

Info. TP4: retour des tableaux numpy

- <https://learn.heeere.com/2020-infospichi-0b73>

Rendu : sous forme d'archive

À la fin, vos réponses seront rendues dans une archive zip : cette archive contiendra un fichier `compte-rendu.txt` (avec votre **nom**, **groupe**, et les réponses aux questions précédées d'une apostrophe) et vos **fichiers pythons**.

Pour faire une **archive zip** de votre TP pour le rendu, exécutez, dans le terminal :

```
cd ~/info-spichi/  
zip -r tp4.zip tp4/
```

Important : mise en place

Suivre les consignes des TP précédents, et travailler dans un dossier `tp4`.

Exercice 1 – Téléchargement et Souvenirs

Dans le dossier `tp4`, téléchargez et décompressez l'archive du tp4, téléchargeable sur le site du cours. Celle-ci contient des fichiers de base dont il faudra copier le contenu pour démarrer certains exercices. Cela peut être réalisé directement depuis le terminal avec les commandes :

```
wget https://learn.heeere.com/2020-infospichi-0b73/raw/fichiers-tp4.zip  
unzip fichiers-tp4.zip
```

Note : vous aurez besoin de copier le fichier `qtido.py` dans chacun des exercices.

Pour cet exercice, travaillez dans un dossier `ex1` lui même dans le dossier `tp4`.

Dans un fichier `base.py`, recopier le programme suivant (ou copier le contenu depuis `base-souvenirs.py`) :

```
1 def cercle_gauche(f, xgauche, rayon):  
2     cercle(f, xgauche+rayon, y, rayon)  
3  
4  
5 f = creer(400, 400)  
6 xgauche = 0  
7 y = 100  
8 cercle_gauche(f, xgauche, 50)  
9 cercle_gauche(f, 100, 50)  
10 y = 200  
11 cercle_gauche(f, xgauche, 10)  
12 attendre_fermeture(f)
```

'Q1) Quelle erreur s'affiche quand vous lancez le programme ? Pourquoi ?

Q2) Résolvez le problème dans `base.py`.

'Q3) Dans un fichier `princ.py`, copiez le programme précédent et appliquez la technique de la fonction principale. Le programme fonctionne-t-il encore ?

Q4) Dans un fichier `princ2.py`, copiez le contenu de `princ.py` et changez le programme pour que la fonction `cercle_gauche` n'essaie pas d'utiliser de variables globales.

Exercice 2 – Affichage d’une grille au hasard

Pour cet exercice, travaillez dans un dossier `ex2` lui même dans le dossier `tp4`.

Créez un nouveau fichier `hasard.py` en partant du contenu de `base-grille.py`.

Q5) Quelle erreur s’affiche quand vous lancez le programme ? Pourquoi ?

Q6) Corrigez ces erreurs.

Q7) Prenez le temps nécessaire pour comprendre ce programme. Faites ensuite que la couleur de chaque carré ne se basent pas toujours sur la valeur 0.5 mais sur la valeur de la grille aux indices correspondants au carré. Vérifiez bien sûr que cela fonctionne.

Q8) Pour l’instant la grille est une grille de zéros (à part deux valeurs). À l’aide de la fonction `normal` (du module `numpy.random`) qui tire des nombres selon une loi normale, faites que la grille soit remplie au hasard avec des nombres tirés au hasard d’une loi normale de moyenne 0.5 et d’écart type 0.2. **Important :** La loi normale peut donner n’importe quel nombre, y compris plus petit que 0 et plus grand que 1. Ces valeurs ne sont pas valables pour la fonction `couleur`. Pour contourner ce problème, vous pouvez utiliser la fonction `np.clip(a, 0, 1)` qui prend un tableau `a` et renvoie un tableau avec les mêmes valeurs que `a` sauf que les valeurs plus petites que 0 sont mises à 0, et celles plus grandes que 1 mises à 1.

Exercice 3 – Aggrégations et notion d’axes

Pour cet exercice, travaillez dans un dossier `ex3` lui même dans le dossier `tp4`.

Créez un nouveau fichier `axes.py` en partant du contenu de `base-axes.py`.

Q9) Lancez le programme `axes.py`, étudiez le fichier Python et la sortie produite.

Certaines fonctions Python acceptent ce que l’on appelle des « paramètres nommés ». C’est le cas de la plupart des fonctions d’aggrégation de numpy, avec les paramètres nommés `axis` (entier) et `keepdims` (booléen).

Q10) Quelle est la différence entre `np.min` et `np.min` avec le paramètre nommé `axis` mis à 0 ? Quel résultat est calculé par `np.min` dans les deux cas ?

Q11) Mêmes questions avec le paramètre nommé `keepdims` ?

Expérimentez avec le programme en modifiant la fin de `axes.py` pour répondre aux questions suivantes.

Q12) Quelles autres valeurs peut prendre `axis` ? Que cela fait-il ?

Q13) Que se passe-t-il avec `keepdims=False` ?

Exercice 4 – Statistiques

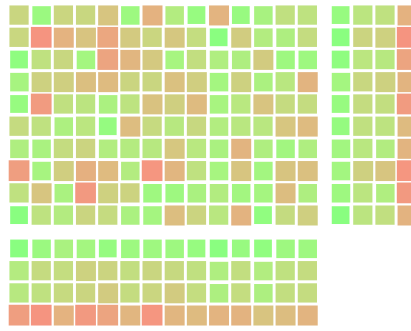
Pour cet exercice, travaillez dans un dossier `ex4` lui même dans le dossier `tp4`.

