Numpy et Images

1. Type array de numpy

`numpy` propose plusieurs types dédiés aux tableaux de nombres : en particulier le type `array` et le type `matrix`. Nous allons étudier ici le type array.

1.1. **Création simple d'un array.** Le type array se déclare à l'aide de la construction `np.array` :

On veut rentrer la matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$. Ceci se fait à l'aide de :

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

Il est également possible de transformer une liste (contenant le bon nombre d'éléments) en array :
`M=np.array([1,2,3,4,5,6,7,8,9,10,11,12]).reshape(3,4)` crée :

```
array([[ 1, 2, 3, 4],
       [ 5, 6, 7, 8],
       [ 9, 10, 11, 12]])
```

1.2. **Attributs des array.** Les commandes suivantes sont utiles :

- `M.shape` (sans parenthèses, il s'agit d'un attribut) 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)
  ```

- `M.size` renvoie le nombre d'éléments de M :

  ```
  M.size
  12
  ```

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

  ```
  M.ndim
  2
  ```

- `M.type` renvoie le type des éléments de M (voir paragraphe sur le typage des array) :

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

- `M.itemsize` renvoie le nombre d'octets occupé par chaque élément de M :

  ```
  M.itemsize 4
  ```

1.3. **Création d'array particuiers.** Certaines matrices particulières peuvent être créées à l'aide d'une commande :

- `np.zeros((n,p))` créé la matrice composée de 0. à n lignes et p colonnes ;
- `np.ones((n,p))` créé la matrice composée de 1. à n lignes et p colonnes ;
- `np.eye(n)` créé la matrice identité flottante d'ordre n ;
- `np.diag(L)` créé la matrice diagonale où les termes diagonaux sont les termes de la liste L. La matrice est du même type que les éléments de L.

1.4. **Opérations sur les array.** Contrairement aux listes de listes, les opérations sur les tableaux numpy sont faites élément par élément :

```
T=np.array([[1,2],[3,4]])
U=np.array([[1,0],[0,1]])
print(T+U) # addition terme à terme.
print(2*T) # multiplication de tous les éléments par 2.
```

Cette série de commandes va afficher :

array([[2, 2],
[3, 5]])

et

array([[2, 4],
[6, 8]])

Avec des tableaux numpy, le produit T*U ne produit pas d'erreur et a bien un sens ici : celui du produit terme à terme des tableaux, comme pour l'addition. Ainsi `print(T*U)` renvoie :

array([[1, 0],
[0, 4]])

Attention, pour avoir un sens, les opérations précédentes doivent se faire sur des tableaux ayant même taille :

```
T = np.array([[0,1],[2,3],[4,5]])
U = np.array([[4,5],[6,7]])
print(T+U) #addition d'un tableau 3*2 avec un tableau 2*2: erreur.
```

1.5. **Typage des array.** Les tableaux numpy sont `statiques` et **typés**.

Lors de la déclaration d'un array, l'espace mémoire nécessaire pour son stockage est calculé, et créé dans une zone mémoire contigüe. Cette méthode permet d'optimiser la vitesse de calcul sur ces tableaux, et explique pourquoi les modules numpy et scipy, très efficaces, sont effectivement utilisés en calcul scientifique professionnel.

Voici un extrait de la documentation numpy :

Numpy supports a much greater variety of numerical types than Python does. This section shows which are available, and how to modify an array's data-type.

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)

Le type peut être déclaré explicitement lors de la création de l'array :

```
np.array([[1,2],[3,4]],dtype='float64')
```

crée ainsi :

array([[1., 2.],
[3. , 4.]])

1.6. **Exercice.**

Exercice 1. On considère M une matrice composée de flottants.

Écrire les matrices dont les éléments sont :

(1) ceux de M divisés par 3
(2) ceux de M au carré
(3) le logarithme de ceux de M
(4) ceux de M auxquels on a appliqué la fonction $f(x) = \cos^2 x$ si $x > 3$ et $f(x) = 1 - x$ sinon.

2. Application à l'algèbre linéaire

Le type array de numpy permet d'effectuer des opérations matricielles d'algèbre linéaire :
- `np.dot(A,B)` effectue le produit matriciel de A par B ;
- `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.

Exercice 2. On considère la matrice J_n, matrice carrée d'ordre n, dont tous les termes sont égaux à 1.
Donner ses éléments propres.

Exercice 3. La commande `take` permet de définir des matrices extraites. Ainsi les lignes de codes suivantes :

```
m = np.arange(16).reshape(4,4)
print(m.take([0,3], axis = 0)
print(m.take(range(1,3), axis = 1)
```

produisent :

