**Temas de etapa 2**

**PEP 20 - El zen de python:**

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than \*right\* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

Cuando un código es explícito, no requiere que el lector tenga que intuir o saber de antemano algún elemento implícito del sistema, haciendo que la legibilidad mejore considerablemente.

Si el sistema está correctamente implementado, cada parte independiente será simple si se estudia de forma aislada, por lo que, el estudio del sistema completo se podría simplificar en el estudio de cada parte simple.

**Simple > Complejo > Complicado**

Cuando se aumenta la densidad del código haciendo en pocas líneas muchas operaciones se puede hacer perder el foco del lector, por tanto es muy recomendable añadir espacios entre bloques lógicos.

**El código se escribe una vez pero se leen cientos de veces**, por lo tanto es muy importante prestar **especial atención a mejorar la legibilidad** lo más posible

Los casos especiales no son lo suficientemente especiales para romper reglas

Si hay un error debe ser visible para poder ser arreglado.

Si la forma de implementar el problema no parece obvia, quizás no sea la más correcta y haya que seguir pensando en otra opción.

Si la implementación es difícil de explicar, es una mala idea

**ENTORNOS VIRTUALES**

Los entornos virtuales son una forma de crear un sistema operativo virtual dentro de otro sistema operativo.

**Verificar donde está python y pip**

which python3

which pip3

**Si estas en linux o wsl debes instalar**

sudo apt install -y python3-venv

**Poner cada proyecto en su propio ambiente, entrar en cada carpeta.**

python3 -m venv env

**Activar el ambiente**

source env/bin/activate

**Salir del ambiente virtual**

deactivate

**Podemos instalar las librerias necesarias en el ambiente virtual como por ejemplo**

pip3 install matplotlib==3.5.0

**Verificar las instalaciones**

pip3 freeze

**REQUIREMENTS**

**Generar el archivo con el siguiente comando**

pip3 freeze > requirements.txt

**Revisar lo que hay dentro del archivo**

cat requirements.txt

**Instalar las dependencias necesarias para contribuir más rápido en proyectos**

pip3 install -r requirements.txt

**PIP**

PIP es el gestor de paquetes de python, es posible buscar librerías en la página pypi.org.

**Ver la versión de pip**

pip3 -V

**Instalación de paquetes**

pip3 install <libreria>.

**Listar las librerías que se tienen en el entorno de python global**

pip3 list.

**Listar todas las librerías de python instaladas por el usuario**

pip3 freeze.

**DOCKER**

**Dockerfile**

FROM python:3.8

WORKDIR /app

COPY requirementes.txt /app/requirementes.txt

RUN pip install --no-cache-dir --upgrade -r /app/requirements.txt

COPY . /app

CMD bash -c "while true; do sleep 1; done"

**docker-compose.yml**

services:

app-csv:

build:

context: .

dockerfile: Dockerfile

**Configuración del yml para automatizar la vinculación de archivos**

services:

app-csv:

build:

context: .

dockerfile: Dockerfile

volumes:

- .:/app

**dockerizando web services**

**dockerfile**

FROM python:3.10

WORKDIR /app

COPY requirements.txt /app/requirements.txt

RUN pip install --no-cache-dir --upgrade -r /app/requirements.txt

COPY . /app

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]

**docker-compose.yml**

services:

web-server:

build:

context: .

dockerfile: Dockerfile

volumes:

- .:/app

ports:

- '8080:80'

Para correr el docker-compose a través de la terminal de comandos:

**docker-compose build**:

Este comando se utiliza para construir las imágenes de los servicios especificados en el archivo “docker-compose.yml”. Esto significa que tomará las especificaciones de construcción de cada servicio, como el Dockerfile y los contextos, y creará una imagen nueva y actualizada.

**docker-compose up -d**:

Este comando se utiliza para iniciar los servicios especificados en el archivo “docker-compose.yml” en segundo plano. El parámetro “-d” indica que se ejecutará en segundo plano.

