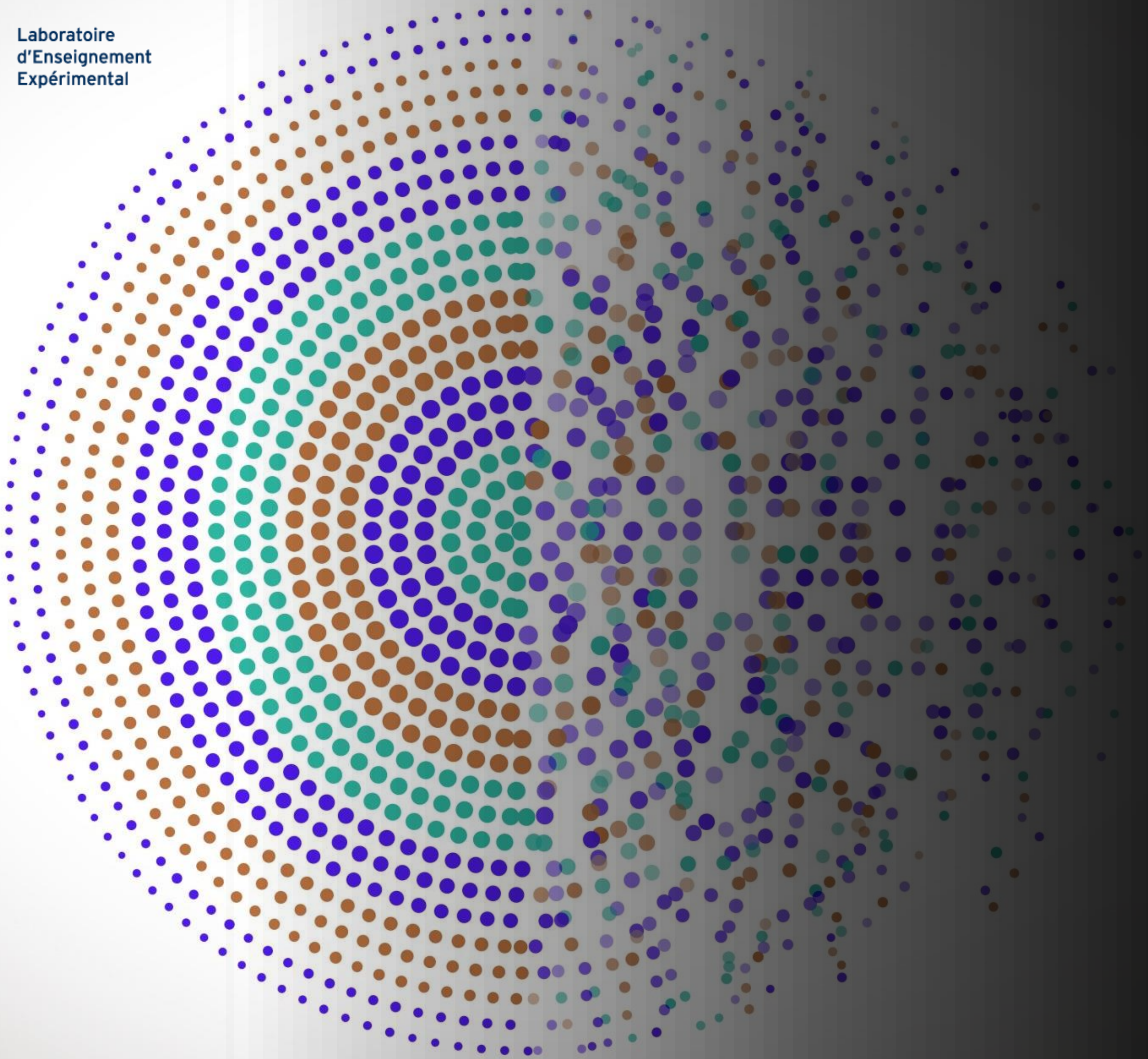
INSTITUT
d'OPTIQUE
GRADUATE SCHOOL
ParisTech

LEnsE
Laboratoire
d'Enseignement
Expérimental



Python
Sciences

Bonnes
pratiques

Outils Numériques / Semestre 5
Institut d'Optique / B3_1

Vecteurs / matrices / Exemple 1

- Exemple 1a

```
N = 10
vect = []
for i in range(N):
	vect.append(0)
```

- Exemple 1b

```
import numpy as np
N = 10
vect = np.zeros(N)
```

Vecteurs / matrices / Exemple 1

- Exemple 1a

```
N = 10
vect = []
for i in range(N):
    vect.append(0)
```

```
print(type(vect))
```

```
<class ‘list’>
```