```
[[ 0 1 2 3]
 [12 13 14 15]]
```

et

```
[[ 1 2]
 [ 5 6]
 [ 9 10]
 [13 14]]
```

Écrire une fonction qui calcule le déterminant d'une matrice carrée récursivement. Justifier sa correction et sa terminaison. Quelle est la complexité ?

3. Images

Les images qu'on va utiliser sont des images dites matricielles. Elles sont composées d'une matrice (tableau) de points colorés appelés pixels. Les formats d'images matricielles qu'on utilisera seront BMP ou PNG en « couleurs vraies ». Chaque pixel est un triplet de nombres entre 0 et 255 : un nombre pour chaque couleur primaire rouge, vert, bleu. Un tel nombre est représentable sur 8 bits et s'appelle un octet. Il y a donc $2^{24} = 16777216$ couleurs possibles. On utilise ici la synthèse additive des couleurs : le triplet (0 ; 0 ; 0) correspond à un pixel noir alors qu'un pixel blanc est donné par (255 ; 255 ; 255). Un pixel « pur rouge » est codé par (255 ; 0 ; 0).

Dans une image en niveaux de gris, chaque pixel est noir, blanc, ou a un niveau de gris entre les deux. Cela signifie que les trois composantes R, V, B ont la même valeur.

Les niveaux de rouge, vert et bleu étant identiques, les informations sont redondantes si l'on sait que l'on a une image en niveau de gris. Il est possible de sauvegarder la matrice des pixels non pas en associant à chaque pixel un triplet de niveau (R,V,B) mais une valeur unique égale au niveau de gris

Voici un code permettant d'ouvrir une image à l'aide du module `PIL` et de la stocker dans un tableau numpy :

```
from PIL import Image #importation du sous-module Image du module PIL
im = Image.open("image.png") #ouverture d'une image au format png dans Python.
tab = np.array(im)
```

Voici un code permettant de créer une image à partir d'un tableau numpy :

```
nouvelle_image = Image.fromarray(tab)
nouvelle_image.show() # pour afficher l'image
nouvelle_image.save("nom_image.png") # pour l'enregistrer au format voulu
```

Exercice 4. Le texte suivant est extrait de wikipedia :

La fréquence d'image, exprimée en images par seconde, peut être imposée par le contexte de diffusion, ou mentionnée après la lettre « p » (ou « i »). Par exemple, « 1080p30 » signifie que la fréquence est de 30 images par seconde pour un balayage progressif de 1 080 lignes.

La signification exacte du sigle 1080p peut dépendre du contexte. Parfois, le terme 1080p signifie que l'écran affiche 1 920 pixels de large et 1 080 pixels de haut.

La fréquence d'image, exprimée en images par seconde, peut être imposée par le contexte de diffusion, ou mentionnée après la lettre « p » (ou « i »). Par exemple, « 1080p30 » signifie que la fréquence est de 30 images par seconde pour un balayage progressif de 1 080 lignes

L'industrie du cinéma a adopté le 1080p24 comme son format numérique de mastering, en 24p natif comme en 24PsF. Le 1080p24 est ainsi devenu un standard de production établi pour la cinématographie numérique et beaucoup d'équipements sont capables de capturer et traiter du contenu 1080p24. C'est peut-être le seul standard vidéo universel qui traverse les frontières des continents, tout comme le film argentique à 24 images par seconde.

Quelle est la taille en GO d'une vidéo, non compressée, d'une durée de 90 minutes tournée en 1080p24 ?

La compression d'image est une application de la compression de données sur des images numériques. Cette compression a pour utilité de réduire la redondance des données d'une image afin de pouvoir l'emmagasiner sans occuper beaucoup d'espace ou la transmettre rapidement.

La compression d'image peut être effectuée avec perte de données ou sans perte. La compression sans perte est souvent préférée là où la netteté des traits est primordiale : schémas, dessins techniques. La compression avec perte, est utile pour les transmissions à bas débit, mais dégrade la qualité de l'image restituée. Les méthodes de compression sans perte sont également préférées là où la précision est vitale : balayages médicaux ou numérisations d'images pour archivage. Les méthodes avec perte restent acceptables pour des photos dans les applications où une perte de fidélité est tolérée pour réduire les coûts de stockage ou d'envoi.

Exercice 5. L'œil est plus sensible à certaines couleurs qu'à d'autres. Le vert (pur), par exemple, paraît plus clair que le bleu (pur). Pour tenir compte de cette sensibilité dans la transformation d'une image couleur en une image en niveaux de gris, on ne prend généralement pas la moyenne arithmétique des intensités de couleurs fondamentales, mais une moyenne pondérée. La formule standard donnant le niveau de gris en fonction des trois composantes est :

$$\text{gris} = \lfloor\ 0.299 * \text{rouge} + 0.587 * \text{vert} + 0.114 * \text{bleu}\ \rfloor$$

Écrire une fonction prenant en paramètre une chaîne de caractères `fichier`, représentant le chemin d'une image. Cette fonction ouvrira ce fichier, le convertira en array numpy, transformera à l'aide de la formule précédente cette image en niveaux de gris, et créera une image dont le nom est la concaténation du nom du fichier et de la chaîne de caractères "NB"