Créez un nouveau fichier `stats_horiz.py` en partant du contenu de `hasard.py`.

Q14) Faites que les 4 colonnes à droite affichent non pas des valeurs constantes mais les statistiques suivantes, extraite à partir de la grille `g` : le minimum (min), la moyenne (mean), la médiane (median), le maximum (max). Ainsi, le premier carré sera coloré en fonction de la valeur minimale de la première ligne de `g`, etc.

Créez un nouveau fichier `stats.py` en partant du contenu de `stats_horiz.py`.

Q15) Sur le modèles des 4 colonnes, faites que le programme affiche aussi 4 lignes de carrés donnant les statistiques (min, mean, median, max) par colonne.



Exercice 5 – Touches 0 à 3

Pour cet exercice, travaillez dans un dossier `ex5` lui même dans le dossier `tp4`.

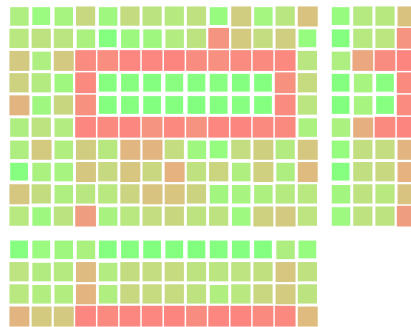
Créez un nouveau fichier `ex5.py` en partant du contenu de `stats.py`.

Q16) Implémentez le comportement de la touche 0, c'est à dire que quand on appuie sur 0 (sur le clavier), la grille `g` doit être remplacée par une nouvelle grille de la même taille contenant des valeurs au hasard (avec la même distribution aléatoire que précédemment).

Q17) Implémentez 1 : plus précisément, l'appui sur 1 doit mettre à la valeur 1 toutes les valeurs du rectangle depuis la ligne d'indice 2 à la ligne 5 (incluses) et de la colonne 3 à la colonne 12 (incluses).

Q18) Implémentez 2 : plus précisément, l'appui sur 2 doit mettre à la valeur 0 toutes les valeurs du rectangle des lignes 3 et 4 (incluses) et de la colonne 4 à la colonne 11 (incluses).

Q19) Implémentez 3 : l'appui sur 3 doit remplacer la grille `g` par une moyenne pondérée entre la grille courante et une nouvelle grille tirée aléatoirement (avec la même distribution aléatoire que précédemment). Plus précisément, sous forme de matrices, la nouvelle valeur de `g` est $0.9 \times g + 0.1 \times r$ (ou r a été tirée au hasard).



Exercice 6 – Touches 4 à 6 : extraction et écriture de blocs

Pour cet exercice, travaillez dans un dossier `ex6` lui même dans le dossier `tp4`.

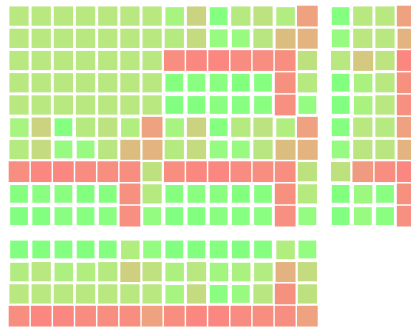
Créez un nouveau fichier `ex6.py` à partir du précédent.

Note : si `W` est un nombre entier (même pair) alors, en Python3, `W/2` sera un nombre réel et ne pourra pas être utilisé comme un indice de liste ou tableau. La division entière `W//2` permet d'obtenir un entier dans ce cas.

Q20) Implémentez 4 : la moitié droite du tableau doit être extraite et ré-écrite dans `g` par dessus la moitié gauche.

Q21) Implémentez 5 : la moitié supérieure du tableau doit être extraite et ré-écrite dans `g` par dessus la moitié inférieure.

Q22) Implémentez `6` : toutes les valeurs du quart supérieur gauche doivent être remplacées par une même valeur qui est calculée comme la moyenne de ce quart supérieur gauche.



Exercice 7 – Touches 7 à 9

Pour cet exercice, travaillez dans un dossier `ex7` lui même dans le dossier `tp4`.

Créez un nouveau fichier `ex7.py` à partir du précédent.

Q23) Implémentez `7` : en utilisant `linspace` et `reshape`, remplacez les deux dernières lignes par des valeurs (donc $2W$ valeurs) espacées régulièrement de 0 à 1.

Q24) Implémentez `8` : remplacez la première ligne de `g` par la dernière ligne de `g` mais prise à l'envers (aide : pensez au pas négatif).

Q25) Implémentez `9` : remplacez la seconde ligne de `g` par l'avant dernière ligne de `g` mais opposée (c'est à dire que chaque valeur est remplacée par $1 - \text{valeur}$).

Q26) Bonus : sur la ligne marquée avec `#M1`, faites que la taille d'un carré dépende de la valeur de la case (par exemple, de taille de plus en plus grande quand la valeur augmente, avec une valeur de 0 qui donne une taille de `taille - 6` et une valeur de 1 qui donne une taille de `taille - 1`).

