Herramientas para trabajar con *bucles* y *datos* en Python

- Vamos a trabajar con esta estructura en toda la materia:

  Con el **script** ubicado en un **directorio** de la PC voy a:

  1) Controlar uno o más instrumentos (en bucle)
  2) Guardar datos
  3) Cargar datos
  4) Graficar datos

https://docs.python.org/3/tutorial/classes.html

1. Control de instrumentos

Usamos el paquete PyVisa
(https://pyvisa.readthedocs.io/en/latest/)

Ejemplo de comunicación con un instrumento

```
 9 import pyvisa as visa
10
11 rm = visa.ResourceManager()
12
13 instrumentos = rm.list_resources()
14 print(instrumentos)
```

```
In [1]: runfile('C:/Users/User/Google Drive/Laboratorio 4 2dp2021/
Instrumentos/comunicación.py', wdir='C:/Users/User/Google Drive/
Laboratorio 4 2dp2021/Instrumentos')
('USB0::0x0699::0x0363::C065093::INSTR', 'ASRL4::INSTR',
'ASRL5::INSTR')

In [5]: instrumentos?
Type:        tuple
String form: ('USB0::0x0699::0x0363::C065093::INSTR', 'ASRL4::INSTR',
'ASRL5::INSTR')
Length:      3
```

```
23 #Con ese nombre abro el vinculo con el osciloscopio
24
25 osc=rm.open_resource(instrumentos[0])
26 #osc=rm.open_resource('USB0::0x0699::0x0363::C065093::INSTR')
```

osc es un objeto que representa a la comunicación con el instrumento

1. Control de instrumentos: bucles (*loops*)

Los bucles sirven para repetir intrucciones una cantidad de veces prefijada (*for*) o hasta cumplir alguna condción (*while*).

for Lleva un objeto que pueda ser iterado: string, range, list, np.Array, etc.
en cada bucle la variable toma un valor de ese objeto hasta recorrer todos sus elementos.

```
import numpy as np

voltajes = np.linspace(10,10000,5)
for i in voltajes:
    # Instrucciones
    print(i)
```

```
10.0
2507.5
5005.0
7502.5
10000.0
```

```
for j in range(5):
    # Instrucciones
    print(j)
```

```
0
1
2
3
4
```

```
for keke in ["a","b","c"]:
    # Instrucciones
    print(keke)
```

```
a
b
c
```

while Lleva un statement que tiene que ser True o False (i.e., bool).
El contenido se ejecuta hasta que el statement sea False.

```
i = 0
while i < 5:
    print(i)
    i += 1
```

```
0
1
2
3
4
```

2. Guardar datos

https://numpy.org/doc/stable/reference/generated/numpy.savetxt.html

numpy.savetxt

```
numpy.savetxt(fname, X, fmt='%.18e', delimiter=' ', newline='\n',
header='', footer='', comments='# ', encoding=None)                [source]
```

Save an array to a text file.

Ejemplo: tengo dos listas o arrays y los quiero guardar en columnas (traspongo):

```
>> np.savetxt("mediciones.txt", np.transpose([tiempos, voltajes]))
```

%.5f float decimal de 5 cifras después de la coma

```
0.00000 1.00000
0.00726 3.00000
0.02322 5.00000
0.03921 7.00000
0.05514 9.00000
```

%.5g mas corto entre f y e/E, hasta 5 cifras significativas

```
0 1
0.0072596 3
0.023217 5
0.039212 7
```

%.18e exponencial 18 digitos

```
0.000000000000000000e+00 1.000000000000000000e+00
7.259607315063476562e-03 3.000000000000000000e+00
2.321743965148925781e-02 5.000000000000000000e+00
3.921198844909667969e-02 7.000000000000000000e+00
5.513596534729003906e-02 9.000000000000000000e+00
7.108902931213378906e-02 1.100000000000000000e+01
```