- Exemple 1b

```
import numpy as np
N = 10
vect = np.zeros(N)
```

```
print(type(vect))
```

```
<class ‘numpy.ndarray’>
```

Vecteurs / matrices / Exemple 1

- Exemple 1a

```
N = 10
vect = []
for i in range(N):
	vect.append(0)
```

```
print(type(vect))
```

```
<class ‘list’>
```

Temps exécution :

N=10 ~1 us
N=1000 ~100 us

- Exemple 1b

```
import numpy as np
N = 10
vect = np.zeros(N)
```

```
print(type(vect))
```

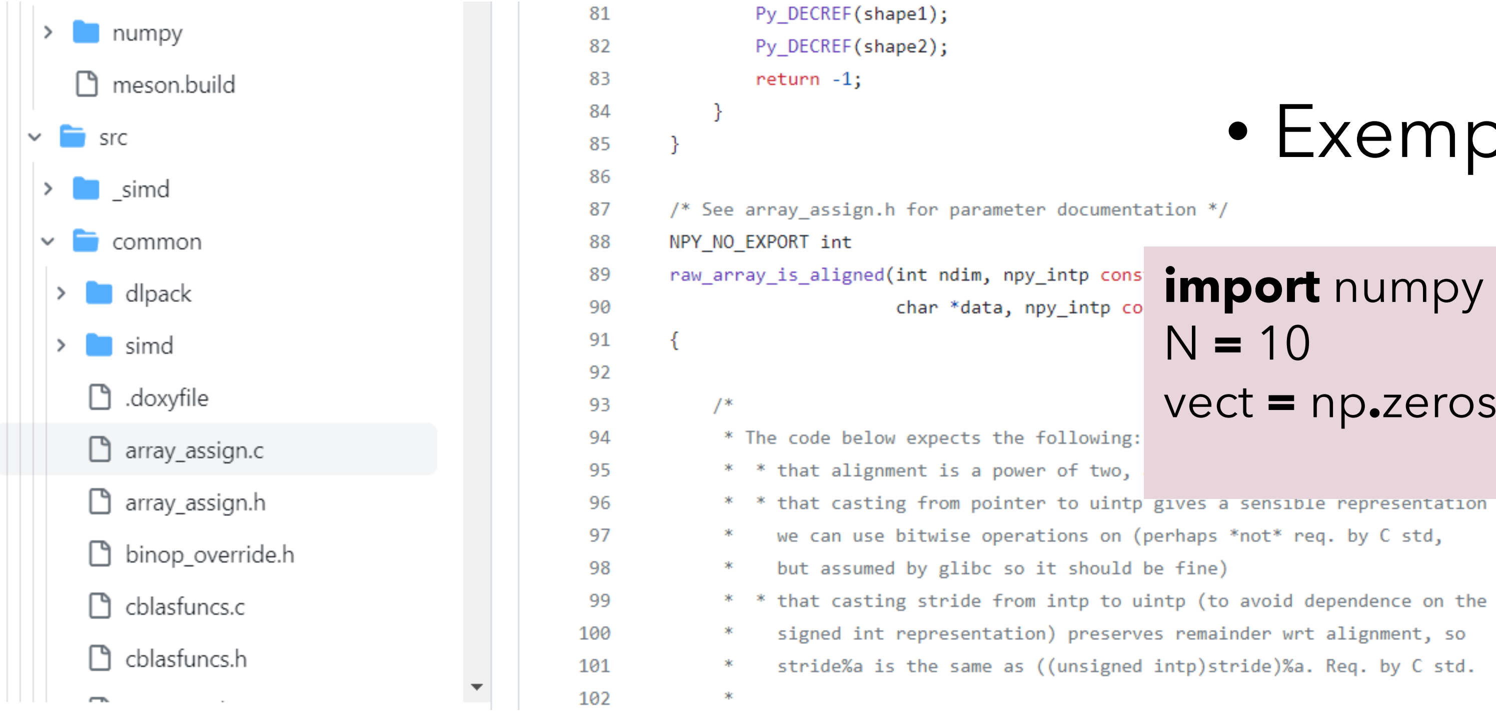
```
<class ‘numpy.ndarray’>
```

Temps exécution :

