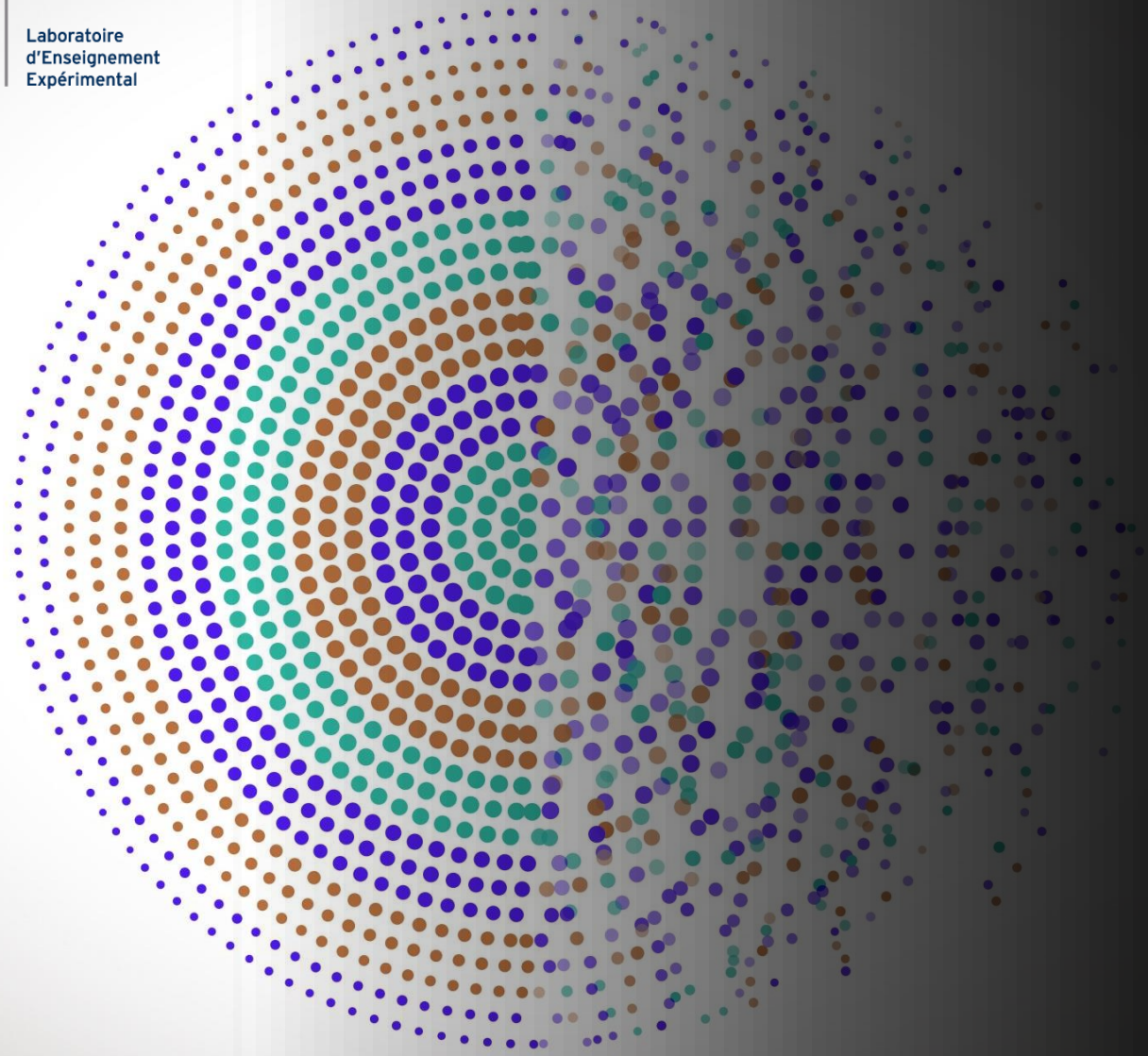
INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental



Python Sciences

Bonnes pratiques (2)

Outils Numériques / Semestre 5
Institut d'Optique / B3_2

Format de données

- Stockage dans des fichiers

Fichier CSV

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

Fichier JPG

Codage binaire
En-tête spécifique

En-tête

Données

```
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
```

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

- Fichier CSV

```
f_raw = open("B3_data_01.csv", "r")
print(type(f_raw))
```

```
<class '_io.TextIOWrapper'>
```

- Fichier binaire

```
f_raw = open("B3_data_02.txt", "rb")
print(type(f_raw))
```

```
<class '_io.BufferedReader'>
```

Fichiers / Exemple 4

- Fichier CSV

```
f_raw = open("B3_data_01.csv", "r")
print(type(f_raw))
```

```
<class '_io.TextIOWrapper'>
```

```
data = f.read()
print(type(data))
```

```
<class 'str'>
```

