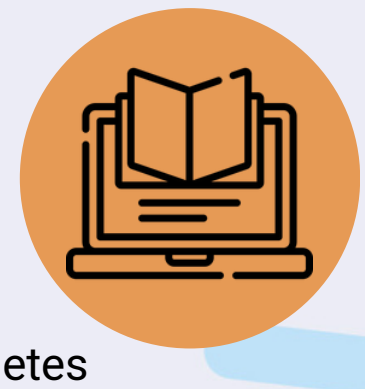
Entornos virtuales

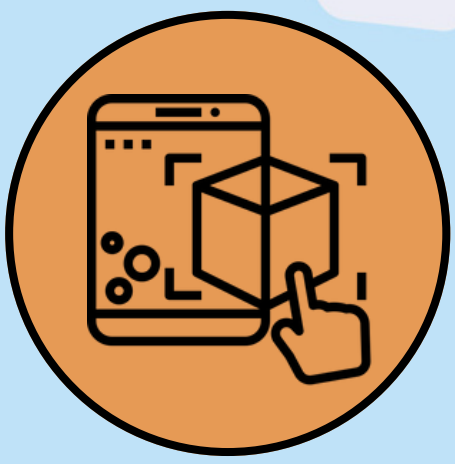
Según la documentación oficial de Python, los entornos virtuales se gestionan con el módulo venv:
“El módulo venv proporciona soporte para crear «entornos virtuales» ligeros con sus propios directorios de ubicación, aislados opcionalmente de los directorios de ubicación del sistema. Cada entorno virtual tiene su propio binario Python (que coincide con la versión del binario que se utilizó para crear este entorno) y puede tener su propio conjunto independiente de paquetes Python instalados en sus directorios de ubicación”.



¿Cuál es la ventaja de tener entornos virtuales?

Imagine tener múltiples proyectos, cada uno con sus propias dependencias, paquetes y versiones específicas de bibliotecas. Sin entornos virtuales, estos proyectos podrían interferir entre sí, resultando en conflictos y problemas de compatibilidad. El propósito principal de los entornos virtuales es evitar este tipo de conflictos. Al crear un entorno virtual para cada proyecto, los desarrolladores pueden definir y gestionar sus propias dependencias sin afectar el sistema global de Python. Esto asegura que cada proyecto tenga acceso a las versiones específicas de las bibliotecas que necesita, de forma aislada a los demás entornos. Es una buena práctica de desarrollo y evita que, al configurar bibliotecas, el desarrollador pierda el tiempo configurando versiones.

Además de evitar conflictos, los entornos virtuales también facilitan la colaboración. Cuando un equipo de desarrolladores trabaja en un proyecto, cada miembro puede tener su propio entorno virtual, garantizando consistencia en las dependencias utilizadas. Esto ayuda a mitigar problemas potenciales cuando se comparten y colaboran en el código, eliminando problemas como: “este programa solo funciona en mi computador”.



Otra ventaja de los entornos virtuales...

Es que puedes compartir fácilmente un archivo de requisitos (llamado requirements.txt) que enumera todas las dependencias y sus versiones exactas. Al replicar el entorno virtual en otro sistema o máquina, otro desarrollador puede recrear exactamente el mismo entorno de desarrollo, garantizando una experiencia consistente sin importar que se ejecute un programa en un computador, en un servidor o en equipos con sistemas operativos diferentes. Es decir, si se quiere replicar el ambiente de trabajo de un programa, basta con tener el archivo de las instrucciones del programa (script) y un archivo requirements.txt y se podrá replicar en cualquier lugar y ejecutar sin problemas.

Si bien venv es un gestor de entornos virtuales que viene con Python, no es el único. De hecho, existen gestores con interfaz gráfica que facilitan la creación y la gestión de entornos, e incluso el manejo de paquetes que tienen muchas dependencias y su instalación es compleja, haciendo fácil el manejo de dependencias. Dentro de estos gestores existe:

- Conda.
- Poetry.
- virtualenv

Existen soluciones de gestión integradas, como es el caso de Anaconda, que es un paquete basado en el gestor de entornos virtuales conda, y también permite gestionar desde una interfaz gráfica de los paquetes, crear entornos e incluso instalar librerías para otros lenguajes de programación como R.

Cuando se instala anaconda, incluye otros programas como jupyterlab que permiten crear cuadernos interactivos de Python, spyder que es un entorno (IDE) de desarrollo, pycharm que es otro IDE de desarrollo, entre otras herramientas que también se pueden emplear para este curso.

Consola de Linux

La consola de Linux, también conocida como terminal o shell, es una interfaz de línea de comandos que permite a los usuarios interactuar con el sistema operativo mediante comandos de texto.
La razón principal para utilizar la consola de Linux es la eficiencia y la potencia que ofrece en la ejecución de tareas específicas. Con simples comandos, los usuarios pueden realizar acciones complejas y automatizar tareas, lo que mejora la productividad y reduce la dependencia de interfaces gráficas. La consola también es esencial en entornos de servidores y sistemas sin interfaz gráfica, donde no hay monitores ni controladores de gráficos consumiendo memoria, por lo que dominarla es esencial en la ciencia de datos.