N=10 ~0,25 us
N=1000 ~0,6 us

Exécution : Core(TM) i7-9750H CPU @ 2.60GHz / 32 Go RAM

Vecteurs / matrices / Exemple 1



```
        Py_DECREF(shape1);
        Py_DECREF(shape2);
        return -1;
    }
}

/* See array_assign.h for parameter documentation */
NPY_NO_EXPORT int
raw_array_is_aligned(int ndim, npy_intp cons
                     char *data, npy_intp co
{

    /*
     * The code below expects the following:
     *  * that alignment is a power of two,
     *  * that casting from pointer to uintp gives a sensible representation
     *    we can use bitwise operations on (perhaps *not* req. by C std,
     *    but assumed by glibc so it should be fine)
     *  * that casting stride from intp to uintp (to avoid dependence on the
     *    signed int representation) preserves remainder wrt alignment, so
     *    stride%a is the same as ((unsigned intp)stride)%a. Req. by C std.
     *
```

- Exemple 1b

```
import numpy as np
N = 10
vect = np.zeros(N)
```

Temps exécution :
N=10 ~1 us
N=1000 ~100 us

Temps exécution :
N=10 ~0,25 us
N=1000 ~0,6 us

Exécution : Core(TM) i7-9750H CPU @ 2.60GHz / 32 Go RAM

Vecteurs / matrices / Exemple 2

```
import numpy as np
data = np.random.rand(1000,5)
print(data.shape)
```

```
(1000, 5)
```

Vecteurs / matrices / Exemple 2

```
import numpy as np
data = np.random.rand(1000,5)
print(data.shape)
```

```
(1000, 5)
```

- Exemple 2a

```
v1 = []
v2 = []
for i in range(len(data)):
   v1.append(data[i, 1])
   v2.append(data[i, 2])
print(type(v1))
```

```
<class ‘list’>
```

- Exemple 2b

```
v1 = data[:, 1]
v2 = data[:, 2]
print(type(v1))
```

```
<class ‘numpy.ndarray’>
```

Vecteurs / matrices / Exemple 2

```
import numpy as np
data = np.random.rand(1000,5)
print(data.shape)
```

```
(1000, 5)
```

- Exemple 2a

```
v1 = []
v2 = []
for i in range(len(data)):
   v1.append(data[i, 1])
   v2.append(data[i, 2])
print(type(v1))
```

```
<class 'list'>
```

Temps exécution :
~400 us

- Exemple 2b

```
v1 = data[:, 1]
v2 = data[:, 2]
print(type(v1))
```

```
<class 'numpy.ndarray'>
```

Temps exécution :
~0,5 us

Fonctions / Exemple 3

```
def sinus(t, A, f):
    return A*np.sin(2*np.pi*f*t)

time_vect = np.linspace(0, 1, 1001)
```

- Exemple 3a

```
TF = np.fft.fft(sinus(time_vect, 1, 10))
plt.figure()
plt.plot(time_vect, sinus(time_vect, 1, 10))
```

- Exemple 3b

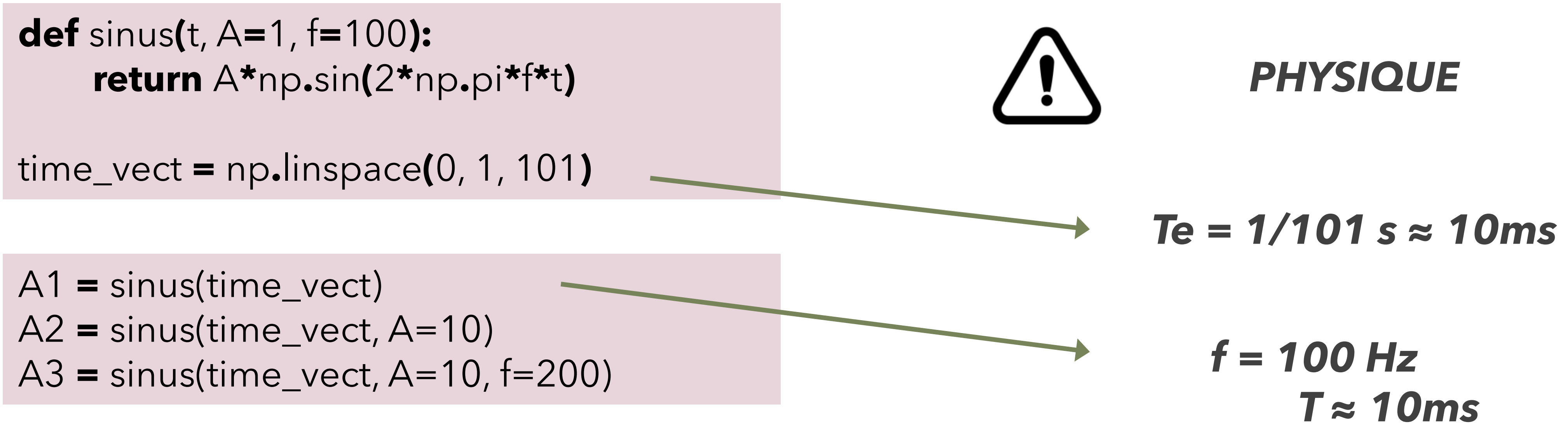
```
sig = sinus(time_vect, 1, 10)
TF = np.fft.fft(sig)
plt.figure()
plt.plot(time_vect, sig)
```