- Fichier binaire

```
f_raw = open("B3_data_02.txt", "rb")
print(type(f_raw))
```

```
<class '_io.BufferedReader'>
```

```
data = f.read()
print(type(data))
```

```
<class ‘bytes'>
```

Données binaires

- Données

octet1	octet2	octet3	octet4	…	octetN

Données binaires

- Données / binaire vs ASCII

octet1	octet2	octet3	octet4	…	octetN
0x00	0x30	0x41	0x42	…	0x0A
NUL	‘0’	‘A’	‘B’	…	LF

<str> -> code ascii

N caractères

Données binaires

- Données / binaire vs ASCII

octet1	octet2	octet3	octet4	…	octetN
0x00	0x30	0x41	0x42	…	0x0A

NUL	‘0’	‘A’	‘B’	…	LF

<str> -> code ascii

N caractères

0	48	65	66	…	10

<bytes> -> valeurs entières

N valeurs

Données binaires

- Données

octet1	octet2	octet3	octet4	…	octetN
0x00	0x10	0x20	0x00	…	0x26

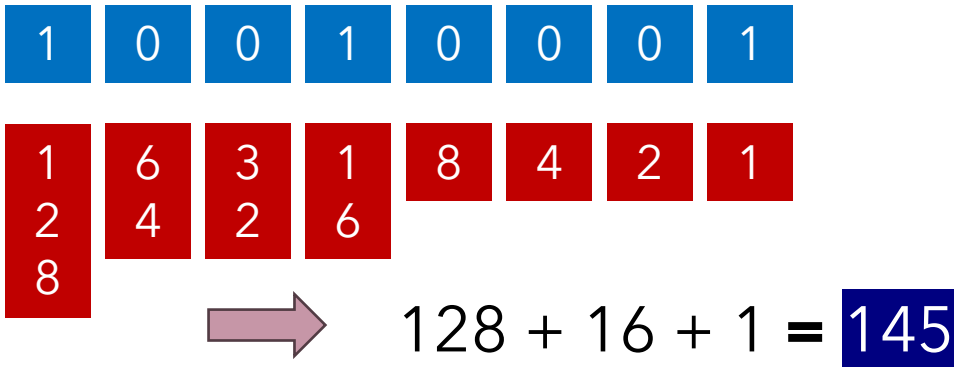
0	16	32	0	…	38

1 échantillon = 1 octet

N échantillons

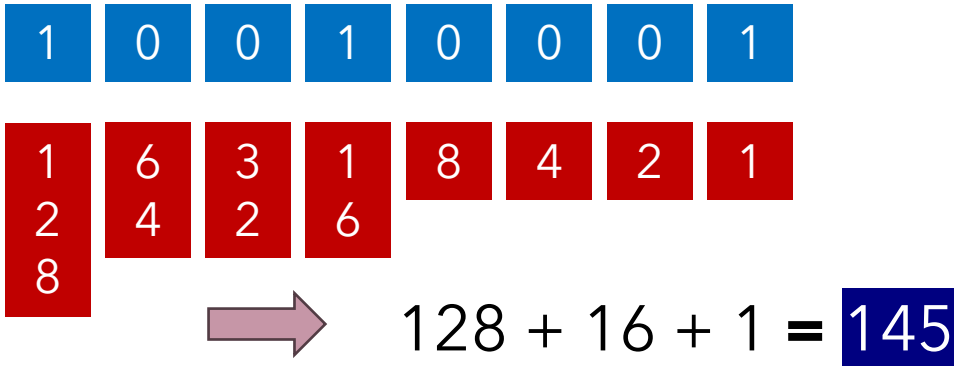
Données binaires / signées ou non signées

- 1 octet
- Donnée non signée (uint8)

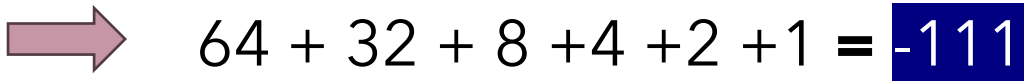


Données binaires / signées ou non signées

- 1 octet
- Donnée non signée (uint8)

