TP : Numpy

La bibliothèque NumPy (http://www.numpy.org/) permet d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres.

Il faut au départ importer le package « numpy » avec l'instruction suivante :

```
>>> import numpy as np
```

Les tableaux peuvent être créés avec « **numpy.array()** ». On utilise des crochets pour délimiter les listes d'éléments dans les tableaux comme suit et peut accéder à un élément en utilisant a[indice] :

```
>>> a = np.array([1, 2, 3, 4])
```

De même pour les tableaux à deux dimensions :

```
>>> b = np.array([[1, 2, 3], [4, 5, 6]])
```

La fonction « **numpy.arange([start,] stop[, step,][, dtype])** » permet de créer également un tableau avec les spécifications suivantes :

- start : [facultatif] début de la plage d'intervalle. Par défaut début = 0
- stop : fin de plage d'intervalle
- step : [optionnel] taille du pas de l'intervalle. Par défaut pas de taille = 1
- dtype : type de tableau de sortie

La fonction « **numpy.linspace(start, stop, nbrElements)** » permet d'obtenir un tableau 1D allant d'une valeur de départ à une valeur de fin avec un nombre donné d'éléments.

NumPy dispose d'un grand nombre de fonctions mathématiques qui peuvent être appliquées directement à un tableau. Dans ce cas, la fonction est appliquée à chacun des éléments du tableau.

Travail à faire :

1. Partie 1 : Création de listes Python
    a. Créer deux listes tab1 et tab2. « tab1 » est une liste à une dimension qui contient les chiffres de 1 à 10. « tab 2 » est une liste à deux dimensions (5,3) qui contient les chiffres de 1 à 15.
    b. Afficher la 2ème ligne de « tab2 ».
    c. Afficher la 2ème colonne de « tab2 ».

2. Partie 2 : Création d'un objet ndarray
    a. Installez le Module Numpy si vous ne l'avez pas et importer le.
    b. Créer un tableau à une dimension qui contient les chiffres de 1 à 6 en utilisant numpy.
    c. Afficher le type des éléments du tableau et changer ce type en float.
    d. Obtenez maintenant la géométrie de ce tableau avec shape.
    e. Changer la forme de ce tableau en un objet 2-D (de 2x3 éléments).
    f. Afficher la 2ème ligne de ce tableau.
    g. Afficher la 2ème colonne de ce tableau.

3. Partie 3 : Les outils de numpy
    a. Créer une matrice de taille 5x5 qui ne contient que des zéros.
    b. Créer une matrice de taille 5x5 qui ne contient que des uns.
    c. Créer une matrice de taille 5x5 qui ne contient que des cinq.
    d. Créer un tableau qui contient les chiffres de 0 à 100.
    e. Il nous arrive d'avoir besoin de créer un objet, sans connaître à l'avance le pas, mais en sachant le nombre de valeurs souhaitées. Créer un tableau avec 30 valeurs allant de pi à 2*pi.
    f. Créez une matrice 3x5 comprenant des entiers aléatoires entre 1 et 10 (compris).

4. Partie 4 : Manipulations d'objets ndarray
    a. Créer le tableau « temp » contenant : array([[0, 1, 2], [3, 4, 5]]).
    b. Depuis la matrice « temp » créer la matrice « m » array([[0, 1, 2, 0, 1, 2, 0, 1, 2], [3, 4, 5, 3, 4, 5, 3, 4, 5]]).
    c. Recréez l'objet « m » contenant :
    array([[0, 1, 2], [3, 4, 5], [0, 1, 2], [3, 4, 5], [0, 1, 2], [3, 4, 5]])
    d. Créez « m2 » à partir de « m », tel que chaque valeur est dupliquée selon le 2ème axe et obtenez ceci : array([[0, 0, 1, 1, 2, 2],
    [3, 3, 4, 4, 5, 5],
    [0, 0, 1, 1, 2, 2],
    [3, 3, 4, 4, 5, 5],
    [0, 0, 1, 1, 2, 2],
    [3, 3, 4, 4, 5, 5]]).
    e. Découper « m2 » verticalement en 2 matrices de mêmes dimensions.

5. Partie 5 : Opérations et tests
    a. Définissez 2 objets tels que : a = np.array([[4, 3, 2], [1, 0, -1], [-2, -3, -4]]) et b = np.array([10, 20, 30]).
    b. Ajouter 10 à valeurs de l'objet a.
    c. Si maintenant vous additionnez les 2 objets a et b. Qu'allez-vous obtenir ? (Répondez à la 2ème partie de la question sous forme d'un commentaire).
    d. Même question que « c » mais cette fois avec le transposée de l'objet « b ». Mais attention, l'opération de transposée pour un objet 1-D (3,) ne fait rien. Il faut tout d'abord le rendre en (3,1).

e. Essayez à présent le produit matriciel b @ a. Est-ce que le résultat est celui que vous attendez? (Répondez à la 2ème partie de la question sous forme d'un commentaire).
f. Essayez maintenant le produit matriciel a @ b. Est-ce que le résultat est celui que vous attendez? (Répondez à la 2ème partie de la question sous forme d'un commentaire).
g. Tapez la commande suivante « test = a > 0 » et afficher « test » que fait cette commande (Réponse en commentaire).
h. Obtenez une matrice de dimension 3x3 « paire » contenant dans chaque cellule paire[i] soit True ou False suivant si a[i] est respectivement paire ou impaire.
i. Avec l'outil np.all, nous pouvons vérifier que toutes les valeurs de l'objet « test » sont vraies (True).
j. Avec l'outil np.all, nous pouvons vérifier que toutes les valeurs de la 2ème ligne de l'objet « test » sont vraies (True).

6. Partie 5 : Indexing and Slicing

a. créez une nouvelle matrice 3-D mat3d, de géométrie (2, 3, 4) avec les valeurs séquentielles de 0 à 23.
b. Récupérez maintenant la valeur 14 de cette matrice.
c. Avec cela, récupérez l'avant dernière valeur de cette matrice.
d. Extrayez de mat3d la vue sur les données suivantes : array([5, 6]).
Note: Avec Numpy, le résultat du slicing est une vue sur la matrice originale. Cela signifie que les données ne sont pas à nouveau copiées en mémoire, mais juste pointées par le nouvel objet. Pour cette raison, si l'on modifie l'un des 2 objets, l'autre sera également modifié en conséquence! Si l'on souhaite séparer les 2 objets, il faudra alors utiliser np.copy().
e. Extrayez à présent de mat3d, la dernière ligne de chaque couche, ce qui donnera : array([[8, 9, 10, 11],
[20, 21, 22, 23]]).
f. Faites encore une extraction. Récupérez les 3 dernières valeurs des 2 premières lignes de toutes les couches de l'objet mat3d. Cela doit donner :
array([[[1, 2, 3],
[5, 6, 7]],

[[13, 14, 15],
[17, 18, 19]]])
g. Mettant à 50 la première colonne de chaque couche de mat3d. Puis écrasez tous les multiples de 3 avec la valeur 0
array([[[50, 1, 2, 0],
[50, 5, 0, 7],
[50, 0, 10, 11]],
[[50, 13, 14, 0],
[50, 17, 0, 19],
[50, 0, 22, 23]]]).