Cours

Rappel et pense-bête sur les tableaux

```
import numpy as np
```

1 Création de tableaux

— Tableaux remplis de 1 ou de 0, avec un profit et un type donné :

```
A = np.zeros((4, 5), dtype=int)
    # dimension 2, profit 4 x 5, entiers
B = np.ones((2, 7), dtype="int32")
    # dimension 2, profit 2 x 7, entiers 32 bits
C = np.zeros(6, dtype=float)
    # dimension 1, profit 6, nombres flottants
D = np.ones(8)
    # dimension 1, profit 8, nombres flottants (par défaut)
```

— Même chose pour tableau “vide”, c’est-à-dire non-initialisé :

```
A = np.empty((2, 3), dtype="int8")
B = np.empty(5, dtype=bool)
```

— Création directement à partir d’une liste :

```
A = np.array([[1, 2, 3], [4, 5, 6]])
    # dimension 2, profit 2 x 3, entiers
B = np.array([[1.0, 2, 3], [4, 5, 6]])
    # dimension 2, profit 2 x 3, flottants
C = np.array([-1, 0, 1])
    # dimension 1, profit 3, entiers
```

— Points également espacés (toujours de dimension 1) :

```
A = np.arange(3, 7, 0.5)
    # de 3 inclus à 7 exclu avec pas de 0.5
B = np.linspace(-2, 5, 300)
    # 300 points de -2 inclus à 5 inclus
```

— Matrice identité (toujours de dimension 2) :

```
A = np.eye(7, dtype=int)
    # identité 7 x 7 d'entiers
B = np.identity(7, dtype=int)
    # même chose
```

— Matrices aléatoires (avec `import numpy.random as rd`) :

```
A = rd.random((3, 4))
    # dimension 2, profit 3 x 4, nombres aléatoires de
    # loi uniforme sur [0, 1)
B = 5*rd.random((4, 5)) - 2
    # ou bien :
B = rd.uniform(-2,3,(4,5))
    # dimension 2, profit 4 x 5, nombres aléatoires de
    # loi uniforme sur [-2, 3)
```

— Matrice construite à partir d’une autre à l’aide d’un test logique :

```
B = np.where(A == 5, 0, A)
    # 0 si A[i, j] vaut 5
```

```
    # A[i, j] si A[i, j] ne vaut pas 5

C = np.where(A < 3, 1 - A, 9)
    # Remplace A[i, j] par 1 - A[i, j] si A[i, j] < 3
    # Remplace A[i, j] par 9 si A[i, j] >= 3

D = np.where(A==0, "␣", "*")
    # " " si A[i, j] vaut 0
    # "*" si A[i, j] est différente de 0.
```

2 Propriétés des tableaux

Pour un tableau `A` :

- Dimension : `A.ndim`
- Profit : `A.shape`
- Dans le cas de dimension 2 :
  - Nombre de lignes : `A.shape[0]`
  - Nombre de colonnes : `A.shape[1]`
- Nombre total d'éléments : `A.size`

Attention, ce sont des attributs, pas des méthodes. Il n'y a pas de parenthèses à la fin !

3 Accès aux éléments d'un tableau

Pour un tableau `A` de dimension 1 et profit n, les éléments sont indexés de 0 à $n-1$. Des indices négatifs sont aussi disponibles, -1 correspond au dernier élément, -2 à l'avant dernier,... , $-n$ au premier.

- `A[i]` : accès à l'élément d'indice i.
- `A[-1]` : accès au dernier élément.
- `A[i:j]` : accès aux éléments de i inclus à j exclu.
- `A[i:]` : tous les éléments à partir de i inclus.
- `A[:j]` : tous les éléments jusqu'à j exclu.
- `A[1:]` : tous les éléments à partir du deuxième.
- `A[:-1]` : tous les éléments jusqu'à l'avant dernier.
- `A[i:j] = x` : modification des éléments de i inclus à j exclu. `x` doit être un scalaire ou un tableau de même profit que `A[i:j]`.
- `A[i:j:k]` : tous les éléments de i inclus à j exclu avec un pas de k : i, $i+k$, $i+2k$, ...
- `A[2:8:2]` : tableau avec `A[2], A[4], A[6]`.

Pour un tableau `A` de dimension 2 et profit $n \times m$, on a deux indices séparés par une virgule, le premier de 0 à $n-1$ et le deuxième de 0 à $m-1$. Les indices négatifs sont aussi disponibles.

- `A[i, j]` : accès à l'élément (i, j).
- `A[i, -1]` : accès au dernier élément de la ligne i.
- `A[iMin:iMax, jMin:jMax]` : accès aux sous-tableau dont les lignes vont de `iMin` inclus à `iMax` exclu et les colonnes de `jMin` inclus à `jMax` exclu.
- `A[i, :]` : accès à la ligne i.
- `A[:, j]` : accès à la colonne j.
- `A[-1, :]` : dernière ligne.
- `A[2:5, 3:7]` : lignes 2 à 4, colonnes 3 à 6.
- `A[1:, :6]` : toutes les lignes à partir de la deuxième, toutes les colonnes jusqu'à la sixième.
- `A[2:17:3, 4:18:2]` : lignes de 2 inclus à 17 exclu avec un pas de 3 (donc lignes 2, 5, 8, 11, 14) et colonnes de 4 inclus à 18 exclu avec un pas de 2 (donc colonnes 4, 6, 8, 10, 12, 14, 16).

Attention ! La notation `A[i][j]`, utilisée pour les listes, est à éviter pour les tableaux. Elle marche correctement pour l'accès à un seul élément, mais ne marche pas pour l'accès à un sous-tableau.

4 Opérations avec les tableaux

— `A + B` : somme élément par élément. Erreur si `A` et `B` n'ont pas le même profit.
— `A + x`, avec `x` scalaire : ajoute `x` à chaque élément de `A`.
— `A * B` ; `A * x` : idem, avec le produit.
— `A ** B` ; `A ** x` : idem, avec la puissance.
— `A.dot(B)` : multiplication matricielle. Erreur si les profits ne sont pas compatibles pour le produit.
— `A.transpose()` : transposée de `A`.
— `np.sin(A)` : sinus élément par élément de `A`. Marche aussi avec `cos`, `tan`, `exp`, `log`, etc.
— `A.sum()` ; `A.prod()` : somme / produit de tous les éléments de `A`.

5 Référence

`https://docs.scipy.org/doc/numpy/reference/routines.html` : liste de toutes les fonctions de `numpy` par catégorie.