Calcul matriciel

Informatique pour tous

Questions

On exécute:

```
L1 = [1, 2]
L2 = L1
L2.append(3)
print(L1)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
L1 = [1, 2]
L2 = L1
L2.append(3)
print(L1)
```

Qu'est ce qui est affiché ? `[1, 2, 3]`

Questions

On exécute:

```
a = 1
b = a
a = 2
print(b)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
a = 1
b = a
a = 2
print(b)
```

Qu'est ce qui est affiché ? `1`

Questions

On exécute:

```
L1 = [1, 2]
L2 = L1[:]
L2.append(3)
print(L1)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
L1 = [1, 2]
L2 = L1[:]
L2.append(3)
print(L1)
```

Qu'est ce qui est affiché ? `[1, 2]`

Questions

On exécute:

```
def f(L):
    L.append(3)
L1 = [1, 2]
f(L1)
print(L1)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
def f(L):
    L.append(3)
L1 = [1, 2]
f(L1)
print(L1)
```

Qu'est ce qui est affiché ? `[1, 2, 3]`

Questions

On exécute:

```
def f(x):
    x = 2
a = 1
f(a)
print(a)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
def f(x):
    x = 2
a = 1
f(a)
print(a)
```

Qu'est ce qui est affiché ? 1

Bilan

Les listes (et les tableaux numpy), lorsque assignés à une autre liste (ou tableau) représentent le **même objet**.

Au contraire, les types de bases (`int`, `float`...) sont **copiés**.

Pour réaliser une copie d'une liste `L` on écrira `L[:]` ou `L.copy()`.

Listes de listes

Si `L = [ [1, 2], ["a", True, 4.2] ]` est une liste de listes:

1. `L[1]` est la liste `["a", True, 4.2]`
2. `L[1][2]` est l'élément d'indice 2 de la liste `L[1]`: `4.2`

Il en est de même pour les tableaux `T` numpy, avec une autre syntaxe possible:

```
T[i, j]
```

Créer une matrice avec des listes

On souhaite créer une matrice 4×4 remplie de `False`:

Créer une matrice avec des listes

On souhaite créer une matrice 4×4 remplie de `False`:

```
M = []
L = [False] * 4
for i in range(4):
    M.append(L)
```

Quel est le problème?

Créer une matrice avec des listes

Code correct:

```
M = []
for i in range(4):
    L = [False] * 4
    M.append(L)
```

Créer une matrice avec des tableaux

Pour créer un tableau numpy à partir d'une liste `L`:

```
np.array(L)
```

Si L est une liste de listes alors `np.array(L)` sera un tableau de tableaux...

Manipulation de matrice

Si `M` est une matrice (liste de listes ou tableau de tableaux):

- L'élément sur la $i$ème ligne, $j$ème colonne est:

Manipulation de matrice

Si `M` est une matrice (liste de listes ou tableau de tableaux):

- L'élément sur la *i*ème ligne, *j*ème colonne est: `M[i][j]`
- Le nombre de lignes de `M` est:

Manipulation de matrice

Si `M` est une matrice (liste de listes ou tableau de tableaux):

- L'élément sur la *i*ème ligne, *j*ème colonne est: `M[i][j]`
- Le nombre de lignes de `M` est: `len(M)`
- Le nombre de colonnes de `M` est:

Manipulation de matrice

Si `M` est une matrice (liste de listes ou tableau de tableaux):

- L'élément sur la *i*ème ligne, *j*ème colonne est: `M[i][j]`
- Le nombre de lignes de `M` est: `len(M)`
- Le nombre de colonnes de `M` est: `len(M[0])`

Extraire des lignes ou des colonnes

On considère la matrice `M` suivante:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Que vaut `M[1]`?

Extraire des lignes ou des colonnes

On considère la matrice `M` suivante:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Que vaut `M[1]`?

`array([3, 4, 5])`
Sélectionne la ligne d'indice 1.

Extraire des lignes ou des colonnes

On considère la matrice `M` suivante:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Que vaut `M[1:3]` ?

Extraire des lignes ou des colonnes

On considère la matrice `M` suivante:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Que vaut `M[1:3]`?

```
array([[3, 4, 5],
       [6, 7, 8]])
```

Sélectionne les lignes d'indices 1 et 2.

Modifier des lignes ou des colonnes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment remplacer la ligne d'indice 1 par `[0, 0, 0]`?

Modifier des lignes ou des colonnes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment remplacer la ligne d'indice 1 par `[0, 0, 0]`?

```
M[1] = np.array([0, 0, 0])
```

Modifier des lignes ou des colonnes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment multiplier la ligne d'indice 2 par 7?
(opération de dilatation: $L_2 \leftarrow 7 \times L_2$)

Modifier des lignes ou des colonnes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment multiplier la ligne d'indice 2 par 7?
(opération de dilatation: $L_2 \leftarrow 7 \times L_2$)

```
M[2] = 7*M[2]
```

Échanger deux lignes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment échanger les lignes d'indices 0 et 1?

Échanger deux lignes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment échanger les lignes d'indices 0 et 1?

```
tmp = M[0]
M[0] = M[1]
M[1] = tmp
```

Échanger deux lignes

```
tmp = M[0]
M[0] = M[1]
M[1] = tmp
```

`M` est alors égale à:

```
array([[3, 4, 5],
       [3, 4, 5],
       [6, 7, 8]])
```

Échanger deux lignes

```
tmp = M[0]
M[0] = M[1]
M[1] = tmp
```

`M` est alors égale à:

```
array([[3, 4, 5],
       [3, 4, 5],
       [6, 7, 8]])
```

Problème: `M[0] = M[1]` modifie aussi `tmp`!

Échanger deux lignes

Contrairement aux listes, `[:]` ne réalise pas une copie d'un tableau...

On peut utiliser la méthode `copy` à la place: `T.copy()`.

Échanger deux lignes

```
In [55]: tmp = M[0].copy()

In [56]: M[0] = M[1].copy()

In [57]: M[1] = tmp

In [58]: M
Out[58]:
array([[3, 4, 5],
       [0, 1, 2],
       [6, 7, 8]])
```

Échanger deux lignes

On peut aussi utiliser la possibilité, en Python, d'assigner plusieurs variables en parallèle:

```
a, b = c, d
```

`a` et `b` prennent alors simultanément les valeurs de `c` et `d`.

Échanger deux lignes

On peut donc échanger les lignes `i` et `j` de `M` en écrivant:

Échanger deux lignes

On peut donc échanger les lignes `i` et `j` de `M` en écrivant:

```
M[i], M[j] = M[j].copy(), M[i].copy()
```