Fonctions / Paramètres optionnels

```
def sinus(t, A=1, f=100):
    return A*np.sin(2*np.pi*f*t)

time_vect = np.linspace(0, 1, 101)
```

```
A1 = sinus(time_vect)
A2 = sinus(time_vect, A=10)
A3 = sinus(time_vect, A=10, f=200)
```

Fonctions / Paramètres optionnels



```
def sinus(t, A=1, f=100):
        return A*np.sin(2*np.pi*f*t)

time_vect = np.linspace(0, 1, 101)
```

```
A1 = sinus(time_vect)
A2 = sinus(time_vect, A=10)
A3 = sinus(time_vect, A=10, f=200)
```

⚠ ***PHYSIQUE***

Te = 1/101 s ≈ 10ms

f = 100 Hz
T ≈ 10ms

Critère de Shanonn-Nyquist non respecté

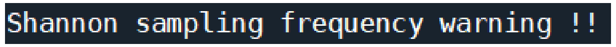
Fonctions / Paramètres optionnels

```
def sinus(t, A=1, f=100):
	if(isinstance(t, np.ndarray)):
		Te = t[0] - t[1]
		if(1/Te < 2*f) print(‘Shannon 
sampling frequency warning !!’)
	return A*np.sin(2*np.pi*f*t)

time_vect = np.linspace(0, 1, 101)
```

```
A1 = sinus(time_vect)
A2 = sinus(time_vect, A=10)
A3 = sinus(time_vect, A=10, f=200)
```

⚠ ***PHYSIQUE***

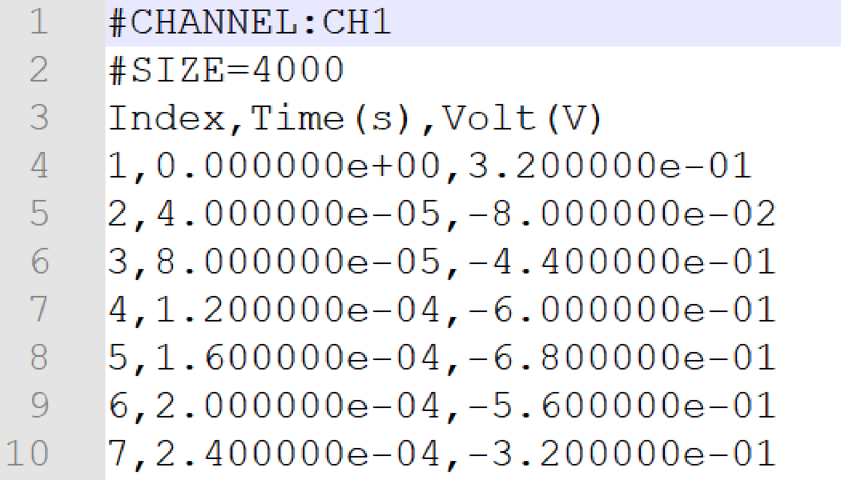
Te = 1/101 s ≈ 10ms



```
Shannon sampling frequency warning !!
```

