TP3_2024_etudiants

January 30, 2024

Calcul scientifique – TP3 – `numpy`

Nom :
Prénom:

Attention: pour ce TP il s'agit d'utiliser numpy et ses fonctionnalités donc pas de bricoler avec des listes et des boucles.

Il n'y aura pas de boucles dans ce TP

0.0.1 1) Création de tableaux

Pour commencer, on importe la librairie `numpy`

[1]:
```
import numpy as np
```

Créer une variable numpy de type `array`, nommée `a` contenant les valeurs de 1 à 20 des 3 manières différentes :

- a1: à partir d'un tableau python t = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
- a2: en utilisant la fonction arange de numpy
- a3: en utilisant la fonction linspace de numpy

puis afficher le contenu et le type de `a1,a2,a3` pour les 3 méthodes. On doit obtenir le même tableau qu'on appellera ensuite a pour continuer l'exercice.

[2]:
```
t = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
# à compléter
print('a1:',a1,'type :',type(a1))

# à compléter
print('a2:',a2,'type :',type(a2))

# à compléter
print('a3:',a3,'type :',type(a3))
```

```
a1: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20] type : <class
'numpy.ndarray'>
a2: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20] type : <class
'numpy.ndarray'>
a3: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20] type : <class
'numpy.ndarray'>
```

Calculer puis afficher la somme des valeurs (en une ligne sans faire de boucles) :

[3]:
```
# à compléter
```

```
somme : 210
```

Créer un tableau de taille 20 qu'on appellera b qui contient 20 nombres aléatoires compris entre 1 et 9 inclus. On utilisera pour cela la fonction `np.random.randint` du module numpy.

[4]:
```
# à compléter
```

```
[6 2 8 3 4 3 5 3 6 8 1 2 5 2 4 2 2 7 1 8]
```

Afficher les dimensions de l' `array a`, le nombre de valeurs selon chaque dimension (shape) ainsi que sa taille (nombre de valeurs) :

[5]:
```
# à compléter
```

```
ndim= 1 shape= (20,) size= 20
```

En utilisant uniquement les opérateurs +, / et l'opérateur de tri de numpy (`np.sort()`) et la fonction np.sum(), calculer la moyenne , la mediane de `b`,ainsi que les valeurs min et max du tableau. (Ne pas utiliser de boucle).

[6]:
```
# à compléter
```

```
moyenne : 4.1 mediane : 4
val max : 8 val min : 1
```

Vérifier vos calculs en utilisant les fonctions numpy existantes.

[7]:
```
# à compléter
```

```
moyenne : 4.1 mediane : 3.5
val max : 8 val min : 1
```

Afficher le maximum et l'indice du premier maximum rencontré dans b Afficher le minimum et l'indice du premier minmum rencontré dans b

[8]:
```
# à compléter
```

```
[6 2 8 3 4 3 5 3 6 8 1 2 5 2 4 2 2 7 1 8]
maxi 8 indice 2 mini 1 indice  10
```

0.0.2 2) Opérations vectorisées

Calculer le tableau numpy qu'on appellera tab2 contenant le carré des valeurs de a, sans faire de boucle.

[9]:
```
# à compléter
```

```
t2 : [  1   4   9  16  25  36  49  64  81 100 121 144 169 196 225 256 289 324
 361 400]
```

Calculer le tableau tab contenant la somme terme à terme des éléments de a et de a2.

[10]:
```
# à compléter
```

```
tab : [  1   4   9  16  25  36  49  64  81 100 121 144 169 196 225 256 289 324
 361 400]
```

0.0.3 3) tableaux multi-dimensions de numpy

Créer un tableau a de taille 6x4 (6 lignes et 4 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus. Vous utiliserez pour cela la fonction `np.random.randint` du module numpy.

Afficher son contenu.

Le résultat attendu ressemble à ce qui suit, mais les valeurs dépendent bien sûr du tirage au sort :

[[5 1 7 2]
[9 3 5 3]
[4 9 2 4]
[3 2 5 1]
[6 2 6 1]
[5 5 9 8]]

On fera ensuite afficher dimension, size et shape.