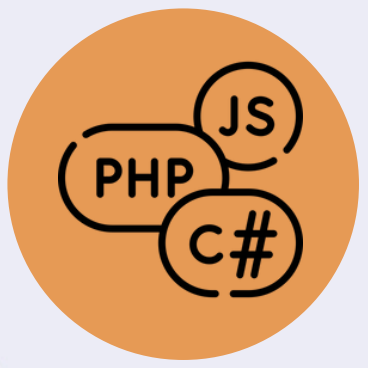
La consola de Linux, ejecuta comandos a través de un programa que interpreta los comandos (un intérprete, similar a como funciona Python) y tiene una sintaxis de escritura y estructuras de control como los lenguajes de programación. En consecuencia, se pueden escribir programas conocidos como Bash o Shell scripts que son archivos de texto que contienen secuencias de comandos de la consola de Linux.

La utilidad de los bash scripts es que se pueden automatizar tareas y comandos haciendo simples operaciones complejas como son la manipulación de archivos, el procesamiento de texto, la ejecución de programas, entre otros. La flexibilidad de los scripts de shell los hace útiles para la automatización de procesos, la gestión de tareas repetitivas y la creación de flujos de trabajo personalizados.

Algunos comandos para conocer la terminal

Los comandos de Linux se ejecutan en el Terminal pulsando Enter al final de la línea. Puedes ejecutar comandos para realizar diversas tareas, desde la instalación de paquetes hasta la gestión de usuarios y la manipulación de archivos.

La sintaxis general de un comando Linux es la siguiente:



Nombredelcomando [opcion(es)] [parametro(s)]

Los comandos Linux pueden contener una opción o un parámetro. En algunos casos, pueden ejecutarse sin ellos. Estas son las tres partes más comunes de un comando:

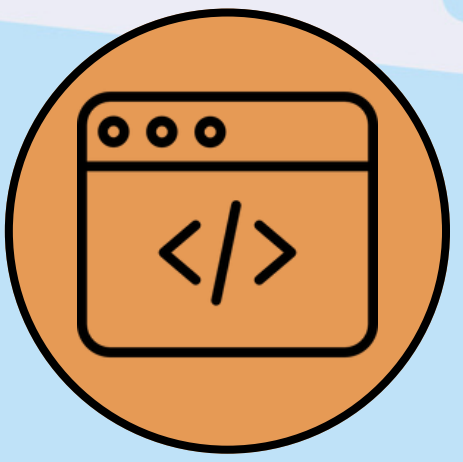
- CommandName es la regla que deseas ejecutar.
- Option o flag modifica el funcionamiento de un comando. Para ejecutarla, utiliza guiones (-) o guiones dobles (—).
- Parameter o argument especifica cualquier información necesaria para el comando.

Todos los comandos de Linux distinguen entre mayúsculas y minúsculas.

Lista de comandos básicos, para practicar a medida que se explican con los estudiantes

1. Comando sudo: es la abreviatura de super usuario, permite realizar tareas con permisos de root (máxima autoridad en el computador), todo comando que sea antecedido por sudo se ejecutará con privilegios elevados.
2. Pwd: permite conocer cuál es la ruta de trabajo actual (carpeta en la que nos encontramos).
3. cd es el comando para navegar entre carpetas desde la terminal, permite movernos a diferentes directorios, por ejemplo cd /home, o cd /usr , al ejecutarlos veremos cómo la ruta en la consola cambia. Para subir un directorio podemos emplear el comando cd.
4. ls permite listar los archivos y carpetas dentro de un directorio, se le pueden agregar opciones, por ejemplo ls -l permite ver en forma de lista o ls -la los publica en forma de lista y muestra archivos ocultos. En Linux los archivos ocultos son aquellos cuyo nombre empieza con un punto. Para listar directorios con subdirectorios y contenidos se puede emplear ls -R (recursivo).
5. cat este comando permite imprimir en la salida estándar (consola) el contenido de los archivos. Es útil para visualizar rápidamente archivos y configuraciones.

6. cp: permite copiar un archivo en otro, por ejemplo: cp nombrearchivo1.txt nombrearchivo2.txt nombrearchivo3.txt /inicio/nombredeusuario/Documentos
7. mv: permite mover archivos de un lugar a otro.
8. mkdir: crea directorios, por ejemplo mkdir micarpeta crearía una carpeta llamada micarpeta.
9. Touch: permite crear archivos vacíos, por ejemplo touch holamundo.py, si inmediatamente después se usa el comando ls se puede apreciar el archivo holamundo.py creado.
10. Top: permite ver los procesos en ecución con su consumo de recursos, para salir se debe pulsar escape o q.



En Linux es común que Python venga pre instalado, como actividad se sugiere que los estudiantes creen un archivo de “hola mundo” y lo ejecuten empleando la consola de Linux: python3 hola_mundo.py, para editar el contenido del fichero puede emplear vim en consola. Si vim no se encuentra instalado, oriente a los estudiantes a la instalación utilizando el comando: sudo apt install vim.