


Práctica de Laboratorio 2b

Explorar las herramientas de desarrollo de Python

Objetivos:

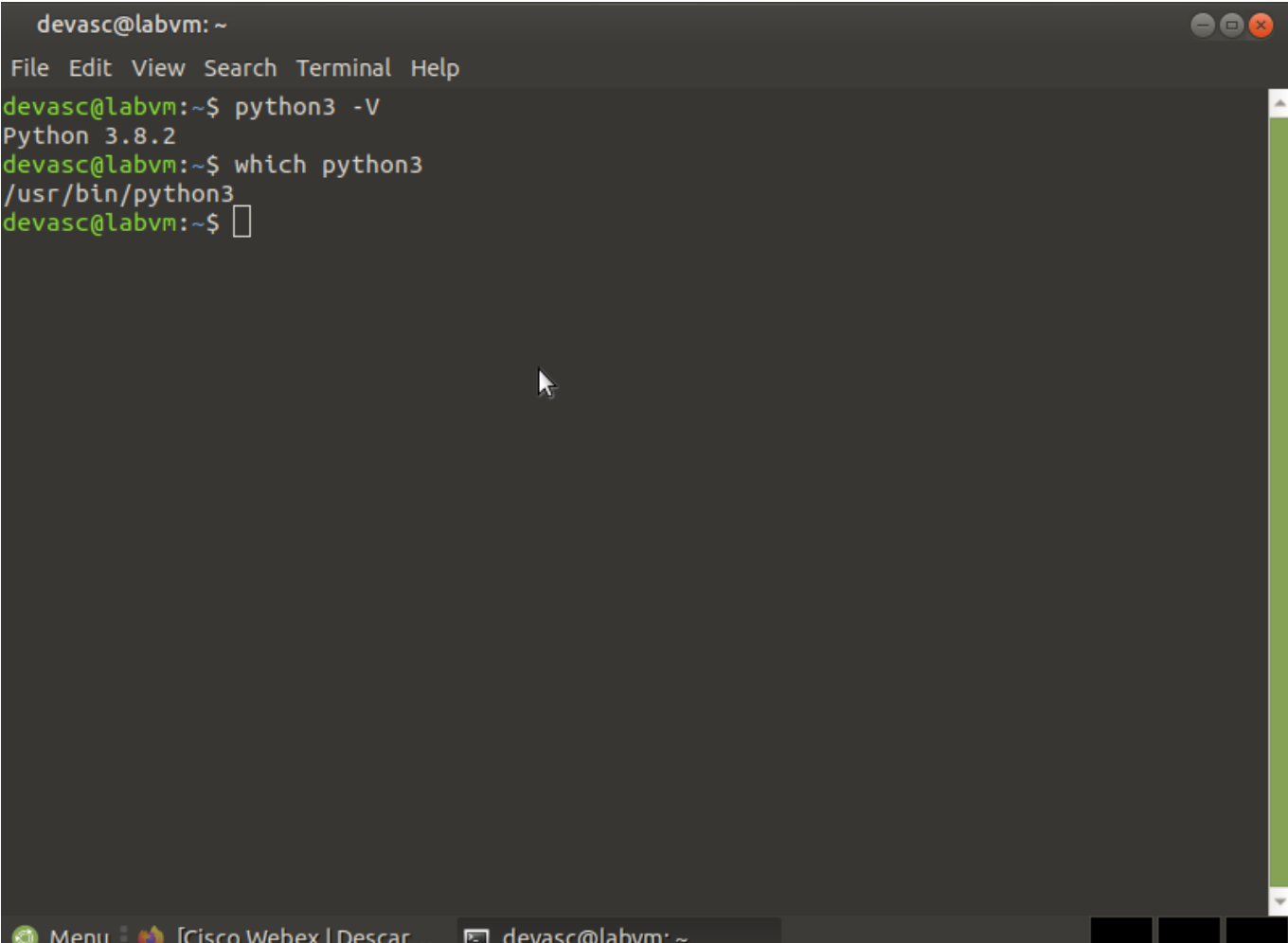
- Parte 1: Iniciar la Máquina Virtual (Virtual Machine) DEVASC.
- Parte 2: Revisar la instalación de Python.
- Parte 3: Entornos virtuales PIP y Python.
- Parte 4: Compartir su entorno virtual.