[11]:
```
import numpy as np
# à compléter
```

```
[[6 8 1 1]
 [6 2 7 2]
 [8 6 8 8]
 [6 7 8 3]
 [7 5 1 6]
 [5 1 3 9]]
ndim= 2 shape= (6, 4) size= 24
```

a) Calculs Créer une matrice de 3x4 contenant des entiers entre 1 et 9 inclus, puis calculer successivement :

```
la somme de toutes les valeurs
la somme de chacune des lignes
la somme de chacune des colonnes
la somme des cubes des valeurs
```

[12]:
```
# à compléter
```

```
Tableau a
 [[1 3 7 7]
 [2 6 3 1]
 [7 9 3 2]]
```

[13]:
```
# à compléter
```

```
somme des valeurs : 51
moyenne des valeurs : 4.25
```

[14]:
```
# à compléter
```

```
somme par ligne : [10 18 13 10]
```

[15]:
```
# à compléter
```

```
somme par colonnes : [18 12 21]
```

[16]:
```
# à compléter
```

```
somme des cubes des nombres : 2073
```

Tester la commande a>=5

En déduire le nombre de valeurs supérieures 5 et la somme des valeurs supérieures 5

[17]:
```
# à compléter
```

```
test a>5:
 [[False False  True  True]
 [False  True False False]
 [ True  True False False]]
```

[18]:
```
# à compléter
```

```
nombre de valeurs supérieures à 5 : 5
```

[19]:
```
# à compléter)
```

```
somme des valeurs supérieures à 5 : 36
```

On notera a_ij le contenu de la case (i,j) du tableau a.

Calculer le tableau b tel que $b_ij = a_ij^2 + 3a_ij - 5$

[20]:
```
# à compléter
```

```
on obtient
  [[ -1  13  65  65]
 [  5  49  13  -1]
 [ 65 103  13   5]]
```

b) Indexation Afficher les valeurs de `tab (fin de la première partie)` dont les **indices** sont pairs (attention, les indices pas les valeurs). L'instruction doit être valable pour n'importe quel tableau à 1 dimension, quel que soit le nombre de valeurs.

[21]:
```
# à compléter
```

```
tab [  2   6  12  20  30  42  56  72  90 110 132 156 182 210 240 272 306 342
 380 420]
val indices pairs : [ 1  3  5  7  9 11 13 15 17 19]
```

Afficher la somme des valeurs de tab dont les **indices** sont pairs

[22]:
```
# à compléter
```

```
val indices pairs : 100
```

c) Modifications dans un tableau Créer un tableau `a` de taille 8x8 (8 lignes et 8 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus.

Pour toutes les questions suivantes, nous travaillerons avec un tableau de taille 8x8 mais vos instructions devraient fonctionner pour un tableau de taille quelconque

On fera un nouveau tableau par question pour mieux observer les résultats produits.

Au moyen du sélecteur approprié, remplacer toutes les valeurs du tableau par la valeur 0. Attention le nombre de lignes / colonnes doit être conservé. Afficher le résultat. On doit remplacer les valeurs, pas créer un nouveau tableau.

[23]:
```
import numpy as np
# à compléter
```

```
[[9 7 9 1 3 6 9 9]
 [4 6 9 1 4 6 9 8]
 [2 4 4 9 2 3 7 7]
 [5 7 6 4 2 3 1 6]
 [1 6 3 3 7 1 2 7]
 [1 7 2 6 4 5 7 8]
 [6 9 7 4 6 1 7 2]
 [1 6 4 8 6 5 9 8]]

résultat après modifications
 [[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
```

Créer un tableau `a` de taille 8x8 (8 lignes et 8 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus, puis mettre la première ligne du tableau à 0.

[24]:
```
# à compléter
```

```
[[0 0 0 0 0 0 0 0]
 [8 2 2 7 1 6 2 8]
 [7 9 8 6 3 5 6 5]
 [4 9 1 3 1 1 5 2]
 [8 4 3 4 2 1 9 7]
 [6 3 8 6 2 5 4 1]
 [8 9 8 5 1 8 7 5]
 [8 3 2 1 5 4 7 9]]
```

Créer un tableau `a` de taille 8x8 (8 lignes et 8 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus puis mettre la colonne numéro 2 du tableau à 0.

