TP2_correction

January 29, 2020

1 TP2 - Prise en main de numpy

1.1 Premiers pas avec numpy

1.1.1 Création de tableaux

Pour commencer, importer (avec import) dans le bloc de code qui suit, la librairie numpy

```
In [2]: import numpy as np
```

Créer, à partir du tableau python t définie ci-dessous, une variable numpy de type array, nommée a contenant les valeurs du tableau, puis afficher le contenu de a et son type

```
In [3]: t = [12, 2, 3, 8, 1]
        a = np.array(t)
        print('a:',a, 'type :',type(a))
```

```
a: [12  2  3  8  1] type : <class 'numpy.ndarray'>
```

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

```
In [4]: s = np.sum(a)
        print('somme : ',s)
```

```
somme :  26
```

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

```
In [5]: print('ndim=',a.ndim,'shape=',a.shape,'size=',a.size)
```

```
ndim= 1 shape= (5,) size= 5
```

Calculer la moyenne (utiliser pour cela la fonction np.sum()) et la mediane de a, puis la valeur min et la valeur max du tableau, en utilisant uniquement les opérateurs +, / et l'opérateur de tri de numpy (np.sort()). Ne pas utiliser de boucle.

```
In [6]: print('moyenne :',np.sum(a)/a.shape[0],'mediane :', np.sort(a)[int(a.shape[0]/2)])
        print('val max :',np.sort(a)[-1],'val min :', np.sort(a)[0])

moyenne : 5.2 mediane : 3
val max : 12 val min : 1
```

1.1.2 Opérations vectorisées

Calculer le tableau numpy a2 contenant le carré des valeurs de a, sans faire de boucle.

```
In [7]: a2 = a**2
        print('t2 : ',a2)

t2 :  [144   4   9  64   1]
```

Calculer le tableau a3 contenant la somme terme à terme des éléments de a et de a2.

```
In [8]: a3=a+a2
        print('a3 : ',a3)

a3 :  [156   6  12  72   2]
```

1.2 Indexation, agrégation et broadcasting des objets `array` de numpy

Créer un tableau a de taille 4x3 (4 lignes et 3 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus. Vous utiliserez pour cela la fonction `np.random.randint`. Afficher son contenu.

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

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

```
In [4]: import numpy as np
        a = np.random.randint(1,10,(4,3))
        print(a)

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

1.2.1 Agrégation

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 carrés des valeurs
le nombre de valeurs supérieures 5
la somme des valeurs supérieures 5
```

Exemple de résultat attendu :
Tableau a [[8 5 2 8][2 3 9 2] [1 7 2 2]]
somme des valeurs : 51
somme par colonne : [11 15 13 12]
somme par ligne : [23 16 12]
somme des carrés : 313
nombre de valeurs supérieures à 5 : 4
somme des valeurs supérieures à 5 : 32

```
In [5]: a = np.random.randint(1,10,(3,4))
        print('Tableau a \n',a)
```

```
Tableau a
 [[1 1 3 4]
 [9 1 7 1]
 [5 3 1 7]]
```

```
In [6]: print('somme des valeurs : ', np.sum(a))
        print('moyenne des valeurs : ', np.mean(a))
```

```
somme des valeurs :  43
moyenne des valeurs :  3.5833333333333335
```

```
In [7]: print('somme par ligne : ', np.sum(a,axis=0))
```

```
somme par ligne :  [15  5 11 12]
```

```
In [8]: print('somme par colonnes : ', np.sum(a,axis=1))
```

```
somme par colonnes :  [ 9 18 16]
```

```
In [9]: print('somme des carrés : ', np.sum(a**2))
```

```
somme des carrés :  243
```

```
In [10]: print('nombre de valeurs supérieures à 5 : ', np.sum(a>5))

nombre de valeurs supérieures à 5 :  3


In [11]: print('somme des valeurs supérieures à 5 : ', np.sum(a[a>5]))

somme des valeurs supérieures à 5 :  23
```

1.2.2 Indexation

Afficher les valeurs de a3 dont les indices sont pairs

```
In [9]: print('val indices pairs : ',a3[0::2])

val indices pairs :  [156  12   2]
```

Afficher la somme des valeurs de a3 dont les indices sont pairs

```
In [10]: print('val indices pairs : ',np.sum(a3[0::2]))

val indices pairs :  170
```

Créer un tableau a de taille 4x4 (4 lignes et 4 colonnes) rempli de valeurs aléatoires entières de 1 à 9 inclus.

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.

```
In [5]: import numpy as np
        a = np.random.randint(1,10,(4,4))
        print(a)
        a[:,:] = 0
        print("\n résultat après modifications\n",a)

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

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

Créer un tableau numpy de dimension 4x4 contenant des valeurs entières aléatoires puis mettre la première ligne du tableau à 0. Le résultat attendu est :

```
[[0 0 0 0]
 [1 8 3 1]
 [4 7 7 7]
 [6 3 3 3]]
```

```
In [13]: a = np.random.randint(1,10,(4,4))
         a[0,:]=0
         print(a)
```

```
[[0 0 0 0]
 [1 2 2 1]
 [3 1 9 2]
 [6 1 2 8]]
```

Créer un tableau numpy de dimension 4x4 contenant des valeurs aléatoires puis mettre la colonne numéro 2 du tableau à 0. Le résultat attendu est :

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

```
In [14]: a = np.random.randint(1,10,(4,4))
         a[:,2]=0
         print(a)
```

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

Créer un tableau numpy de dimension 4x4 contenant des valeurs aléatoires puis mettre la première et la dernière colonne à 0 (en une seule instruction). Le résultat attendu est :

