

Numpy, Arrays & Images

Pour certaines activités du calcul scientifique, nous devons manipuler de grandes quantités de données. Le module `numpy` propose un type structuré adapté à cet usage : le type `array`.

Dans ce chapitre, le module `numpy` sera noté `np` en utilisant la syntaxe `import numpy as np`.

1 Type array de numpy

Un objet de type `array` est un tableau de dimension supérieure ou égale à 1. A la différence d'une liste de liste, il est codé en machine sous la forme d'un tableau contigu. La **convention** de codage est de le représenter comme une **liste de lignes**.

1.1 Création des arrays

1.1.1 Création explicite

La création explicite d'un `array` passe par la conversion d'une liste de listes :

```
M=np.array([[1,2],[3,4]])
```

1.1.2 Création automatique

1. Tableau monodimensionnel

- **arange** : équivalent à *range*, mais génère un `array` :
`arange(b_inf,b_sup,pas)`, où la borne inférieure est incluse, et la borne supérieure est exclue.

```
>>> np.arange(0,2.5,.5)
array([0.,0.5,1.,1.5,2.])
```

- **linspace**(i, f, n) : construit un tableau de dimension 1 de n nombres décimaux uniformément répartis sur l'intervalle $[i; f]$:

```
>>> np.linspace(0,10,5)
array([ 0., 2.5, 5., 7.5, 10.])
```

2. Tableau de dimension supérieure à 1

- **zeros**($[l, c]$) : construit un tableau de l lignes et c colonnes rempli de 0

```
>>> np.zeros([2,3])
array([[0.,0.,0.],
       [0.,0.,0.]])
```

Il existe une fonction équivalente `np.ones()`.

- **identity**(n) : construit un tableau correspondant à la matrice identité de dimension n

```
>>> np.identity(3)
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

1.2 Attributs des arrays

Un **array** dispose d'un certain nombre d'attributs le caractérisant. Ils sont accessibles en utilisant les commandes suivantes :

- **M.dtype** renvoie le type des éléments de M
(voir en annexe les différents types des array)

```
M.dtype
dtype('int32')
```

- **M.ndim** renvoie le nombre de dimensions de M :

```
M.ndim
2
```

- **M.shape** renvoie la forme de l'array M. Si il s'agit d'une matrice bidimensionnelle, il s'agira du tuple (nlig,ncol)

```
M.shape
(3, 4)
```

D'autres attributs peuvent être utiles :

- **M.size** renvoie le nombre d'éléments de M;
- **M.itemsize** renvoie le nombre d'octets occupé par chaque élément de M;
- **M.nbytes** renvoie le nombre total d'octets occupé par M.

1.3 Manipulation

1.3.1 Redimensionnement

On considère le tableau `H=np.zeros((60))`. Pour conserver les informations d'un tableau et modifier ses dimensions, on peut modifier l'attribut :

```
H.shape=(15,4)
```

La méthode **reshape** permet de renvoyer un tableau avec de nouvelles dimensions :

```
I=H.reshape((20,3))
```

1.3.2 Arithmétique

Un tableau de type *array* n'est pas une liste : les opérations "+" et "*" réalisent les opérations mathématiques d'addition et de multiplication **terme à terme** :



```
>>> A=np.array([1,2,3,4,5,6]).reshape((2,3))
>>> A+A
array([[ 2,  4,  6],
       [ 8, 10, 12]])
>>> A*A
array([[ 1,  4,  9],
       [16, 25, 36]])
```

1.4 Extraction ou *Slicing*

On peut accéder de différentes façon aux éléments de l'*array* A :

- accès à l'intégralité du tableau avec A
- accès à un élément du tableau avec $A[3, 2]$
- accès à une ligne du tableau avec $A[3, :]$
- accès à une colonne du tableau avec $A[:, 2]$
- accès à un sous tableau avec $A[2 : 4, 1 : 3]$

```
>>> a[0,3:5]
array([3,4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```

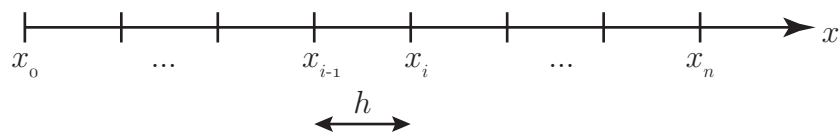
0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