[25]:
```
# à compléter
```

```
[[8 1 0 9 6 9 2 2]
 [2 5 0 4 5 1 5 4]
 [1 6 0 3 8 1 7 3]
 [8 5 0 8 8 4 1 8]
 [5 8 0 2 1 2 6 2]
 [2 8 0 8 2 9 6 9]
 [3 4 0 9 4 1 3 9]
 [8 6 0 2 9 1 5 4]]
```

Créer un tableau `a` de taille 8x8 (8 lignes et 8 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus, puis mettre la première et la dernière colonne à 0 (en une seule instruction)

[26]:
```
# à compléter
```

```
[[0 7 2 7 9 8 9 0]
 [0 1 8 1 8 8 8 0]
 [0 5 2 9 8 9 4 0]
 [0 8 3 8 2 2 3 0]
 [0 5 2 8 2 2 9 0]
 [0 9 6 6 7 8 9 0]
 [0 8 7 8 9 5 5 0]
 [0 3 2 1 3 1 8 0]]
```

Créer un tableau `a`de taille 8x8 (8 lignes et 8 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus puis mettre les 9 cases de la partie supérieure gauche du tableau à 0 en une seule instruction.

[27]:
```
# à compléter
```

```
[[0 0 0 9 8 4 2 9]
 [0 0 0 5 8 9 5 2]
 [0 0 0 3 5 4 8 7]
 [9 4 9 1 2 6 1 5]
 [5 9 3 9 8 9 5 2]
 [4 7 5 1 1 4 5 4]
 [4 7 4 6 9 1 2 6]
 [5 2 7 6 8 3 2 3]]
```

Créer un tableau `a` de taille 8x8 (8 lignes et 8 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus puis mettre les cases qui sont sur des lignes **et** des colonnes paires à 0.

[28]:
```
# à compléter
```

```
[[0 6 0 5 0 7 0 2]
 [5 3 6 8 5 6 6 3]
 [0 7 0 3 0 3 0 6]
 [3 4 5 7 5 4 5 4]
 [0 8 0 4 0 8 0 6]
 [5 1 9 9 3 6 9 3]
 [0 6 0 7 0 9 0 1]
 [4 1 4 7 4 8 2 1]]
```

Créer un tableau `a` de taille 8x8 (8 lignes et 8 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus puis mettre les cases $[1, 3]$ et $[3, 2]$ à 0 (en une seule instruction).

[29]:
```
# à compléter
```

```
[[6 4 3 9 5 3 1 1]
 [3 7 4 0 6 9 4 9]
 [2 3 8 3 9 7 8 6]
 [2 4 0 8 3 7 1 6]
 [5 6 3 4 3 3 1 3]
 [7 6 9 4 6 3 2 8]
 [5 3 8 4 9 5 7 7]
 [6 7 5 2 9 4 8 3]]
```

Créer un tableau `a` de taille 8x8 (8 lignes et 8 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus puis mettre à 0 toutes les lignes pour lesquelles le vecteur `vect= np.array([False, True, True, False,False, 'True',False, 'True'])` vaut `True` (en une seule instruction).

[30]:
```
# à compléter
```

```
[False  True  True False False  True False  True]
[[3 1 9 5 9 5 9 5]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [3 5 4 2 7 6 3 6]
 [9 9 5 4 8 6 1 3]
 [0 0 0 0 0 0 0 0]
 [5 2 6 1 6 2 9 1]
 [0 0 0 0 0 0 0 0]]
```

A l'aide de la fonction np.ones, créer un tableau carré de taille 7 ne contenant que des 1. Puis en maximum deux lignes, faire un "damier" en remplaçant certains 1 par des 0 (voir résultat ci-dessous)

[31]:
```
# à compléter
```

```
creation de  a
```

```
 [[1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]]
apres transformation
 [[0. 1. 0. 1. 0. 1. 0.]
 [1. 0. 1. 0. 1. 0. 1.]
 [0. 1. 0. 1. 0. 1. 0.]
 [1. 0. 1. 0. 1. 0. 1.]
 [0. 1. 0. 1. 0. 1. 0.]
 [1. 0. 1. 0. 1. 0. 1.]
 [0. 1. 0. 1. 0. 1. 0.]]
```

0.0.4 4. Partage de valeurs dans un tableau

a) " Vues" (extrait avec des slices) Créer un tableau `a` de taille 4x4 rempli de valeurs aléatoires entières de 1 à 9 inclus que l'on nommera a.