**docker-compose ps:**

Este comando se utiliza para ver el estado de los servicios especificados en el archivo “docker-compose.yml”. Muestra si los servicios están corriendo o detenidos, y también muestra el nombre del contenedor, el puerto y la imagen utilizada.

**docker-compose exec app-csv bash:**

Este comando se utiliza para ejecutar una terminal dentro del contenedor especificado, en este caso “app-csv”, y conectarse a él mediante una sesión interactiva de bash. Esto permite realizar tareas dentro del contenedor, como verificar los archivos o ejecutar comandos.

**docker-compose down:**

Este comando se utiliza para detener y eliminar los contenedores, redes y volúmenes creados por “docker-compose up”.

| **FUNCIÓN** | **DEFINICIÓN Y SINTAXIS** | **EJEMPLO** |
| --- | --- | --- |
| **MAP** | El uso de map aplica una determinada función a todos los elementos de una entrada o lista. Esta es su forma:  map(funcion\_a\_aplicar,lista\_entradas) | lista = [1, 2, 3, 4, 5]  al\_cuadrado = list(map(**lambda** x: x\*\*2, lista)) |
| **FILTER** | filter crea una lista de elementos si usados en la llamada a una función devuelven True. Es decir, filtra los elementos de una lista usando un determinado criterio. | lista = range(-5, 5)  menor\_cero = list(filter(**lambda** x: x < 0, lista))  print(menor\_cero)  *# Salida: [-5, -4, -3, -2, -1]* |
| **REDUCE** | reduce es muy útil cuando queremos realizar ciertas operaciones sobre una lista y devolver su resultado. | **from** **functools** **import** reduce  producto = reduce((**lambda** x, y: x \* y), [1, 2, 3, 4])  *# Salida: 24* |

**COMANDOS DE TERMINAL**

**¿Qué es un comando?**

1. Un programa ejecutable

2. Un comando de utilidad de la shell

3. Una función de shell

4. Un alias

type => permite conocer de qué tipo es cada comando

help comando => despliega un menú con ayudas para conocer más sobre las funcionalidades del comando

comando --help => despliega un menú con ayudas para conocer más sobre las funcionalidades del comando

man comando => despliega el manual de usuario de un comando

info comando => despliega el manual de un comando

whatis comando => descripción breve del comando

**Alias**

---Cómo crear alias---

Atención => los alias no duran para siempre, se eliminan al reiniciar la terminal. Para guardarlos deben ser incluidos en el archivo de configuración de la terminal

alias l="ls -lh"

.bashrc => aquí se pueden crear alias

**COMANDOS DE LISTADO**

**ls** => listar elementos del directorio

**ls -l** => listar elementos con diferentes características

**ls -lh** => listar elementos con diferentes características en vista humana

**ls -al** => listar elementos del directorio incluidos elementos ocultos

**ls -lS** => lista elementos del directorio por tamaño de archivo

**ls -lSh** => lista elementos del directorio por tamaño de archivo en vista humana

**ls -r** => lista elementos del directorio en orden inverso

**tree** => despliegaa todos los directorios como si fueran un arbol

**tree -L 2(nivel de profundidad)** => despliegaa todos los directorios como si fueran un arbol ed acuerdo al nivel de profundidad indicada

COMANDOS DE DESPLAZAMIENTO

**cd nombre\_de\_dir** => Cambiar de directorio

**pwd** => Muestra ruta actual

**cd ./** => referencia el directorio actual

**cd ..** => regresar un directorio

COMANDOS DE LIMPIEZA Y RESETEO

**clear** => limpia la vista de la terminal

**cont + l** => limpia la vista de la terminal

**reset** => resetea la terminal

**cont + c** => Mata "kill" procesos activos en la terminal

**ARCHIVOS**

**mkdir nomber\_de\_archivo** => Crea un nuevo directorio con el nombre indicado

**mkdir dir1 dir2 dir3** => Crea múltiples directorios con los nombres indicados

**touch nomber\_de\_archivo** => crea un nuevo archivo del tipo indicado (.txt, .py, .js, etc)

**touch file1 file2 file3** => crea múltiples archivos con los nombres y tipos indicados

**cp file1 copy\_file1** => copia un nuevo archivo con el nombre indicado, también se puede establecer la ruta a copiar. También se pueden renombrar archivos