Fichiers / Exemple 4

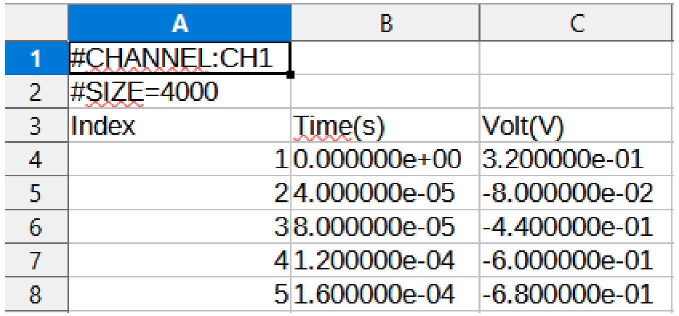
- Différents types de fichiers
  - ASCII / texte
  - Binaires



```
#CHANNEL:CH1
#SIZE=4000
Index,Time(s),Volt(V)
1,0.000000e+00,3.200000e-01
2,4.000000e-05,-8.000000e-02
3,8.000000e-05,-4.400000e-01
4,1.200000e-04,-6.000000e-01
5,1.600000e-04,-6.800000e-01
6,2.000000e-04,-5.600000e-01
7,2.400000e-04,-3.200000e-01
```

Fichier CSV

Codage ASCII
Délimiteur de colonnes
(défaut : point-virgule)



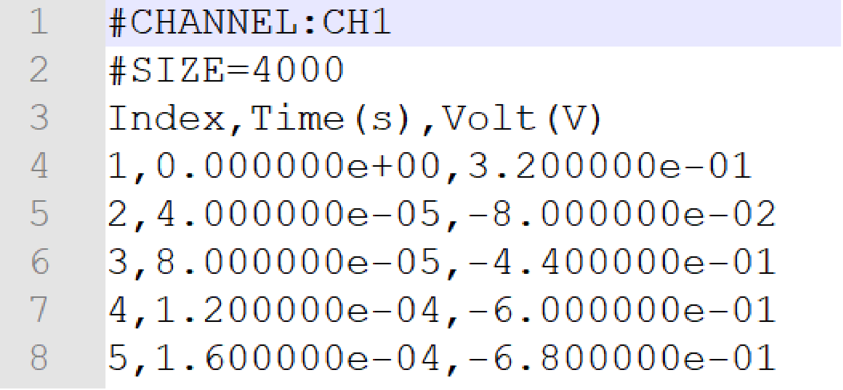
	A	B	C
1	#CHANNEL:CH1		
2	#SIZE=4000		
3	Index	Time(s)	Volt(V)
4	1	0.000000e+00	3.200000e-01
5	2	4.000000e-05	-8.000000e-02
6	3	8.000000e-05	-4.400000e-01
7	4	1.200000e-04	-6.000000e-01
8	5	1.600000e-04	-6.800000e-01

Fichiers / Exemple 4

- Selon les applications

En-tête

Données



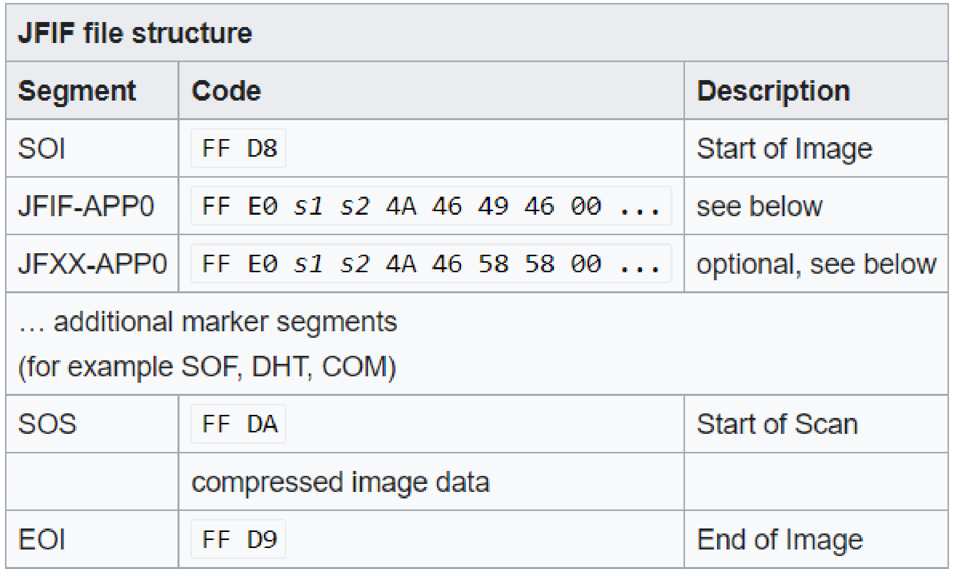
```
#CHANNEL:CH1
#SIZE=4000
Index,Time(s),Volt(V)
1,0.000000e+00,3.200000e-01
2,4.000000e-05,-8.000000e-02
3,8.000000e-05,-4.400000e-01
4,1.200000e-04,-6.000000e-01
5,1.600000e-04,-6.800000e-01
```