1. Iniciar la Máquina Virtual (Virtual Machine) DEVASC.

 alt text

2. Revisar la instalación de Python.

- Abra una terminal dentro de la VM DEVASC.
- Verifique la versión de Python instalada utilizando `python3 -V`.
- Compruebe la ubicación del ejecutable de Python con el comando `which python3`.



```
devasc@labvm: ~  
File Edit View Search Terminal Help  
devasc@labvm:~$ python3 -V  
Python 3.8.2  
devasc@labvm:~$ which python3  
/usr/bin/python3  
devasc@labvm:~$
```

3. Entornos virtuales PIP y Python.

- Navegue al directorio donde trabajará con Python en la VM.
- Cree un entorno virtual usando el módulo `venv` con `python3 -m venv devfun`.
- Active el entorno virtual e instale las dependencias necesarias utilizando `pip3`.
- Use `pip3 freeze` para listar las dependencias instaladas.

Pip3 freeze antes (vacío):

[illegible]

pip3 freeze después (con dependencias):

```
devasc@labvm: ~/labs/devnet-src/python
File Edit View Search Terminal Help
rsing-3.1.4 python-dateutil-2.9.0.post0 pytz-2024.1 requests-2.32.3 six-1.16.0 sniffio-
1.3.1 starlette-0.38.4 typing-extensions-4.12.2 tzdata-2024.1 urllib3-2.2.2 zipp-3.20.1
(devfun) devasc@labvm:~/labs/devnet-src/python$ pip freeze
annotated-types==0.7.0
anyio==4.4.0
certifi==2024.8.30
charset-normalizer==3.3.2
contourpy==1.1.1
cycler==0.12.1
exceptiongroup==1.2.2
fastapi==0.114.0
fonttools==4.53.1
idna==3.8
importlib-resources==6.4.4
kiwisolver==1.4.7
matplotlib==3.7.5
numpy==1.24.4
packaging==24.1
pandas==2.0.3
pillow==10.4.0
pydantic==2.9.0
pydantic-core==2.23.2
pyparsing==3.1.4
python-dateutil==2.9.0.post0
pytz==2024.1
requests==2.32.3
six==1.16.0
sniffio==1.3.1
starlette==0.38.4
typing-extensions==4.12.2
tzdata==2024.1
urllib3==2.2.2
zipp==3.20.1
```

Ojo: se instaló requests, numpy, pandas, matplotlib y FastAPI

Se comprueba la instalación de requests:  alt text

- Guardando las dependencias en un archivo requirements.txt:

```
(devfun) devasc@labvm:~/labs/devnet-src/python$ pip freeze > requirements.txt
(devfun) devasc@labvm:~/labs/devnet-src/python$ cat requirements.txt
annotated-types==0.7.0
anyio==4.4.0
certifi==2024.8.30
charset-normalizer==3.3.2
contourpy==1.1.1
cycler==0.12.1
exceptiongroup==1.2.2
```

- Instalando nuevo entorno con requirements.txt:

```
(devfun) devasc@labvm:~/labs/devnet-src/python$ deactivate
devasc@labvm:~/labs/devnet-src/python$ python3 -m venv devnew
devasc@labvm:~/labs/devnet-src/python$ source devnew/bin/activate
(devnew) devasc@labvm:~/labs/devnet-src/python$ pip3 install -r requirements.txt
Collecting annotated-types==0.7.0
  Using cached annotated_types-0.7.0-py3-none-any.whl (13 kB)
Collecting anyio==4.4.0
```

- Verificando requests en el nuevo entorno:

```
1.3.1 starlette-0.38.4 typing-extensions-4.12.2 tzdata-2024.1 urllib3-2.2.2 zipp-3.20.1
(devnew) devasc@labvm:~/labs/devnet-src/python$ pip freeze | grep requests
requests==2.32.3
(devnew) devasc@labvm:~/labs/devnet-src/python$
```

- Por último se desactivó el entorno virtual con `deactivate`.

Reflexiones

Levantar un entorno virtual, como hicimos para Python en este laboratorio, es algo que todos deberíamos adoptar rutinariamente. Esto evita una cantidad de problemas comunes, como conflictos entre dependencias y problemas de compatibilidad, que pueden hacer que un proyecto de desarrollo se detenga.

He descubierto que el manejo de entornos virtuales no es sólo una habilidad técnica esencial, sino una forma de asegurar que nuestros proyectos sean reproducibles y estables en cualquier máquina. Esto es crucial para trabajar en equipo y para mantener la sanidad del código a largo plazo.

André Pacheco