1.5 Comparaison Array et List

1.6 Usage en Sciences

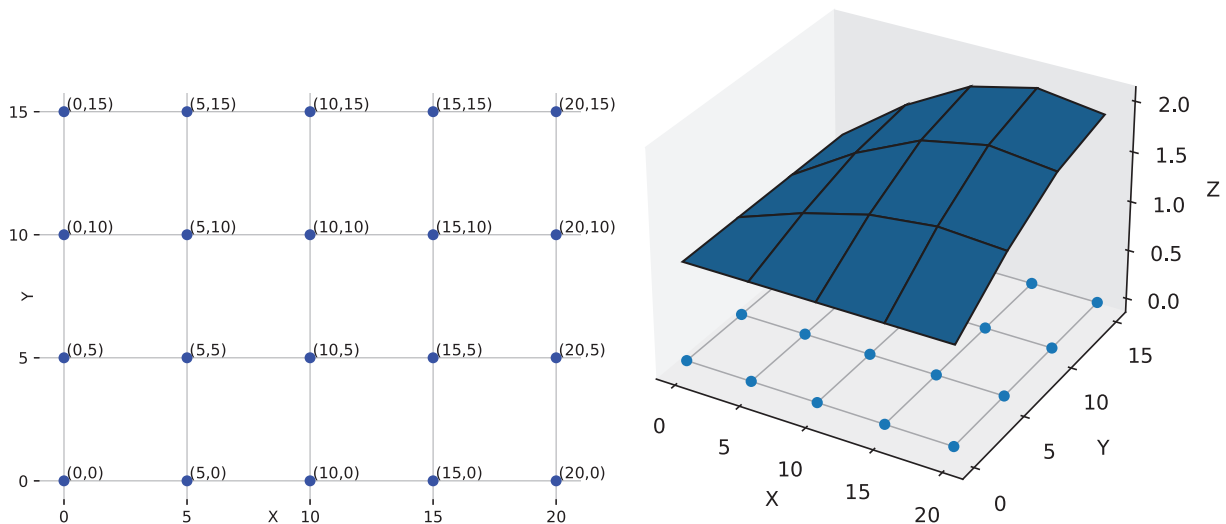
1.6.1 Discrétiser une ligne (axe x ou le temps)

Ne pouvant traiter des grandeurs continues, un programme va traiter des grandeurs **discrétisées**. Dans notre cas, l'axe des abscisses est découpé en intervalles réguliers, dits **de pas** $h = \Delta x$. Seules les valeurs particulières notées x_i sont traitées.



1.6.2 Discrétiser un plan

Pour étudier un phénomène dépendant des coordonnées d'un point dans le plan, il faut discrétiser le plan.



On peut alors définir les coordonnées des points par deux tableaux à 2 dimensions, regroupant chacun les abscisses et les ordonnées des points :

```
X= [[0, 5, 10, 15, 20],
     [0, 5, 10, 15, 20],
     [0, 5, 10, 15, 20],
     [0, 5, 10, 15, 20]]
Y= [[0, 0, 0, 0, 0],
     [5, 5, 5, 5, 5],
     [10, 10, 10, 10, 10],
     [15, 15, 15, 15, 15]]
```

1.6.3 Application à l'algèbre linéaire

Le type array de numpy permet d'effectuer des opérations matricielles d'algèbre linéaire :

- `np.dot(A,B)` ou `A@B` effectue le produit matriciel de A par B ;
- `A.transpose()` (méthode) renvoie la matrice transposée de A. On obtient le même résultat avec l'attribut : `A.T` ;
- `np.linalg.inv(A)` renvoie l'inverse de A ;
- `np.linalg.det(A)` renvoie le déterminant de A ;
- `np.linalg.matrix_rank(A)` renvoie le rang de A ;
- `np.linalg.eigvals(A)` renvoie un array liste contenant les valeurs propres de A ;
- `np.linalg.eig(A)` renvoie un tuple contenant deux éléments :

1. Le premier élément est un array contenant les valeurs propres de A
2. Le deuxième élément est une matrice de vecteurs propres de A, dans l'ordre de l'array précédent.

2 Images

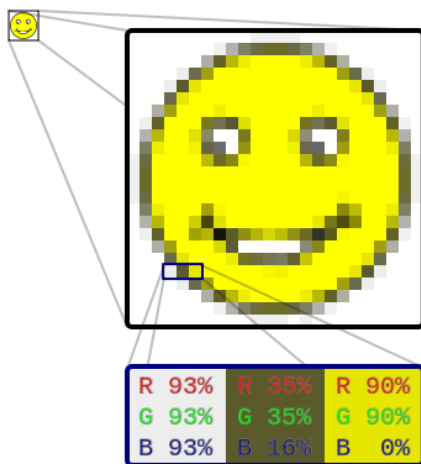
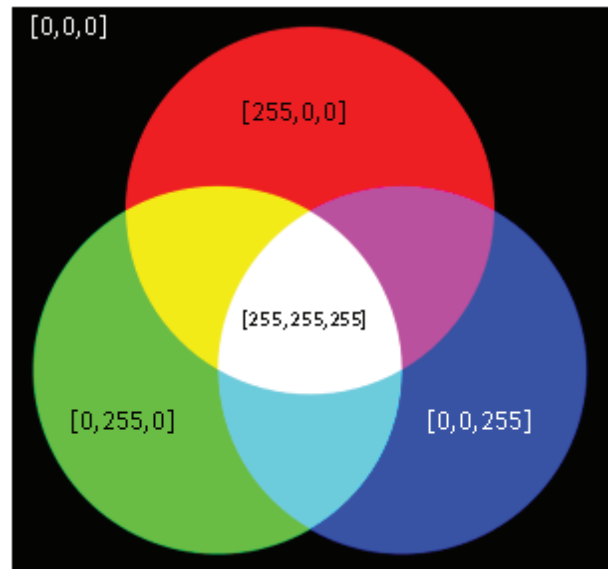
2.1 Codage d'une image