Fichier CSV

Codage ASCII
Délimiteur de colonnes
(défaut : point-virgule)

Fichier JPG

Codage binaire
En-tête spécifique



JFIF file structure

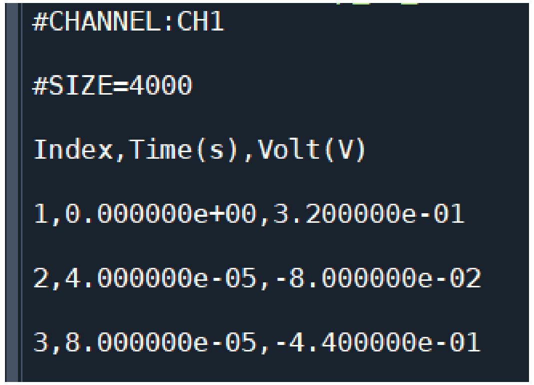
Segment	Code	Description
SOI	`FF D8`	Start of Image
JFIF-APP0	`FF E0 s1 s2 4A 46 49 46 00 ...`	see below
JFXX-APP0	`FF E0 s1 s2 4A 46 58 58 00 ...`	optional, see below
… additional marker segments (for example SOF, DHT, COM)		
SOS	`FF DA`	Start of Scan
	compressed image data	
EOI	`FF D9`	End of Image

https://en.wikipedia.org/wiki/JPEG_File_Interchange_Format

Fichiers / Exemple 4

- Exemple 4a

```
f = open("B3_data_01.csv", "r")
cpt = 0
N = 10
for line in f:
    if cpt < N :
        print(line)
        cpt += 1
f.close()
```



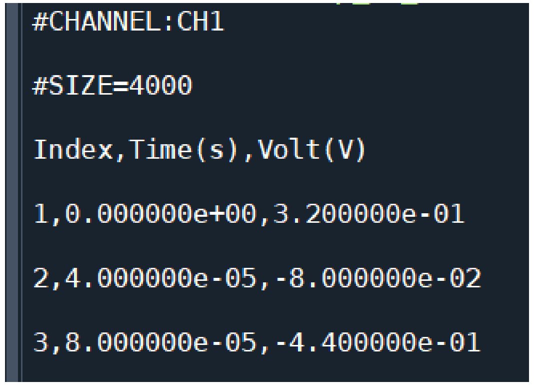
```
#CHANNEL:CH1

#SIZE=4000

Index,Time(s),Volt(V)

1,0.000000e+00,3.200000e-01

2,4.000000e-05,-8.000000e-02

3,8.000000e-05,-4.400000e-01
```

Fichiers / Exemple 4

- Exemple 4a

```
f = open("B3_data_01.csv", "r")
cpt = 0
HEADER = 2
NB_DATA = 4000
delimiter = ','
t = np.zeros(NB_DATA )
v = np.zeros(NB_DATA )
```

```
for line in f:
    if (cpt > HEADER) and (cpt < (HEADER + NB_DATA + 1)) :
        data = line.split(delimiter)
        t[cpt-HEADER-1] = float(data[1])
        v[cpt-HEADER-1] = float(data[2])
    cpt += 1
f.close()
```



```
#CHANNEL:CH1

#SIZE=4000

Index,Time(s),Volt(V)

1,0.000000e+00,3.200000e-01

2,4.000000e-05,-8.000000e-02

3,8.000000e-05,-4.400000e-01
```

Fichiers / Exemple 4

- Exemple 4a

```
f = open("B3_data_01.csv", "r")
cpt = 0
HEADER = 2
NB_DATA = 4000
delimiter = ','
t = np.zeros(NB_DATA )
v = np.zeros(NB_DATA )
```

```
for line in f:
    if (cpt > HEADER) and (cpt < (HEADER +
NB_DATA + 1)) :
        data = line.split(delimiter)
        t[cpt-HEADER-1] = float(data[1])
        v[cpt-HEADER-1] = float(data[2])
    cpt += 1
f.close()
```

Temps exécution :
~6 ms (pour 4000 données)

- Exemple 4b

```
data = np.genfromtxt("B3_data_01.csv",
delimiter=',', skip_header=2,
skip_footer=6)
t = data[:,1]
v = data[:,2]
```

Temps exécution :
~11 ms (pour 4000 données)