Tiempo de lectura
17 min

# Grabación de la Clase 9


# Principales Objetivos de Aprendizaje para esta Clase

- **Comprender cómo realizar la vinculación con Datos Externos**
- **Comprender cómo interactuar con el sistema de archivos**

# Vinculación con Datos Externos

Es muy importante tener en cuenta que va a ser necesario interactuar con el usuario y trabajar con datos que están alojados en medios externos, puede tratarse de un sistema de archivos ó de una tabla en una base datos, entre otras fuentes.
Es posible realizar operaciones de interacción con el usuario y manipular archivos usando métodos para leer y escribir desde y hacia el sistema de archivos.

## Entrada / Salida

Se utilizan funciones ya integradas en el intérprete del lenguaje. Para solicitar información al usuario se utiliza **input**:

```
>>> color = input('¿Cuál es tu color favorito?:
¿Cuál es tu color favorito?: Azul
>>> print(color)
Azul
>>> nro = input('¿Cuál es el doble de 3?: ')
```

**Dejanos tu feedback!** 👍

```
>>> else:
>>>   print('Respuesta Incorrecta')
```

## Contenido de la clase



Tips: Notar que input() devuelve un tipo de dato string, motivo por el cual al preguntar la igualdad con el 6, lo convertimos a entero

La **salida estándar** es la forma general de mostrar información por pantalla es mediante una consola de comando, generalmente podemos mostrar texto y variables separándolos con comas, para este se usa la sentencia **print**

```
>>> print("el resultado de raíz cuadrada de dos
el resultado de raíz cuadrada de dos es: 1.414
```

Una forma esencial de escribir un programa es a través de archivos llamados scripts, que no son más que lotes de instrucciones. Luego se envía este archivo al intérprete como parámetro desde la terminal de comando (si es un lenguaje interpretado como Python) y éste ejecutará todas las instrucciones en bloque.
Una característica interesante de los scripts es que pueden recibir datos desde la propia terminal de comando en el momento de la ejecución, algo muy útil para agregar dinamismo a los scripts a través de parámetros o argumentos personalizables.
Para poder enviar información a un script y manejarla, tenemos que utilizar la librería de sistema **sys**. En ella encontraremos la lista **argv** que almacena los argumentos enviados al script.
Suponiendo que creamos un script llamado "ejemplo_parametros.py" que tenga el siguiente código:

```
import sys
# Comprobación de seguridad, ejecutar sólo si s
# argumentos realemente
if len(sys.argv) == 3:
    texto = sys.argv[1]
    repeticiones = int(sys.argv[2])
    for r in range(repeticiones):
        print(texto)
else:
    print("ERROR: Introdujo uno (1) o más de d
    print("SOLUCIÓN: Introduce los argumentos
    print('Ejemplo: ejemplo_parametros.py "Text
print("El argumento 0 es el nombre del archivo
```

Luego en la consola, realizar la ejecución del script, por ejemplo:

python ejemplo_parametros.py 'Hola' 4

Tips: Notar que la lista de argumentos argv contiene en su elemento 0 el nombre del archivo.

**Dejanos tu feedback!** 👍

archivos de texto o también llamados secuenciales. Para esto Python incorpora un tipo integrado llamado **file**. Es importante notar que estos archivos siempre devuelven los datos como cadenas de caracteres, por lo tanto, de ser necesario, otros tipos deben ser convertidos.

- **Crear un archivo a partir de datos presentes en una lista:**

```
>>> import os
>>> lineas_archivo = ['Esto es un archivo de e
>>> archivo = open('datos.txt', 'w')
>>> for linea in lineas_archivo:
>>>     archivo.write(linea+'\n')
>>> archivo.close()
```

Es importante cerrar el archivo, ya que si no lo hacemos, el sistema operativo interpreta que está en uso, tal como si estuviera abierto en cualquier editor de texto.

- **Abrir un archivo, e iterar en él para leerlo:**

```
>>> import os
>>> archivo = open('datos.txt', 'r')
>>> for linea in archivo:
>>>     print linea
>>> archivo.close()
Esto es un archivo de ejemplo
Segunda linea
Tercera linea
```

Tips: Notar que la función open tiene como primer parámetro el nombre del archivo y como segundo parámetro un carácter que indica el modo de apertura:
Con 'x' creará el archivo, sin embargo retorna error si ya existe
Con 'a' agregá contenido si el archivo existe, si no lo crea
Con 'w' crea el archivo directamente, si existe y tiene contenido lo pisa

## Interactuar con el sistema de archivos

El módulo **os** de Python permite realizar operaciones dependiente del Sistema Operativo como crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, etc.