**mv file1 .. (o dirección de destino)** => mover archivo a nuevo directorio. También sirve para mover directorios enteros

**rm file1** => elimina el archivo

**rm -i file1** => pregunta antes de eliminar el archivo

**rm -r dir1** => elimina directorios. También se pueden eliminar múltiples directorios al mismo tiempo

**rm -rf** => elimina de forma forzada el directorio

**file nomber\_de\_archivo** => se utiliza para determinar el tipo de archivo

**LECTURA DE ARCHIVOS DE TEXTO**

**head file.tx**t=> muestra la cabecera (primeras 10 líneas) del archivo de texto

**head file.txt -n 15** => muestra las primeras n líneas del archivo de texto según sea indicado

**tail file.txt** => muestra las últimas 10 líneas del archivo de texto

**tail file.txt -n 15**=> muestra las últimas n líneas del archivo de texto según sea indicado

**less** => permite visualizar y navegar por archivos de texto largos de manera interactiva. Al utilizar less, puedes desplazarte hacia arriba y hacia abajo por el contenido del archivo, buscar texto, saltar a líneas específicas y realizar otras acciones útiles.

al interior de less se puede:

**/palabra** => realizar búsquedas

**q** => salir de la interfaz

**xdg-open file.txt** => abre el archivo de texto con el editor de texto predeterminado. Usualmente VScode

**WILDCARDS**

Las wildcards (comodines) son caracteres especiales que se utilizan en la línea de comandos para representar conjuntos de caracteres o nombres de archivos de forma concisa y flexible.

**ls \*.txt** => muestra todas las coincidencias de archivos que terminen con .txt o la terminación indicada

**ls file\*** => Coincide con cualquier archivo que comience con "file"

**ls file?** => busca todas las palabras que tengan "file" al inicio y solo un caracter al final. si se quiere especificar la cantidad de caracteres al final se pueden agregar más signos de interrogación

**ls [[:upper:]]\*** => se utiliza para listar los archivos y directorios en el directorio actual cuyos nombres comienzan con una letra mayúscula, seguidos de cualquier combinación de caracteres adicionales.

**ls -d [[:upper:]]\*** => se utiliza para listar los nombres de los directorios en el directorio actual que comienzan con una letra mayúscula, en lugar de listar los contenidos de los directorios. Al agregar la opción -d al comando ls, se le indica que liste solo los nombres de los directorios en lugar de mostrar los contenidos de los mismos.

**ls -d [[:lower:]]\***

**ls []** => no permite buscar por clases

**ls [ad]\*** => se utiliza para listar los archivos y directorios en el directorio actual cuyos nombres comienzan con "a" o "d", seguidos de cualquier combinación de caracteres adicionales.

**ls [:alnum:]**

**[:digit:]**

**REDIRECCIONES**

se utilizan para redirigir la salida de un comando hacia un archivo en lugar de mostrarla en la pantalla. La redirección > permite crear o sobrescribir un archivo con la salida del comando especificado.

**comando > archivo.txt**

La redirección >> agrega la salida al final del archivo en lugar de sobrescribirlo.

**comando >> archivo.txt**

**ls dirconerror 2> error.txt** => La redirección 2> se utiliza para redirigir el flujo de error estándar, que normalmente se muestra en la pantalla, hacia un archivo específico. El número "2" representa el descriptor de archivo estándar para el flujo de error.

