EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: PRIMEROS PASOS.
- EXERCISE 2: VERIFICACIÓN E INSTALACIÓN DE PYTHON 3.
- EXERCISE 3: VERIFICACIÓN DE LA INSTALACIÓN DE PAQUETE PIP3 – PYTHON PACKAGE INDEX(PYPI).
- EXERCISE 4: INSTALACIÓN Y CREACIÓN DEL ENTORNO VIRTUAL (VIRTUALENVWRAPPER).
- EXERCISE 5: INSTALACIÓN DE DJANGO EN EL ENTORNO VIRTUAL.

EXERCISE 1: PRIMEROS PASOS.

En este ejercicio se configura, instala y prueba un entorno de desarrollo virtual de Django en Linux (Ubuntu). Su objetivo es plantear una guía paso a paso para la virtualización de un proyecto de desarrollo en Django.

REQUISITOS PREVIOS:

- Tener conocimiento de un terminal o línea de comando e instalación de paquetes de software en el sistema operativo, en este caso Linux Ubuntu.
- Tener previamente instalado la versión de Python 3.

PROCEDIMIENTOS DE LA PRÁCTICA

El entorno de desarrollo de Django es una instalación en el computador local que se utiliza para el desarrollo de aplicaciones, con la finalidad de probarlas antes de desplegarlas al entorno de producción.

Django proporciona herramientas, y también un conjunto de scripts de Python que permiten crear y desarrollar proyectos Django, basados con servidor web de desarrollo simple, el cual se usa para probar de forma local las aplicaciones web Django con un navegador web del computador.

Existe un conjunto de herramientas periféricas que forman parte del entorno de desarrollo, como lo es un editor de textos o IDE para editar código (VSC), una herramienta de gestión del control de fuentes como Git para gestionar con seguridad las diferentes versiones de tu código.

El framework Django se ejecuta por encima de Python, y puede utilizarse con Python 2 o con Python 3 (o ambos inclusive).

Para ello es importante tener en cuenta que:

- **Python 2:** es una versión tradicional del lenguaje, y que no va a tener más características nuevas pero que tiene disponible para los desarrolladores un enorme repositorio de bibliotecas de terceros de alta calidad (algunas de las cuales no están disponibles en Python 3).
- **Python 3**: es una actualización de Python 2 que, aunque similar, es más consistente y fácil de usar pues se encuentra en constante mejoramiento y evolución.

EXERCISE 2: VERIFICACIÓN E INSTALACIÓN DE PYTHON 3.

El framework Django tiene como requerimiento que se debe tener instalado Python en el sistema operativo. Si se tiene instalado Python 3, se necesita la Python Package Index — pip3 — que se usa para gestionar (instalar, actualizar y eliminar) los paquetes/bibliotecas Python utilizados por Django y otras aplicaciones en Python.

Para verificar si tenemos instalado Python procedemos a abrir una consola terminal en Linux, y escribimos:

```
1 $ python -V
2 Python 3.8.3
```

Observamos que tenemos instalado la versión de Python 3.8.3, y si no tenemos instalado Python, procedemos a instalar la última versión.

Primero, instalamos el requisito previo para agregar PPA personalizados:

```
1 $ sudo apt install software-properties-common -y
```

En segundo lugar, se agrega el deadsnakes/ppa a la lista de fuentes del administrador de paquetes de APT:

```
1 $ sudo add-apt-repository ppa:deadsnakes/ppa -y
```

Se actualiza el listado de paquetes disponibles:

```
$ sudo aptitude update
```

Instalación de Python:

```
$ sudo aptitude install python3
```

Verificación de la instalación:

```
$ python3 --version
Python 3.8.3
```

EXERCISE 3: VERIFICACIÓN DE LA INSTALACIÓN DE PAQUETE PIP3 – PYTHON PACKAGE INDEX(PYPI).

Ubuntu Linux incluye Python 3 por defecto. Sin embargo, la herramienta Python Package Index que se requiere para instalar paquetes de Python 3, por ejemplo, Django, no está disponible por defecto. Se instala de la siguiente manera en la consola de la terminal:

```
$ sudo aptitude install python3-pip
```

Para revisar que efectivamente se encuentra instalado, podemos verificar para listar los paquetes instalados:

```
$ pip3 list
```

EXERCISE 4: INSTALACIÓN Y CREACIÓN DEL ENTORNO VIRTUAL (VIRTUALENVWRAPPER).