3. Cargar datos

https://numpy.org/doc/stable/reference/generated/numpy.genfromtxt.html

numpy.genfromtxt

```
numpy.genfromtxt(fname, dtype=<class 'float'>, comments='#', delimiter=None,
                 skip_header=0, skip_footer=0, converters=None, missing_values=None,
                 filling_values=None, usecols=None, names=None, excludelist=None,
                 deletechars=" !#$%&'()*+, -./:;<=>?@[\\]^{|}~", replace_space='_',
                 autostrip=False, case_sensitive=True, defaultfmt='f%i', unpack=None,
                 usemask=False, loose=True, invalid_raise=True, max_rows=None,
                 encoding='bytes', *, like=None)
```

Ejemplos

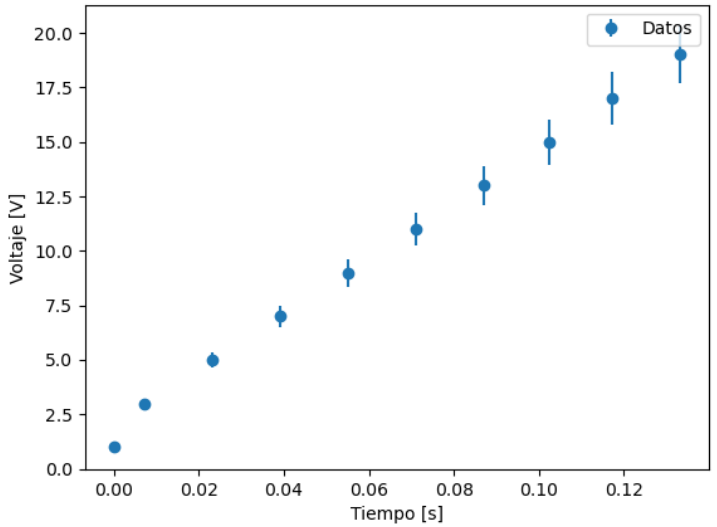
delimiter =	'\t' ',' ';' ' '
Skip_header =	2
Usemask = True	Si no todas las columnas tienen la misma cantidad de filas
Usecols	Permite cargar solo columnas de interés
Unpack	Permite guardar las columnas por separado:

tiempo, voltaje = np.genfromtxt("mediciones.txt", unpack=True)

4. Graficar datos

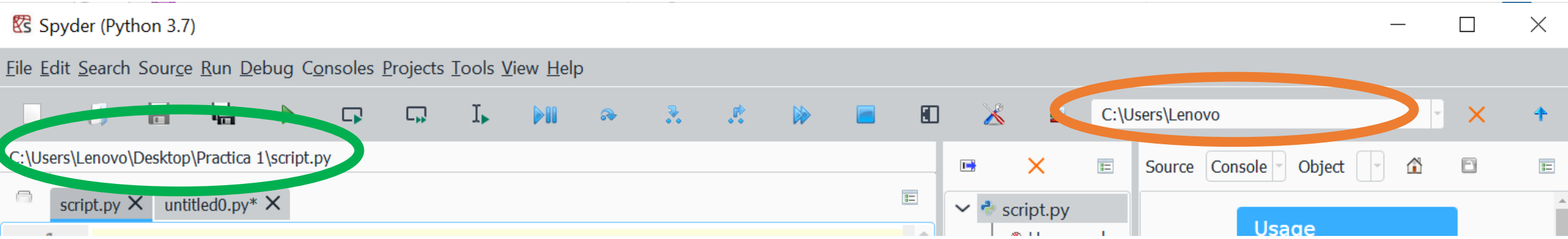
https://matplotlib.org/stable/tutorials/introductory/usage.html

```
fig, ax = plt.subplots(num="Grafico")

ax.errorbar(datos[:,0], datos[:,1], yerr=0.07*datos[:,1], fmt='o', label="Datos")

# Nombres de ejes
ax.set_xlabel("Tiempo [s]")
ax.set_ylabel("Voltaje [V]")
ax.legend()
# guardamos la figura graficada como un archivo
plt.savefig('figurita1.png', format='png') #esta línea guarda la figura
# trae la figura al frente.
plt.show()
```



Yapa: Reglas sobre directorios

- Cuando abro Spyder, la consola (Python) se ejecuta por defecto en:
  C:\Users\Lenovo
- Si ejecuto *script.py* entero (con F5) ubicado en:
  C:\Users\Lenovo\Desktop\Practica 1\script.py
  entonces el directorio de la consola pasa a ser el del script.
- Si solo ejecuto línea por línea, el directorio de la consola no cambia.