Créer ensuite une vue (appelée `b`) de la matrice `a` correspondant aux sélecteurs précédents en prenant pas ar exemple :

```
b = a[0,:]
```

Modifier le contenu de `b` et regarder l'effet sur `a`. Puis modifier `a` et regarder l'effet sur `b`

Exemple de ce qui est attendu :

```
a= [[6 3 8 1]
 [3 9 4 4]
 [9 5 4 5]
 [4 2 9 8]]
 b= [6 3 8 1]
Après modification de la case 0,3 de a (mise à -1) et la case 0,0 de b (mise à 0)     on obtien
a= [[ 0  3  8 -1]
 [ 3  9  4  4]
 [ 9  5  4  5]
 [ 4  2  9  8]]
 b= [ 0  3  8 -1]
```

[32]:
```
# à compléter
```

```
a= [[5 4 2 4]
 [1 5 4 2]
 [7 3 9 4]
 [4 7 9 2]]
première ligne b= [5 4 2 4]
Après modification de la case 0,3 de a (mise à -1) et la case 0,0 de b (mise à
0) on obtient :
```

```
a= [[ 0  4  2 -1]
 [ 1  5  4  2]
 [ 7  3  9  4]
 [ 4  7  9  2]]
b= [ 0  4  2 -1]
```

b) Copies Reproduire la même expérience, mais cette fois-ci en définissant `b` comme une copie de `a` (et non plus une vue) à l'aide de la fonction copy avec l'instruction a[0,:].copy(). Comparer les résultats.

[33]:
```
# à compléter
```

```
a= [[8 8 9 5]
 [7 5 9 2]
 [4 6 8 9]
 [3 2 3 9]]
b= [8 8 9 5]
Après modification de la case 0,3 de a (mise à -1) et la case 0,0 de b (mise à
0) on obtient :
a= [[ 8  8  9 -1]
 [ 7  5  9  2]
 [ 4  6  8  9]
 [ 3  2  3  9]]
b= [0 8 9 5]
```

0.0.5 5) Tableaux de tailles différentes

Créer un array numpy nommé `a` de taille 3x3 contenant des valeurs aléatoires entières entre 1 et 9 compris.

Créer un second array nommé `b` de taille 1x3 également rempli de valeurs aléatoires entières entre 1 et 9 compris.

Créer un troisième array nommé `c` de taille 3x1 également rempli de valeurs aléatoires entières entre 1 et 9 compris.

[34]:
```
# à compléter
```

```
a = [[9 9 6]
 [9 4 9]
 [2 5 1]]
b = [[4 4 3]]
c= [[1]
 [9]
 [5]]
```

Ajouter la constante 5 à a, afficher le résultat :

[35]:
```
# à compléter
```

```
[[14 14 11]
 [14  9 14]
 [ 7 10  6]]
```

Calculer (a+b). Que représente ce calcul?

Calculer (a+c). Que représente ce calcul?

Afficher b.T

En déduire une façon d'ajouter b à chaque colonne de a

[36]:
```
# à compléter
```

```
a = [[9 9 6]
 [9 4 9]
 [2 5 1]]
b = [[4 4 3]]
c= [[1]
 [9]
 [5]]
a+b [[13 13  9]
 [13  8 12]
 [ 6  9  4]]
a+c [[10 10  7]
 [18 13 18]
 [ 7 10  6]]
b.T [[4]
 [4]
 [3]]
a+b.T [[13 13 10]
 [13  8 13]
 [ 5  8  4]]
a = [[9 9 6]
 [9 4 9]
 [2 5 1]]
b = [[4 4 3]]
c= [[1]
 [9]
 [5]]
```

Calculer la somme b+c, afficher le résultat et le commenter.

[37]:
```
# à compléter
```

```
b:
 [[4 4 3]]
c:
 [[1]
 [9]
```

```
 [5]]
b+c:
 [[ 5  5  4]
 [13 13 12]
 [ 9  9  8]]
```