- **Crear una carpeta nueva**

```
>>> import os
>>> os.makedirs("Mi_Carpeta")
```

**Dejanos tu feedback!** 👍

```
>>> import os
>>> os.listdir("./")
['Mi_Carpeta']
```

## Contenido de la clase



- **Mostrar el actual directorio de trabajo**

```
>>> import os
>>> os.getcwd()
'/home/usuario/python/'
```

- **Mostrar el tamaño en bytes del archivo pasado como parámetro**

```
>>> import os
>>> os.path.getsize("Mi_Archivo")
4096
```

- **Verificar si es un archivo el parámetro pasado**

```
>>> import os
>>> os.path.isfile("Mi_Archivo")
True
```

- **Verificar si es una carpeta el parámetro pasado**

```
>>> import os
>>> os.path.isdir("Mi_Carpeta")
True
```

- **Cambiar directorio/carpeta**

```
>>> import os
>>> os.chdir("Mi_Carpeta")
>>> os.getcwd()
'/home/usuario/python/Mi_Carpeta'
>>> os.listdir("./")
[]
>>> os.chdir("../")
>>> os.getcwd()
'/home/usuario/python'
```

- **Renombrar un archivo**

```
>>> import os
>>> os.rename("Mi_Archivo","Mi_Otro_Archivo")
```

**Dejanos tu feedback! 👍**

- **Eliminar un archivo**

```
>>> import os
>>> os.chdir("Mi_Carpeta")
>>> archivo = open(os.getcwd()+'/datos.txt', 'w
>>> archivo.write("Hola Mundo!")
>>> archivo.close()
>>> os.getcwd()
'/home/usuario/python/Mi_Carpeta'
>>> os.listdir("./")
['datos.txt']
>>> os.remove(os.getcwd()+"/datos.txt")
>>> os.listdir("./")
[]
```

- **Eliminar una carpeta**

```
>>> os.rmdir("Mi_Carpeta")
>>> os.chdir("Mi_Carpeta")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
OSError: [Errno 2] No such file or directory:
```

Lanza una excepción OSError cuando intenta acceder al directorio que previamente elimino y este no encuentra.

# Extra

## Entornos Virtuales

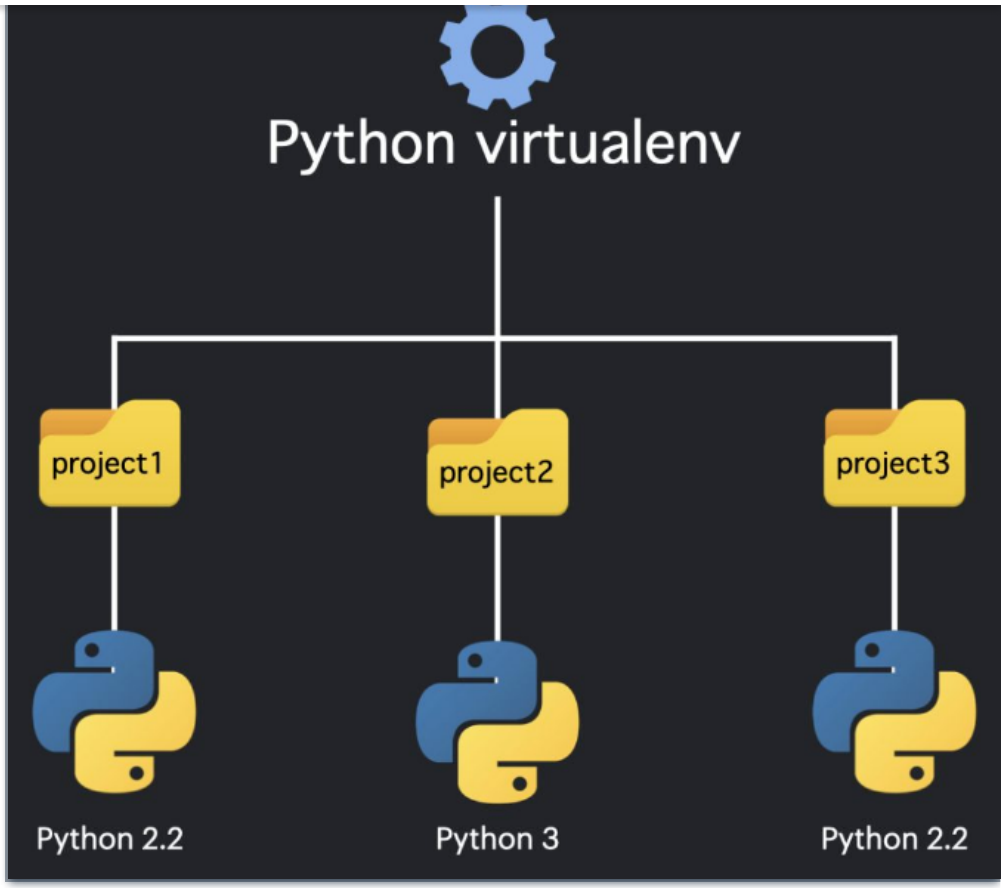
### Creación de Entornos Virtuales

Em adelante, veremos que es muy común trabajar con diferentes librerías ó módulos y uno de los problemas que se puede tener a la hora de manejar la instalaciones de paquetes es que si queremos tener varias versiones de la misma librería, para lo que deberíamos usar lo que se conoce como **entornos virtuales**.
Por ejemplo tenemos un proyecto en el que estamos trabajando con la versión de Python 2.2 y tenemos otro con la versión 3. Para este tipo de cosas es necesario tener dos entornos virtuales y es donde librerías como virtualenv.

