TP1 : NumPy Part 1

Comment créer un tableau numpy ?

On peut créer un tableau numpy en utilisant la fonction np.array() ,avec une liste ou un objet similaire à une liste.

Créer un tableau 1D à partir d'une liste.

```
import numpy as np
list1 = [0,1,2,3,4]
arr1d = np.array(list1)
```

Imprimer le tableau et son type.

```
print(type(arr1d))#> class 'numpy.ndarray'
arr1d #> array([0, 1, 2, 3, 4])
```

Si vous voulez ajouter 2 à chaque élément de la liste, vous pourriez simplement le formuler ainsi :

```
list1 + 2 # error
```

Cela n'était pas possible avec une liste. Mais vous pouvez le faire avec un tableau np.array.

```
arr1d + 2 #array([2, 3, 4, 5, 6])
```

Vous pouvez utiliser une liste de listes pour créer un tableau 2D.

```
list2 = [[0,1,2], [3,4,5], [6,7,8]]
arr2d = np.array(list2)
arr2d """ array([[0, 1, 2],
                 [3, 4, 5],
                 [6, 7, 8]])"""
```

Vous pouvez préciser le type de données avec l'argument dtype. Les types couramment utilisés sont 'float', 'int', 'bool', 'str' et 'object', pour un contrôle plus fin de la mémoire.

```
arr2d_f = np.array(list2, dtype='float')
arr2d_f """array([[ 0., 1., 2.],
                  [ 3., 4., 5.],
                  [ 6., 7., 8.]]"""
```

Vous pouvez également le convertir en un type de données différent en utilisant la méthode .astype().

```
# Convert to int then to str datatype
arr2d_f.astype('int').astype('str')
"""array([['0', '1', '2'],
          ['3', '4', '5'],
          ['6', '7', '8']],
            dtype='U21')"""
```

Un tableau numpy doit avoir des éléments du même type, contrairement aux listes. Vous pouvez définir le type de données comme 'object'.

```
arr1d_obj = np.array([1, 'a'], dtype='object')
arr1d_obj #array([1, 'a'], dtype=object)
```

Vous pouvez facilement convertir un tableau en une liste Python en utilisant la méthode tolist().

```
arr1d_obj.tolist() # [1, 'a']
```

Comment inspecter la taille et la forme d'un tableau numpy ?

- Pour déterminer s'il s'agit d'un tableau 1D, 2D ou plus, utilisez ndim.
- Pour connaître le nombre d'éléments dans chaque dimension, utilisez shape.
- Pour identifier le type de données, utilisez dtype.
- Pour connaître le nombre total d'éléments, utilisez size.
- Pour voir des échantillons des premiers éléments du tableau, utilisez l'indexation.

```
list2 = [[1, 2, 3, 4],[3, 4, 5, 6], [5, 6, 7, 8]]
arr2 = np.array(list2, dtype='float')
arr2
"""array([[ 1., 2., 3., 4.],
          [ 3., 4., 5., 6.],
          [ 5., 6., 7., 8.]])"""
# shape
print('Shape: ', arr2.shape) # Shape:  (3, 4)
#dtype
print('Datatype: ', arr2.dtype)# Datatype:  float64
# size
print('Size: ', arr2.size)# Size:  12
# ndim
print('Num Dimensions: ', arr2.ndim)# Num Dimensions:  2
```

Comment extraire des éléments spécifiques d'un tableau ?

```
arr2
""" array([[ 1., 2., 3., 4.],
           [ 3., 4., 5., 6.],
           [ 5., 6., 7., 8.]])"""
arr2[:2, :2]""" array([[ 1., 2.],
                       [ 3., 4.]])"""
list2[:2, :2] """ error """
```

Obtenez la sortie booleen en appliquant la condition à chaque élément :

```
b = arr2 > 4
b"""array([[False, False, False, False],
           [False, False, True, True],
           [ True, True, True, True]], dtype=bool)"""
arr2[b] #array([ 5., 6., 5., 6., 7., 8.])
```