**ls directorio > output.txt 2>&1 => ls dirconerror:** Intenta listar el contenido del directorio llamado "directorio". > output.txt: Redirige el "flujo de salida estándar" (stdout) generado por el comando anterior y lo guarda en el archivo "output.txt". 2>&1: Redirige el "flujo de error estándar" (stderr) generado por el comando anterior hacia el mismo destino que el flujo de salida estándar.

**PIPE OPERATOR**

**echo** "hola futuro programador" => se utiliza para imprimir texto en la salida estándar. Puede ser utilizado para mostrar mensajes, variables o cualquier contenido de texto en la pantalla.

**cat file1 file2** => se utiliza para concatenar archivos y mostrar su contenido en la salida estándar (normalmente la pantalla).

Además de simplemente mostrar el contenido de los archivos, el comando cat también puede ser utilizado para combinar y redirigir el contenido de los archivos hacia otros comandos o archivos.

**cat archivo.txt > nuevo\_archivo.txt**

**cat archivo1.txt archivo2.txt > resultado.txt**

Este comando concatenará el contenido de "archivo1.txt" y "archivo2.txt" y lo guardará en "resultado.txt".

El pipe operator **(|)** es un símbolo utilizado en la línea de comandos de Unix, como en Ubuntu, para conectar la salida de un comando con la entrada de otro comando. Permite enviar el resultado de un comando como entrada directa a otro comando, creando una cadena de comandos que trabajan juntos para realizar tareas más complejas.

**ls | grep "archivo"** => En este caso, el comando ls lista los archivos y directorios en el directorio actual, y luego el resultado se redirige como entrada al comando grep. El comando grep realiza una búsqueda de la palabra "archivo" en la entrada recibida, mostrando solo las líneas que coinciden con el patrón.

**less | tee output.txt** => less: Inicia el visor de archivos de texto "less". tee output.txt: El comando tee lee la entrada que recibe y la muestra en la salida estándar (pantalla), pero también la redirige y guarda en un archivo especificado. En este caso, la salida de less se muestra en la pantalla y se guarda en el archivo "output.txt".

**OPERADORES DE CONTROL**

**ls; mkdir file1; cal** => ";" se utiliza como un separador de comandos en una línea. Indica al intérprete de comandos que se deben ejecutar los comandos de manera secuencial, uno después del otro, en el mismo orden en el que se especifican.

**ls & mkdir file1 & cal** => El símbolo "&" se utiliza para ejecutar los comandos en segundo plano o en paralelo. Cuando se utiliza el símbolo "&" después de un comando, indica al intérprete de comandos que debe ejecutar ese comando en segundo plano y continuar con la ejecución de los comandos siguientes sin esperar a que el comando en segundo plano termine.

**mkdir dir1 && cd dir1** => El operador "&&" se utiliza para ejecutar los comandos en secuencia y de manera condicional. Cuando se utiliza "&&", el segundo comando se ejecuta solo si el primer comando tiene éxito.

**mkdir dir1 || cd dir1 || touch file1** => El operador "||" se utiliza para ejecutar comandos de manera condicional, pero en este caso se ejecutan en orden hasta que uno de ellos tenga éxito.

**CÓMO SE MANEJAN PERMISOS**

**chmod 755 file1** => establece los permisos de acceso para el archivo "file1" utilizando el sistema de permisos numéricos en Linux. usuario - gurpo de usuarios - all

**chmod u-r file1** => quita permiso de lectura a usuario

**chmod u+r file1 =**> se le concede permiso de lectura al usuario

**chmod u-x,go=w** => se modifican permisos a usuario, grupo y otros

**sudo su** => cambiar al usuario raiz

**passwd** => cambia la contraseña del actual usuario

**CÓMO CONFIGURAR VARIABLES DE ENTORNO**

**printenv** => muestra todas las variables de entorno configuradas

**echo $HOME =**> Muestra la ruta home del usuario

**echo $PATH =>** Muestra la ruta de todos los binarios que ejecuta el sistema

**COMANDOS DE BÚSQUEDA**

**which code** => El comando "which" se utiliza en sistemas basados en Unix, como Linux, para determinar la ubicación de un ejecutable específico en el sistema.