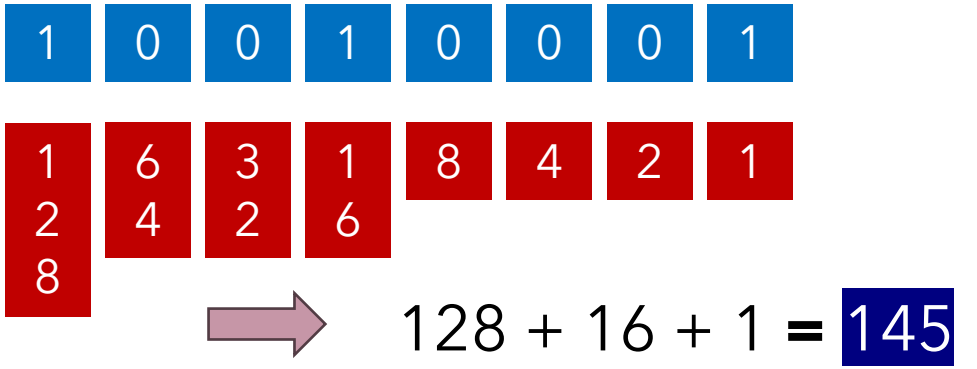


- Donnée signée (int8)
  (complément à 2)

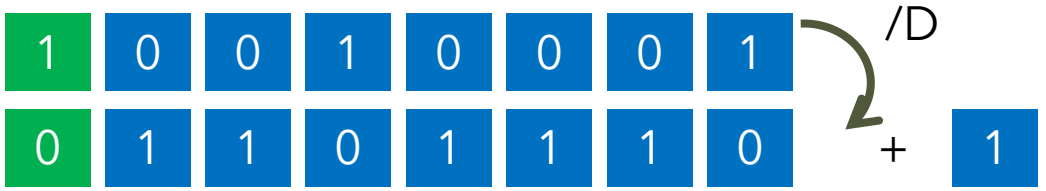


Données binaires / signées ou non signées

- 1 octet
- Donnée non signée (uint8)



- Donnée signée (int8)
  (complément à 2)



Données binaires

- Données

octet1	octet2	octet3	octet4	…	octetN
0x00	0x10	0x20	0x00	…	0x26

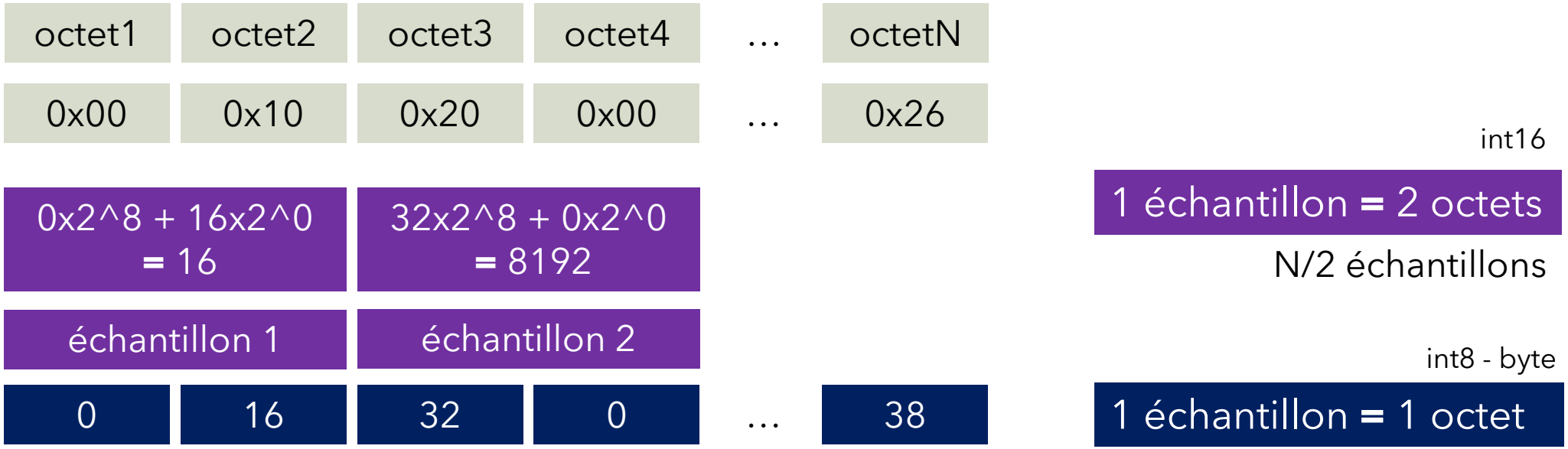
0	16	32	0	…	38

1 échantillon = 1 octet

N échantillons

Données binaires

- Données



Fichier à décoder

Fichier binaire Base64
Signal utile : 24kHz / 16bits

- Fichier **B3_data_02.txt**

```
f_raw = open("B3_data_02.txt", "rb")
```

Fichier à décoder

Fichier binaire Base64
Signal utile : 24kHz / 16bits

- Fichier **B3_data_02.txt**

```
f_raw = open("B3_data_02.txt", "rb")
data = f_raw.read()
```