- Comment inverser les lignes et l'ensemble d'un tableau numpy ?
  Inverser uniquement les positions des lignes : `arr2[::-1, ]`

  Inverser les positions des lignes et des colonnes : `arr2[::-1, ::-1]`

- Comment représenter les valeurs manquantes et l'infini
  Insérer un nan (Not a Number) et un infini dans le tableau.

```
""" np.nan représente les valeurs manquantes
    np.inf représente l'infini                """
arr2[1,1] = np.nan # not a number
arr2[1,2] = np.inf # infinite
arr2
"""array([[ 1., 2., 3., 4.],
          [ 3., nan, inf, 6.],
          [ 5., 6., 7., 8.]])"""
```

  Remplacer NaN et l'infini par -1 sans utiliser arr2 == np.nan.

```
missing_bool = np.isnan(arr2) | np.isinf(arr2)
arr2[missing_bool] = -1
arr2
""" array([[ 1., 2., 3., 4.],
           [ 3., -1., -1., 6.],
           [ 5., 6., 7., 8.]])"""
```

- Comment calculer la moyenne, le minimum et le maximum sur le ndarray ?

```
print("Mean value is: ", arr2.mean())# Mean value is: 3.58333333333
print("Max value is: ", arr2.max())#  Max value is: 8.0
print("Min value is: ", arr2.min())# Min value is: -1.0
```

```
# Row wise and column wise min
print("Column wise minimum: ", np.amin(arr2, axis=0))
#> Column wise minimum: [ 1. -1. -1. 4.]
print("Row wise minimum: ", np.amin(arr2, axis=1))
#> Row wise minimum: [ 1. -1. 5.]
```

```
# Cumulative Sum
np.cumsum(arr2)# array([1.,3.,6.,10.,13.,12.,11.,17.,22.,28.,35.,43.])
```

Comment créer un nouveau tableau à partir d'un tableau existant ?

Attribuer une partie de arr2 à arr2a. Ne crée pas vraiment un nouveau tableau.

```
arr2a = arr2[:2,:2]
arr2a[:1, :1] = 100
arr2""" array([[ 100., 2., 3., 4.],
               [ 3., -1., -1., 6.]
               [ 5., 6., 7., 8.]])"""
```

Copier une partie de arr2 dans arr2b.

```
arr2b = arr2[:2, :2].copy()
arr2b[:1, :1] = 101
arr2
"""array([[100.,2.,3.,4.],[3.,-1.,-1.,6.],[5.,6.,7.,8.]])"""
```

Remodelage et aplatissement des tableaux multidimensionnels

Remanions le tableau arr2 d'une forme 3×4 à une forme 4×3.

```
arr2.reshape(4, 3)
""" array([[ 100., 2., 3.],
           [ 4., 3., -1.],
           [ -1., 6., 5.],
           [ 6., 7., 8.]])"""
```

- Quelle est la différence entre `flatten()` et `ravel()` ?
  Il existe deux méthodes populaires pour mettre à plat (flatten) un tableau, en utilisant la méthode `flatten()` et l'autre en utilisant la méthode `ravel()`.

  - Aplatir le tableau pour en faire un tableau 1D : `arr2.flatten()`

  - Changer le tableau aplati ne modifie pas le tableau d'origine :

```
b2 = arr2.ravel()
b2[0] = 101 # Changer b2 modifie également arr2.
```

Comment créer des séquences, des répétitions et des nombres aléatoires avec numpy ?