Se utilizarán las bibliotecas virtualenvwrapper para Linux para crear los entornos virtuales (disponible para Mac OS y Windows), la cual emplea como base la herramienta virtualenv. Esta herramienta wrapper crea una interfaz consistente para la gestión de interfaces en todas las plataformas. Para instalar virtualenvwrapper se realizará por medio de pip3:

```
$ sudo pip3 install virtualenvwrapper
```

Luego, se debe agregar al archivo de inicio de la terminal o shell oculto **.bashrc** que se encuentra en el directorio home del usuario, las siguientes lineas al final:

```
$ nano .bashrc
```

```
##### Entornos Virtuales Python ##############
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
export PROJECT_HOME=$HOME/Dev-Python
source /usr/local/bin/virtualenvwrapper.sh
```

Las líneas anteriores permiten ajustar la localización de donde deberían estar alojados los entornos virtuales, la localización de los directorios de tus proyectos de desarrollo, y la localización del script instalado con este paquete.

Se recarga el archivo de inicio, o se reinicia la terminal:

```
$ source ~/.bashrc
```

Una vez que hayas instalado virtualenvwrapper, se procede a crear un nuevo entorno virtual con el comando **mkvirtualenv**.

```
$ mkvirtualenv django_env
```

Al finalizar, observamos que el nuevo entorno virtual se encuentra activo, y que el comienzo del prompt será el nombre del entorno entre paréntesis.

```
$ $ mkvirtualenv django_env
created virtual environment CPython3.8.3.final.0-64 in 924ms
creator CPython3Posix(dest=/home/luispc/.virtualenvs/django_env,
clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle,
wheel=bundle, via=copy,
app_data_dir=/home/luispc/.local/share/virtualenv)
```

```
added seed packages: pip==22.0.4, setuptools==62.1.0, wheel==0.37.1
activators
BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellAc
tivator,PythonActivator
virtualenvwrapper.user_scripts creating
/home/luispc/.virtualenvs/django_env/bin/predeactivate
virtualenvwrapper.user_scripts creating
/home/luispc/.virtualenvs/django_env/bin/postdeactivate
virtualenvwrapper.user_scripts creating
/home/luispc/.virtualenvs/django_env/bin/preactivate
virtualenvwrapper.user_scripts creating
/home/luispc/.virtualenvs/django_env/bin/postactivate
virtualenvwrapper.user_scripts creating
/home/luispc/.virtualenvs/django_env/bin/get_env_details
(django_env)(base) dev-django@FullStack-Dev:~$
```

Para el uso del entorno virtual tenemos los siguientes comandos que son los más usados:

`deactivate` — Salir del entorno virtual Python actual.

```
$ (django_env) (base) dev-django@FullStack-Dev:~$ deactivate
```

`workon` — Listar los entornos virtuales disponibles.

```
$ (django_env) (base) dev-django@FullStack-Dev:~$ workon
django_env
my_django_environment
```

`workon nombre_entorno_virtual` — Activar el entorno virtual Python especificado.

```

$ (django_env) (base) dev-django@FullStack-Dev:~$ workon django_env
(django_env) (base) dev-django@FullStack-Dev:~$
```

`rmvirtualenv nombre_entorno_virtual` — Borrar el entorno especificado.

```
$ (django_env) (base) dev-django@FullStack-Dev:~$ rmvirtualenv django_env
Removing django_env...
```

```
3
```

EXERCISE 5: INSTALACIÓN DE DJANGO EN EL ENTORNO VIRTUAL.

Iniciamos, y entramos al ambiente virtual django_env, ahí procedemos a instalar Django con pip3:

```
$ (base) dev-django@FullStack-Dev:~$ workon django_env
(django_env) (base) luispc@FullStack-Dev:~$ pip3 install django
```

Verificamos los paquetes instalados en el entorno virtual con **`pip list`**:

```
$ (django_env) (base) luispc@FullStack-Dev:~$ pip list
Package            Version
------------------ -------
asgiref            3.5.2
backports.zoneinfo 0.2.1
Django             4.0.5
pip                22.0.4
setuptools         62.1.0
sqlparse           0.4.2
wheel              0.37.1
```

Observamos que tenemos instalado Django en la versión 4.0.5.