```
[[0 9 0 3]
[0 5 0 3]
[0 6 0 9]
[0 6 0 2]]
```

```
In [3]: a = np.random.randint(1,10,(4,4))
        a[:,[0, -1]]=0
        print(a)
```

```
[[0 3 2 0]
 [0 3 1 0]
 [0 1 3 0]
 [0 6 5 0]]
```

Créer un tableau numpy de dimension 4x4 contenant des valeurs aléatoires puis mettre les 9 cases de la partie supérieure gauche du tableau à 0 en une seule instruction. Le résultat attendu est :

```
[[0 0 0 1]
[0 0 0 7]
[0 0 0 8]
[7 2 9 2]]
```

```
In [15]: a = np.random.randint(1,10,(4,4))
         a[0:3,0:3]=0
         print(a)
```

```
[[0 0 0 4]
 [0 0 0 5]
 [0 0 0 4]
 [4 6 2 5]]
```

Créer un tableau numpy de dimension 4x4 contenant des valeurs aléatoires puis mettre les cases qui sont sur des lignes et des colonnes paires à 0. Le résultat attendu est :

```
[[0 6 0 7]
 [2 8 2 7]
 [0 3 0 3]
 [1 1 3 7]]
```

```
In [16]: a = np.random.randint(1,10,(4,4))
         a[::2,::2]=0
         print(a)
```

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

Créer un tableau numpy de dimension 4x4 contenant des valeurs aléatoires puis mettre les cases $[1,3]$ et $[3,2]$ à 0 (en une seule instruction). Le résultat attendu est :

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

```
In [18]: a = np.random.randint(1,10,(4,4))
         a[[1,3],[3,2]]=0
         print(a)
```

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

Créer un tableau numpy de dimension 4x4 contenant des valeurs aléatoires, puis mettre à 0 toutes les lignes pour lesquelles le vecteur `i = np.array([False, True, True, False])` vaut `True` (en une seule instruction). Le résultat attendu est :

```
[[3 1 6 7]
[0 0 0 0]
[0 0 0 0]
[5 9 7 3]]
```

```
In [19]: a = np.random.randint(1,10,(4,4))
         i = np.array([False, True, True, False])
         print(i)
         a[i,:]=0
         print(a)
```

```
[False  True  True False]
[[6 5 8 3]
 [0 0 0 0]
 [0 0 0 0]
 [6 5 1 5]]
```

Dans cette question, il est demandé de créer une vue (appelée `b`) de la matrice `a` correspondant aux sélecteurs précédents. Par exemple :

```
b = a[0,:] (a est défini dans la première question)
```

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 obtier
a= [[ 0  3  8 -1]
 [ 3  9  4  4]
 [ 9  5  4  5]
 [ 4  2  9  8]]
 b= [ 0  3  8 -1]
```

```
In [20]: a = np.random.randint(1,10,(4,4))
         b = a[0,:]
         print('a=',a,'\n b=',b)
         b[0] = 0
         a[0,3] =  -1
         print("Après modification de la case 0,3 de a (mise à -1) et la case 0,0 de b (mise à
         print('a=',a,'\n b=',b)

a= [[1 5 8 4]
 [9 1 8 8]
 [8 9 8 5]
 [4 4 7 7]]
 b= [1 5 8 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  5  8 -1]
 [ 9  1  8  8]
 [ 8  9  8  5]
 [ 4  4  7  7]]
 b= [ 0  5  8 -1]
```

Reproduire la même expérience, mais cette fois-ci en définissant `b` comme une copie de `a` (et non plus une vue). Commenter le résultat.

```
In [21]: a = np.random.randint(1,10,(4,4))
         b = a[0,:].copy()
         print('a=',a,'\n b=',b)
         b[0] = 0
         a[0,3] =  -1
         print("Après modification de la case 0,3 de a (mise à -1) et la case 0,0 de b (mise à
         print('a=',a,'\n b=',b)

a= [[9 8 6 7]
 [6 8 7 6]
 [4 3 1 1]
 [1 2 5 9]]
 b= [9 8 6 7]
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= [[ 9  8  6 -1]
 [ 6  8  7  6]
 [ 4  3  1  1]
 [ 1  2  5  9]]
 b= [0 8 6 7]
```

1.2.3 Broadcasting

Créer un array numpy nommé `a` de taille 3x3 contenant des valeurs aléatoires entières entre 1 et 9 comprises. Créer un second array nommé `b` de taille 1x3 également rempli de valeurs aléatoires entières entre 1 et 9 comprises.

```
In [29]: a = np.random.randint(1,10,(3,3))
         b = np.random.randint(1,10,(1,3))
         print('a =',a,'\n b =',b)
```

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

Ajouter la constante 5 à `a`, afficher le résultat :

```
In [30]: print(a+5)
```

```
[[12  9  9]
 [ 8  6  7]
 [ 7  7 12]]
```

Ajouter `b` à chacune des lignes de `a`, afficher le résultat :

```
In [31]: print(a+b)
```

```
[[10  6 13]
 [ 6  3 11]
 [ 5  4 16]]
```

Ajouter `b` à chacune des colonnes de `a`, afficher le résultat :

```
In [32]: print(a+b.T)
```

```
[[10  7  7]
 [ 5  3  4]
 [11 11 16]]
```

Soit `c` un array de taille 3x1 contant des valeurs aléatoires entières entre 1 et 9 compris. Calculer la somme `b+c`, afficher le résultat et le commenter.

```
In [33]: c = np.random.randint(1,10,(3,1))
         print('\n b: \n',b,'\n c:\n ',c,'\n b+c:\n ',b+c)
```

```
b:
[[3 2 9]]
c:
 [[6]
[4]
[4]]
b+c:
 [[ 9  8 15]
[ 7  6 13]
[ 7  6 13]]
```