luego veremos cómo hacer esto.

Vamos ahora a eliminar el entorno. Primero lo desactivamos con `deactivate`, para volver al base:

```
conda deactivate
```

Y ahora lo eliminamos con `remove`:

```
conda env remove -n PWC_project
```

Muy importante usar el comando `clean` para eliminar los cachés o los instaladores de paquetes. Sorprende mucho cuánto espacio se puede liberar:

```
conda clean --all
```

Comprobemos que efectivamente lo hemos eiminado:

```
conda env list
```

Y ahora MUCHA ATENCIÓN... Repliquémoslo!!!!

```
conda env create -f environment.yml
conda activate PWC_project
jupyter-notebook
```

Et voilá! Podemos replicar nuestro entorno exactamente igual en cualquier momento y en cualquier máquina. Esto abre una puerta a poder programar de un modo colaborativo y ordenado.

Obviamente no hay que instalar todo cada vez. Imaginemos que nuestro colaborador ha añadido un par de nuevas librerías al proyecto, y ha generado su environment.yml que acabamos de actualizar desde el repositorio remoto. Para instalar esas librerías unicamente en mi entorno lo que tengo que hacer es lo siguiente:

```
conda activate PWC_project
conda env update --file environment.yml --prune
```

Y esto instalará los nuevos cambios que se hayan producido sobre lo que ya tenía.

Ejercicio

- Crear un entorno virtual desde cero, instalar una serie de librerías y un código que las utilice y pasarle a un compañero el código y el archivo .yml para que lo replique.
- Con el entorno replicado, instalar alguna librería más, añadir alguna línea de código más que use esas librerías, generar el archivo .yml y devolvérselo al compañero para que actualice su proyecto con los nuevos cambios
- Comprobar que el código funciona y que todas las librerías están disponibles

> Obviamente esto de "pasar" al compañero lo haremos de una manera mucho más seria y eficiente con GitHub

Entornos con prefijo

https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#creating-an-environment-with-commands

`conda` instala los entornos virtuales en el directorio anaconda3/envs. Esto puede estar bien, pero en ocasiones es interesante poder instalarlos en otros sitios. Por ejemplo: si estamos escribiendo código colaborativo, todos necesitamos instalar el entorno en el mismo directorio, que es el especificado en el archivo environment.yml. Lo recomendable es tenerlo en el directorio del proyecto en el que estemos trabajando. Se puede hacer del siguiente modo:

```
conda create --prefix ./envs jupyterlab=3.2 matplotlib=3.5 numpy=1.21
```

Para activar los entornos creados con prefijo se hace lo mismo que en los creados por nombre:

```
conda activate ./envs
```

Las ventajas de esto son:

- Es fácil saber que el proyecto usa un entorno aislado puesto que está incluído en él como subdirectorio.
- Hace que el proyecto sea autocontenido
- Es más fácil compartirlo
- Podemos usar siempre el mismo nombre para los entornos

Pero tenemos que tener en cuenta dos cosas:

- `conda` ya no puede encontrar el entorno con la bandera --name. Hay que pasarselo en general la ruta completa al directorio precedido de la bandera --prefix
- El prompt se vuelve muy molesto, con cosas del tipo:
  `(/Users/USER_NAME/research/data-science/PROJECT_NAME/envs) $`
  Para evitar esto hay que modificar la entrada env_prompt del archivo .condarc.
  `$ conda config --set env_prompt '({name})'`

Ahora el prompt mostrará solo el nombre genérico del entorno, que es el nombre de la carpeta que lo contien.

Actualizando

```
conda env update --prefix ./envs  environment.yml --prune
```

Eliminándolo

```
conda env remove  --prefix ./envs
```

Para que se instale siempre como un subdirectorio del directorio actual, hay que modificar el archivo `environment.yml`

`.gitignore`

Tenemos que asegurarnos de incluir el directorio del entorno en el archivo `.gitignore`. En caso contrario podemos estar subiendo al control de versiones todas las librerías allí instaladas, con el correspondiente coste de tiempo y espacio.

Ejercicio

- Crear un entorno virtual con prefijo desde cero en un directorio que llamaremos Proyecto_01-
- Instalar en él las librerías básicas de python
- Generar el archivos `environment.yml`