Une image est formée de *pixel* (« picture element ») : un pixel est un carré de la plus petite dimension affichable par l'écran (ou projecteur) considéré, au quel on affecte une couleur.

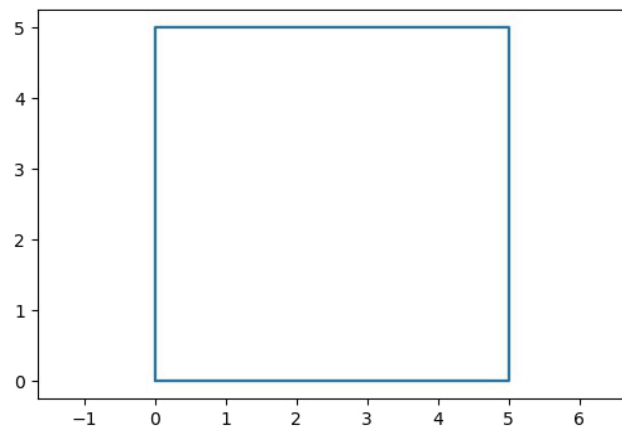
A chaque pixel est associé une information :

- booléenne pour une image en noir ou blanc ;
- entier naturel pour une image en nuance de gris (compris entre 0 et 255) ;
- 3 entiers naturels pour une image en couleur.

Pour définir la couleur d'un pixel, il faut préciser les trois intensités des trois couleurs primaires qui formeront la couleur du pixel.



Par exemple, le module `matplotlib` crée ses figures par défaut avec des dimensions de 6 x 4 pouces, et une résolution de 100 pixels par pouce.



2.2 Manipulation d'image



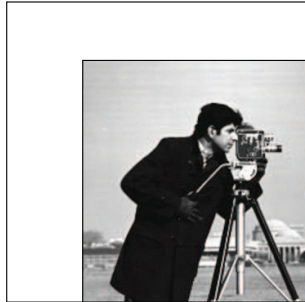
On considère l'image ci-contre :

2.2.1 Manipulations géométriques

Découpage et répétition

Sans répétition :

Translation :



Avec répétition :



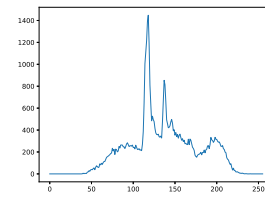
Rotation :



Mise à l'échelle

2.2.2 Traitement des couleurs

Histogramme :

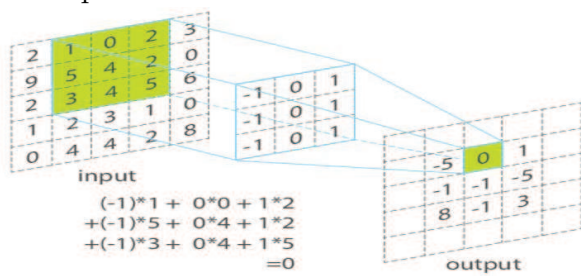


Luminosité :

Contraste :

2.2.3 Traitement par convolution discrète

Principe :



3 Annexes

3.1 Les types d'array

Data type	Description
bool_	Boolean (True or False) stored as a byte
int_	Default integer type (same as C long ; normally either int64 or int32)
intc	Identical to C int (normally int32 or int64)
intp	Integer used for indexing (same as C ssize_t ; normally either int32 or int64)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float_	Shorthand for float64.
float16	Half precision float : sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float : sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float : sign bit, 11 bits exponent, 52 bits mantissa
complex_	Shorthand for complex128.
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

3.2 Mémo – Manipulation d'images



```

import numpy as np          #bibliothèque de gestion des tableaux
from PIL import Image       #bibliothèque de gestion des images
import matplotlib.pyplot as plt  #bibliothèque de gestion des graphiques

T = np.array(Image.open("image.png")) #charge une image dans un tableau

#Pour afficher l'image d'un tableau T, quatre instructions sont nécessaires.
plt.figure()
plt.imshow(T)               #pour une image en couleurs RGB(A) ou noir & blanc
plt.imshow(T,cmap='gray')   #pour une image en nuances de gris
plt.axis("off")             #pour cacher les graduations
plt.show()

Image.fromarray(T).save("fichier.png") #enregistrer une image en "fichier.png"

```