data | octet1 | octet2 | … | octetN

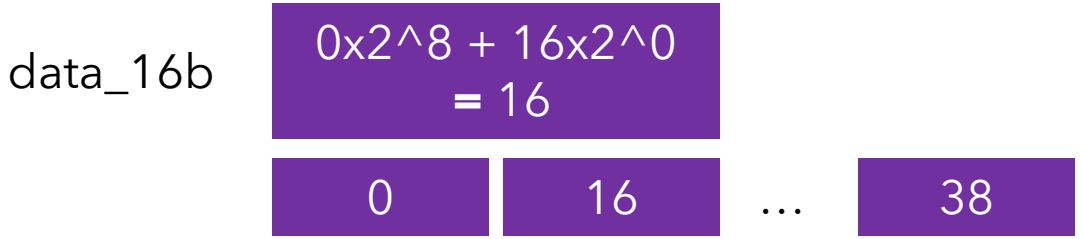
Fichier à décoder

Fichier binaire Base64
Signal utile : 24kHz / 16bits

- Fichier **B3_data_02.txt**

```
f_raw = open("B3_data_02.txt", "rb")
data = f_raw.read()
```


```
data_16b = np.frombuffer(data_d,
dtype=np.int16)
```



Fichier à décoder

Fichier binaire Base64
Signal utile : 24kHz / 16bits

- Fichier **B3_data_02.txt**

```
f_raw = open("B3_data_02.txt", "rb")
data = f_raw.read()
```

```
data_d = base64.b64decode(data)
```

```
data_16b = np.frombuffer(data_d,
dtype=np.int16)
```

Décodage / décompression

data	octet1	octet2	…	octetN
	octet1	octet2	…	octetN2
data_d	0x00	0x10	…	0x26

data_16b

échantillon 1
0x2^8 + 16x2^0 = 16

Fichier à décoder

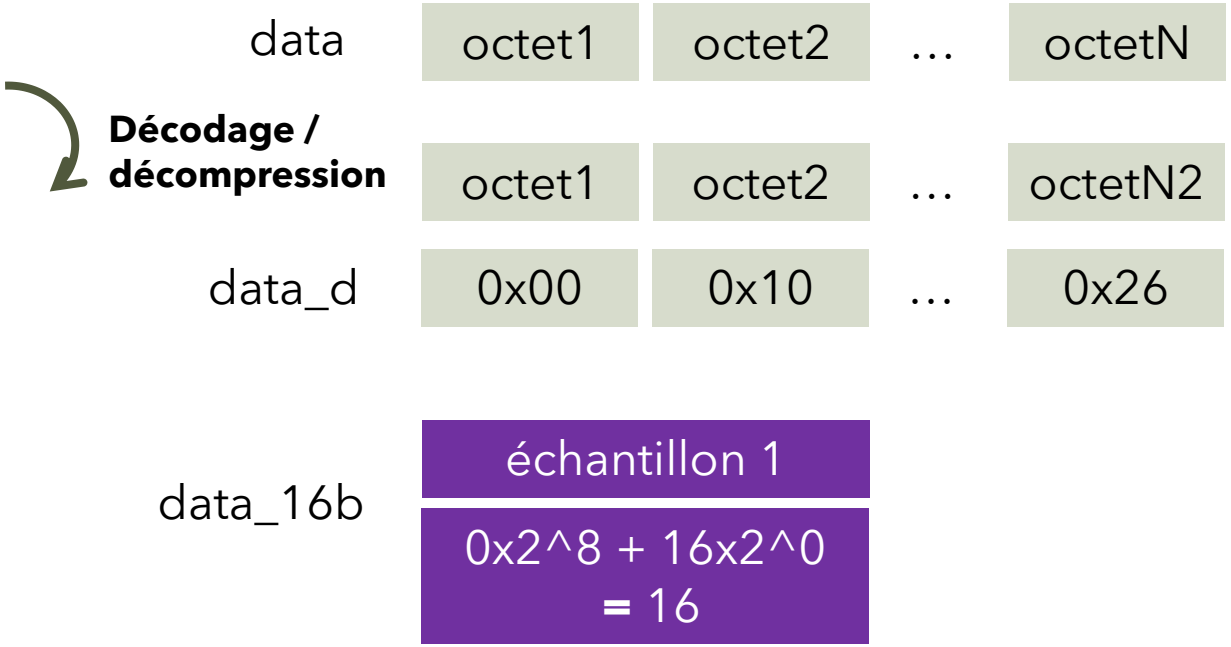
Fichier binaire Base64
Signal utile : 24kHz / 16bits

- Fichier **B3_data_02.txt**

```
f_raw = open("B3_data_02.txt", "rb")
data = f_raw.read()
```

```
data_d = base64.b64decode(data)
```

```
data_16b = np.frombuffer(data_d,
dtype=np.int16)
```



+ generation du vecteur temps (Fe = 24kHz)

INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental