
TP1 - Introduction à NumPy

Pour l'ensemble des TP vous pourrez utiliser au choix l'EDI PyCharm ou le docker Jupyter notebook. Pour lancer les éditeurs :

- Ouvrir un terminal,
- `pycharm-community` ou `_jupyter_notebook`
- Créer un nouveau projet pour chaque TP et un fichier pour chaque exercice.

ATTENTION : Chaque TP fera l'objet d'un dépôt sur Moodle qui sera noté à chaque séance.

Date limite de dépôt pour le TP1 **jeudi 6 Février 2020, 23h59**.

Une documentation Python est disponible à cette adresse : <https://docs.python.org/3/tutorial/>

Une documentation NumPy est disponible à cette adresse : <https://docs.scipy.org/doc/numpy/reference/>

Une documentation Matplotlib : <https://matplotlib.org/3.1.1/api/index.html>

Exercice 1 - Utilisation des listes en Python

1. Construire une liste `liste1` qui contient les nombres de 1 à 99 (avec la fonction `range`).
2. Ajouter à l'aide d'une boucle les éléments de 100 à 199, à une liste vide `liste2`.
3. Ajouter la liste `liste2` à la liste `liste1`.
4. Inverser `liste1` pour qu'elle aille de 199 à 1.
5. Afficher `liste1` dans l'ordre croissant (2 possibilités en modifiant la liste et sans la modifier).
6. Avec la fonction `range()`, construire une `liste3` composée de tous les nombres pairs de 0 à 100 (inclus).
7. À l'aide d'une boucle supprimez tous les multiples de 10 de `liste3`.
8. Affichez la longueur de `liste3`
9. Affichez tous les multiples de 8 de `liste3`
10. Affichez les 20 premiers éléments de `liste3`
11. Affichez les éléments suivants les 20 premiers éléments de `liste3`
12. Affichez les 10 derniers éléments de `liste3`

Python propose la compréhension de listes permettant de construire une liste en une seule ligne avec la forme :

```
newList = [valeur for element in liste if condition]
```

La nouvelle liste sera composée des éléments `valeur` si la condition est respectée.

13. Créer une `liste4` composée uniquement des nombres impairs en une ligne
14. Construire une `liste5` comprenant la suite de fibonacci avec une compréhension de liste en initialisant les deux premières valeurs de la liste à 1

Il est aussi possible d'appliquer une fonction à une liste à l'aide de fonctions de callback. Une fonction de callback a pour forme :

```
def mafonctioncallback(callback, liste):  
    nouvelle_liste = []  
    for element in liste:
```

```

        [action a realiser sur liste a l'aide de la fonction callback]
    return nouvelle_liste

def mafonction(x):
    return [calcul_sur_x]

```

15. Écrire une fonction de callback qui permet d'effectuer un map (appliquer le carré sur une liste et appliquer un test de parité (renvoie une liste contenant True ou False pour chaque élément))
16. Écrire une fonction de callback qui permet d'effectuer un filtre (retourne une liste contenant uniquement des nombres pairs)
17. Écrire une fonction de callback qui permet d'effectuer un reduce (retourner la somme des éléments, retourner le produit des éléments)
18. Écrire une fonction minMaxMoy() qui reçoit une liste d'entiers et qui renvoie le minimum, le maximum et la moyenne de cette liste. Le programme principal appellera cette fonction avec la liste : [10, 18, 14, 20, 12, 16] et sauvegardera ces trois résultats dans un fichier "minMaxMoy.txt"
19. Créer une liste6 avec les valeurs [1,2,3] puis créer une copie de liste6 dans une liste6bis (attention si on change une valeur de liste6bis, la liste6 ne doit pas être modifiée)

Exercice 2 - Utilisation des tableaux NumPy

NumPy est une bibliothèque qui propose des traitements efficaces sur les tableaux de nombres. Il est possible d'utiliser des tableaux d'objets mais cela réduit l'efficacité des fonctions intégrées à NumPy.

Tableaux à une dimension

1. Importer la librairie NumPy.
2. Créer un numpy array vide de dimension 1 et de taille 10.
3. Créer un numpy array de dimension 1 et de taille 10 contenant toujours la valeur 0.
4. Créer un numpy array de dimension 1 et de taille 10 contenant toujours la valeur 1.
5. Créer un numpy array de dimension 1 et de taille 10 contenant toujours la valeur 5 (2 méthodes).
6. Créer un numpy array de dimension 1 contenant les 10 premiers entiers au format chaîne de caractères.
7. Créer un numpy array tab1 de dimension 1 contenant tous les nombres de 1 à 10 avec un pas de 0,5.
8. Créer un numpy array tab2 de dimension 1 contenant 20 nombres de 1 à 10 (à intervalle régulier).
9. Créer un numpy array tab3 de dimension 1 avec les entiers de 10 à 30.
10. Calculer la somme des entiers de tab3 (une seule ligne).
11. Calculer la somme cumulée des entiers de tab3 (une seule ligne).
12. Calculer la moyenne des entiers de tab3 (une seule ligne).
13. Effectuer le produit scalaire des vecteurs tab2 et tab3.
14. Sélectionner les éléments de tab3 dont la valeur est supérieure à 20.
15. Sélectionner les éléments 1, 5, 10 et 15 de tab3.

Les matrices

1. Générer une matrice identité de taille 9.
2. Créer un numpy array de dimension 2 et de taille 10×10 contenant toujours la valeur 1.
3. Afficher la taille de la matrice précédente.
4. Afficher la dimension de la matrice précédente.
5. Afficher la forme de la matrice précédente.
6. Créer un numpy array de dimension 1 avec les valeurs de 0 à 8 et le transformer en matrice 3 × 3
7. Créer la matrice A suivante :

$$\begin{pmatrix} 0.01 & 0.02 & 0.03 & 0.04 & 0.05 & 0.06 & 0.07 & 0.08 & 0.09 & 0.1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \\ 0.31 & 0.32 & 0.33 & 0.34 & 0.35 & 0.36 & 0.37 & 0.38 & 0.39 & 0.4 \\ 0.41 & 0.42 & 0.43 & 0.44 & 0.45 & 0.46 & 0.47 & 0.48 & 0.49 & 0.5 \\ 0.51 & 0.52 & 0.53 & 0.54 & 0.55 & 0.56 & 0.57 & 0.58 & 0.59 & 0.6 \\ 0.61 & 0.62 & 0.63 & 0.64 & 0.65 & 0.66 & 0.67 & 0.68 & 0.69 & 0.7 \\ 0.71 & 0.72 & 0.73 & 0.74 & 0.75 & 0.76 & 0.77 & 0.78 & 0.79 & 0.8 \\ 0.81 & 0.82 & 0.83 & 0.84 & 0.85 & 0.86 & 0.87 & 0.88 & 0.89 & 0.9 \\ 0.91 & 0.92 & 0.93 & 0.94 & 0.95 & 0.96 & 0.97 & 0.98 & 0.99 & 1.0 \end{pmatrix}$$

8. Extraire la matrice B suivante de la matrice A précédente :

$$\begin{pmatrix} 0.61 & 0.62 & 0.63 & 0.64 \\ 0.71 & 0.72 & 0.73 & 0.74 \\ 0.81 & 0.82 & 0.83 & 0.84 \\ 0.91 & 0.92 & 0.93 & 0.94 \end{pmatrix}$$

9. Générer une matrice de 1 de la taille de la matrice B.

10. Extraire la matrice C suivante de la matrice A :

$$\begin{pmatrix} 0.21 & 0.23 & 0.25 & 0.27 \\ 0.31 & 0.33 & 0.35 & 0.37 \end{pmatrix}$$

11. Calculer le produit matriciel entre la matrice C et la matrice B.

12. Récupérer la valeur 0.55 dans la matrice A

13. Récupérer la 4ème colonne de la matrice A

14. Récupérer la 4ème ligne de la matrice A

15. Récupérer la matrice A en ne gardant que les lignes telles que la valeur de la colonne 4 est supérieure à 0.5

16. Calculer la moyenne globale de cette matrice A.

17. Calculer la moyenne de chaque ligne de la matrice A.

18. Calculer la moyenne de chaque colonne de la matrice A.

19. Calculer le produit scalaire de la première colonne de la matrice A avec la première ligne de la matrice A.

20. Remplacer le centre de la matrice par la valeur 0 comme l'exemple suivant :

$$\begin{pmatrix} 0.01 & 0.02 & 0.03 & 0.04 & 0.05 & 0.06 & 0.07 & 0.08 & 0.09 & 0.1 \\ 0.11 & 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 \\ 0.21 & 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.3 \\ 0.31 & 0.32 & 0.33 & 0 & 0 & 0 & 0 & 0.38 & 0.39 & 0.4 \\ 0.41 & 0.42 & 0.43 & 0 & 0 & 0 & 0 & 0.48 & 0.49 & 0.5 \\ 0.51 & 0.52 & 0.53 & 0 & 0 & 0 & 0 & 0.58 & 0.59 & 0.6 \\ 0.61 & 0.62 & 0.63 & 0 & 0 & 0 & 0 & 0.68 & 0.69 & 0.7 \\ 0.71 & 0.72 & 0.73 & 0.74 & 0.75 & 0.76 & 0.77 & 0.78 & 0.79 & 0.8 \\ 0.81 & 0.82 & 0.83 & 0.84 & 0.85 & 0.86 & 0.87 & 0.88 & 0.89 & 0.9 \\ 0.91 & 0.92 & 0.93 & 0.94 & 0.95 & 0.96 & 0.97 & 0.98 & 0.99 & 1.0 \end{pmatrix}$$

21. Récupérer les indexes auxquels sont situées les valeurs 0