**Dejanos tu feedback!** 👍

## Contenido de la clase






Si se desea una version especifica de Python, esta debe estar instalada en la PC. Descargar desde la pagina oficial.
Se recomienda crear una carpeta de facil acceso donde almacenaremos nuestros ambientes. Nos paramos en esta carpeta para ejecutar lo siguiente.

En la terminal:

1. **Instalar virtualenv:** `pip install virtualenv`
2. **Verificamos la versión:** `virtualenv --version`
   **opcional: Podemos hacer un update de pip** `python -m pip install --upgrade pip`
3. **Creamos el entorno:** `virtualenv env_name`
   **Para especificar python:** `virtualenv env_name --python=python=3.x` **x: version**
   **si esto tira error, directamente hay que pasarle la ruta del ejecutable de python:** `virtualenv --python="C:\Users\...\PythonXX\python.exe" env_name`
4. **Iniciamos el entorno virtual** `.\env_name\Scripts\activate`
5. **Instalamos paquetes necesarios. Chequear que en la direccion en la tarminal nos tiene que aparecer** `(env_name) PS C:....` **. Esto nos asegura que estamos dentro del ambiente creado**
6. **Instalamos todos los paquetes necesarios para nuestro entorno:** `pip install paquetes`

**Dejanos tu feedback!** 👍

- Para chequear lista de paquetes instalados: `pip freeze`
- Para exportar los "requierments" o nombres de paquetes instalados: `pip freeze > requirements.txt`
- Para instalar `requirements.txt` : pip install -r requirements.txt

7. Finalmente desactivamos el entorno virtual: `deactivate`

Algunos errores:

Recordar que toda PC es diferente y puedes tener distintos errores, estos son solo algunos que pueden pasar

1. si al cargar una libreria: OSError: Windows requires Developer Mode to be activated, or to run Python as an administrator, in order to create symlinks. In order to activate Developer Mode go to https://learn.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
   Solución: Coloque Windows en modo desarrollador: Configuración → Actualizaciones y seguridad → Para desarrolladores → Usar funciones de desarrollador: seleccione "Modo desarrollador"

2. al activar el entorno. Error: "La ejecución de scripts está deshabilitada en este sistema"
   Solución:

   - Para poder ver la política actual de ejecución abriremos PowerShell a nivel administrador. Para ello deberemos hacer clic en Inicio, escribir "Windows PowerShell", hacer clic con el botón derecho encima de la aplicación y finalmente hacer clic en "Ejecutar como administrador".
   - Una vez abierta la aplicación ejecutaremos el siguiente comando: `Get-ExecutionPolicy -List`
   - Esto nos muestra que la política de ejecución no está definida. Para poder corregir esto deberemos ejecutar el siguiente comando: `Set-ExecutionPolicy RemoteSigned -Scope CurrentUser`

### Pipenv

Pipenv es otro administrador de ambientes virtuales para Python.

Instalación:

1. Abrimos la consola/terminal/bash. En Windows podemos buscarla en el buscador

**Dejanos tu feedback!** 👍

2. **Escribimos la instrucción 'pip install pipenv' en la consola, lo que hace que se installen pipenv y las dependencias necesarias en el directorio donde tenemos Python. IMPORTANTE: Recuerden que deben tener Python instalado y agregado al PATH. Esto se hace descargando el instalador de python.org y siguiendo las instrucciones (recurden chequear la opción Agregar Python al PATH)**
3. **Una vez instalado pipenv, nos dirigimos al directorio donde queremos crear el ambiente e utilizamos el comando pipenv shell. Este comando inicializa el ambiente contenido en ese directorio y, sino existe ninguno, lo crea.**

Algunos comandos útiles son:

- **pipenv -h**
- **pipenv install**
- **pipenv lock**
- **pipenv install --ignore-pipfile**
- **pipenv graph**
- **pipenv uninstall**
- **pipenv uninstall --all**

## Web Scrapping

Se utiliza esta técnica para extraer información desde sitios web, en primer lugar es necesario tener descargada una librería de Python, llamada Beautiful Soup que nos permite realizar esta acción:

```
>>> !pip install beautifulsoup4
>>> from bs4 import BeautifulSoup
>>> import urllib.request
```

Luego, vamos a extraer el HTML de un sitio web, por ejemplo de Wikipedia:

```
response = urllib.request.urlopen('https://es.
html = response.read()
```

Ahora, es necesario quitar todo lo que corresponda a código, y quedarse sólo con la parte del texto, que es lo que realmente interesa

```
soup = BeautifulSoup(html, 'html.parser')
text = soup.get_text()
```

En la variable 'text' ahora tenemos el texto del sitio web al que consultamos.

**Dejanos tu feedback! 👍**

# Homework



Completa la tarea descrita en el archivo **README**

Si tienes dudas sobre este tema, puedes consultarlas en el canal #python de Slack

Hecho con 💛 por alumnos de Henry

**Dejanos tu feedback! 👍**