**find .~/ -name file** => busca en el directorio actual y sus subdirectorios todos los archivos que se llamen "file" y muestra sus ubicaciones.

**find .~/ -name \*.txt**

**find .~/ -name \*.txt | less**

**find .~/ -type df -name file1**

**find ./ -type f \*.log** => encontrar todos los archivos de log

**findos ./ -size 20M** => busqueda por tamaño de archivo

**grep palabra file.csv** => "grep" se utiliza para buscar líneas dentro del archivo "file.csv" que contengan la palabra especificada, en este caso, "palabra".

**grepi -i the file.csv** => El comando "grep -i" se utiliza para buscar líneas dentro de un archivo o una entrada de texto sin distinguir entre mayúsculas y minúsculas

**grp -c palabra file.csv** => El comando "grep -c" se utiliza para contar el número de ocurrencias de un patrón de búsqueda en un archivo o una entrada de texto.

**grep -v palabra file.csv** => "grep -v" se utiliza para buscar líneas dentro de un archivo o una entrada de texto que no contengan el patrón de búsqueda especificado.

**wc file.csv =>** cuenta el número de palabras, líneas y bits en el archivo

**wc -l file.csv** => número de líneas en el documento

**wc -w file.csv** => número de palabras

**wc -c file.csv** => número de bits

**UTILIDADES DE RED**

**ifconfig** => lista de las interfaces de red disponibles en tu sistema, como Ethernet, Wi-Fi, y Loopback.

**ping www.pagina.com** => verificar la conectividad entre un dispositivo local y un dispositivo remoto a través de la red.

**curl www.pagina.com** => Al ejecutar el comando "curl" seguido de una URL, el comando realiza una solicitud a esa URL y muestra la respuesta en la terminal.

**wget www.pagina.com** => Al ejecutar el comando "wget" seguido de una URL, el comando inicia una solicitud de descarga del archivo especificado en la URL y guarda ese archivo en el directorio actual (o en el directorio especificado mediante opciones adicionales).

**traceroute www.pagina.com** => El comando "traceroute" se utiliza para rastrear la ruta que sigue un paquete a través de una red desde tu dispositivo hacia un destino especificado.

**netstat -i** => se muestra una tabla con información sobre las interfaces de red disponibles en el sistema.

**COMPRIMIR ARCHIVOS TAR Y ZIP**

**tar -cvzf tocompres.tar.gz tocompress** => "tar": Es el programa utilizado para manipular archivos tar. "-c": Es una opción que indica a tar que cree un nuevo archivo. "-v": Es una opción que activa el modo verboso (verbose), lo que significa que se **mostrará información detallada sobre los archivos** que están siendo procesados.

**tar -xvzf tocompres.tar.gz** => descomprime y extrae los archivos de un archivo tar.gz

**zip -r tocompress.zip tocompres** => se utiliza para comprimir y empaquetar un directorio o varios archivos en un archivo ZIP

**unzip tocompress.zip** => ", se extraen todos los archivos y carpetas que se encuentran dentro del archivo ZIP

**MANEJO DE PROCESOS**

**ps** => mostrar información sobre los procesos en ejecución en el sistema

**kill pid(códido)** => enviar una señal a un proceso en ejecución, solicitando que se detenga o realice una acción específica.

**top** => mostrar una vista en tiempo real de los procesos en ejecución y el uso de recursos del sistema.

**jobs** => mostrar los trabajos (procesos) que están en ejecución en segundo plano en la sesión actual del shell.

**fg** => traer un trabajo (proceso) en segundo plano al primer plano

**EDITORES DE TEXTO EN LA TERMINAL**

**vim** => es un editor de texto altamente configurable y potente que se utiliza en sistemas Unix-like, como Linux.

con :q se sale de vim

con "i" se puede escribir directamente

:wq!