- Las **direcciones absolutas** contienen toda la ruta hasta el archivo.
  Ej.: “C:\\Users\\Lenovo\\Desktop\\Practica 2\\script.txt”

Absolute paths

→ No son compatibles con cambiar de ubicación la carpeta de trabajo
→ Como \ es un “caracter de escape” (se usa para indicar saltos de línea **\n** , tabulaciones **\t**, etc.), la contrabarra de Windows se escribe: ****

- Las **direcciones relativas**, en cambio, son relativas al directorio donde se está ejecutando el script (i.e.: cwd*).

Relative paths

- **Practica 1**
  - Medicion1.txt → ..\Practica 1/Medicion1.txt

cwd

- **Practica 2**
  - script.py → Script.py
  - graficar.py
  - **Mediciones**
    - Medicion1.txt → Mediciones/Medicion1.txt
    - Medicion2.txt

.. **Sirve para ir al *parent directory***

* *Current work directory*

Yapa: Reglas sobre directorios

- **Practica 1**
  - Medicion1.txt
- **Practica 2**
  - script.py
  - graficar.py
  - **Mediciones**
    - Medicion1.txt
    - Medicion2.txt

- Ejemplo: guardo en carpeta Mediciones:

```
np.savetxt("Mediciones\\" + filename, np.transpose([tiempos, voltajes]), delimiter="\t", fmt="%.4g")
```

- Si quiero cambiar el cwd, puedo usar la biblioteca **os**

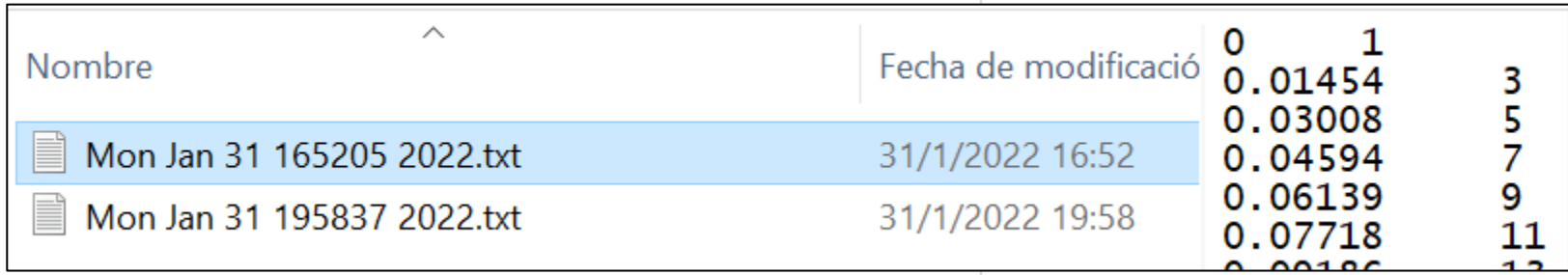
  import os

  os.chdir(C:\\Users\\Lenovo**)**

Yapa 2: usar time para guardar datos

```
time.ctime()
'Mon Jan 31 19:57:12 2022'
```

```python
import time
import numpy as np
import matplotlib.pyplot as plt

tiempos = []
voltajes = []

t0 = time.time() # Esto devuelve los segundos desde EPOCH
for i in range(10):
    # Agrego tiempo actual
    t = time.time() - t0
    tiempos.append(t)

    # Agrego medicion
    v = 2*i + 1 # v = osc.query("Acá alguna instrucción que devuelve un valor")
    voltajes.append(v)

    time.sleep(0.001)

# Guardo los datos en Mediciones\
filename = time.ctime().replace(':','') + ".txt"
np.savetxt("Mediciones\\" + filename, np.transpose([tiempos, voltajes]), delimiter="\t", fmt="%.4g")
datos = np.genfromtxt("mediciones.txt")
```




```
0       1
0.01454     3
0.03008     5
0.04594     7
0.06139     9
0.07718     11
```