- La limite inférieure est par défaut fixée à 0.

```
print(np.arange(5))#[0 1 2 3 4]
# 0 à 9
print(np.arange(0, 10))#[0 1 2 3 4 5 6 7 8 9]
# 0 à 9 pas=2
print(np.arange(0, 10, 2))#[0 2 4 6 8]
# 10 à 1
print(np.arange(10, 0, -1))#[10 9 8 7 6 5 4 3 2 1]
```

- Créer un tableau exactement avec 10 nombres entre 1 et 50

```
np.linspace(start=1, stop=50, num=10, dtype=int)#array([1 6 11 17 22 28 33 39 44 50])
```

- Limitez le nombre de chiffres après la virgule à 2.
  np.set_printoptions(precision=2)
- Créer un tableau exactement avec 10 commencez à 10^1 et terminez à 10^50.

```
np.logspace(start=1, stop=50, num=10, base=10)
#array([1.00e+01 2.78e+06 7.74e+11 2.15e+17 5.99e+22 1.67e+28 4.64e+33 1.29e+39 3.59e+44 1.00e+50]
```

- Les fonctions `np.zeros` et `np.ones` vous permettent de créer des tableaux de la forme souhaitée où tous les éléments sont soit des 0 ou des 1.

```
np.zeros([2,2])#array([[ 0., 0.],[ 0., 0.]])
np.ones([2,2])#array([[ 1., 1.],[ 1., 1.]])
```

- Comment créer des séquences répétitives ?

Répétez l'intégralité de 'a' deux fois :

```
a = [1,2,3]
print('Tile: ', np.tile(a, 2))#Tile:  [1 2 3 1 2 3]
```

Répétez chaque élément de 'a' deux fois :

```
print('Repeat: ', np.repeat(a, 2))#Repeat:  [1 1 2 2 3 3]
```

- Comment générer des nombres aléatoires ?

```
import numpy as np
np.set_printoptions(precision=2)
"""nombres aléatoires entre [0,1) de forme 2x2."""
print(np.random.rand(2,2))
"""distribution normale, moyenne 0, variance 1, forme 2x2.'
print(np.random.randn(2,2))
"""entiers aléatoires entre [0, 10) de forme 2x2."""
print(np.random.randint(0, 10, size=[2,2]))
"""un nombre aléatoire entre [0,1)."""
print(np.random.random())
"""nombres aléatoires entre [0,1) de forme 2x2."""
print(np.random.random(size=[2,2]))
```

```
[[0.31 0.92]
 [0.05 0.95]]
[[ 1.54 -0.29]
 [-1.65 -0.18]]
[[2 2]
 [6 7]]
0.042948431723431
[[0.93 0.79]
 [0.12 0.53]]
```

```
"""choisir 10 éléments d'une liste donnée"""
print(np.random.choice(['a', 'e', 'i', 'o', 'u'], size=10))
"""pick 10 items from a given list with a predefined probability 'p' """
print(np.random.choice(['a', 'e', 'i', 'o', 'u'], size=10,
p=[0.3, .1, 0.1, 0.4, 0.1]))
```

```
['a' 'u' 'e' 'u' 'u' 'i' 'u' 'e' 'a' 'u']
['a' 'o' 'a' 'o' 'o' 'o' 'a' 'a' 'i' 'o']
```

Créer l'état aléatoire (Random State) :

rn = np.random.RandomState(100)

Créer des nombres aléatoires entre [0,1) de forme 2x2 :

print(rn.rand(2,2))

```
[[0.54 0.28]
 [0.42 0.84]]
```

Définir la graine aléatoire (random seed) :

np.random.seed(100)

Créer des nombres aléatoires entre [0,1) de forme 2x2 :

print(np.random.rand(2,2))

```
[[0.54 0.28]
 [0.42 0.84]]
```

Comment obtenir les éléments uniques et leurs fréquences ?

Créer les entiers aléatoires de taille 10 entre[0,10) :

```
np.random.seed(100)
arr_rand = np.random.randint(0, 10, size=10)
print(arr_rand)#[8 8 3 7 7 0 4 2 5 2]
```

obtenir les objets uniques et leur nombre :

```
uniqs, counts = np.unique(arr_rand, return_counts=True)
print("Unique items : ", uniqs)# Unique items :  [0 2 3 4 5 7 8]
print("Counts : ", counts)# Counts :  [1 2 1 1 1 2 2]
```