UNIVERSITÉ DE CAEN
U.F.R. des Sciences

L2
Calcul scientifique

TP 1 : Découverte de numpy, scipy et matplotlib

---

1 Introduction

L'objectif de ce TP est de se familiariser avec les librairies numpy, scipy et matplotlib. Il sera à réaliser en python 3. Les librairies utilisées sont installées sur les machines de l'université, vous pouvez néanmoins les installer sur vos propres machines à l'aide de l'utilitaire pip présent par défaut avec python.

N'hésite pas à regarder régulièrement la documentation de ces librairies, des exemples d'utilisation accompagnent généralement l'explication de chaque fonction.

- Python 3 : `https://docs.python.org/3/`
- Numpy : `https://docs.scipy.org/doc/numpy/reference/`
- Scipy : `https://docs.scipy.org/doc/scipy/reference/`
- Matplotilb : `https://matplotlib.org/contents.html`

À part si cela est précisé, vous ne devez pas utiliser directement de boucle (for,while) ou de branchement conditionnel (if) durant ce TP.

2 Les vecteurs

2.1 Création de vecteurs

1. Créez un vecteur de taille 5 contenant que des zéros. Affichez-le et affichez sa taille.
2. Créez un vecteur contenant les nombres compris entre 3 et 12 (exclus) avec un pas de 0.5
3. Créez un vecteur contenant les carrés des entiers compris entre -5 et 5 (exclus).
4. Créez un vecteur contenant toutes les puissances de deux d'entiers compris entre 1 et 65536 ($= 2^{16}$) inclus. Le résultat attendu ressemble à :

[1, 2, 4, 8, 16,..., 32768, 65536]

2.2 Visualisation de fonctions

1. Tracez avec matplotlib la fonction sinus entre 0 et 2*np.pi en calculant un point tous les 1e-3. Vérifiez que les valeurs en abscisse et en ordonnée sont correctes.
2. Faites l'affichage de la précédente question, mais avec uniquement 10 points. Il faut pour cela bien choisir l'écart entre les points des abscisses. Vous pouvez visualiser ces points en ajoutant '+-' aux arguments de la fonction plot.
3. Tracez sur la même figure les fonctions x,x^2,x^3 entre 0 et 2. Choisissez un écart entre chaque point suffisamment petit pour que les courbes paraissent lisses.

3 Comparaison python classique et fonction numpy (partie 1)

1. Calculez la somme des entiers entre 0 et 100 millions avec une **boucle for** en python sans utiliser de librairie.

2. Créez un vecteur contenant tous les entiers entre 0 et 100 millions puis calculez la somme des valeurs de ce vecteur en utilisant numpy. Aucun for ou while ne devra être utilisé.
3. Comparez les vitesses d'exécution des deux scripts. Vous pouvez mesurer le temps d'exécution en python avec les instructions suivantes :

```
import time

t1 = time.time()
%code python à tester
t2 = time.time()
print(t2-t1,'s')
```

Question bonus : Calculez cette somme en utilisant vos connaissances sur la somme des suites arithmétiques. Retrouvez-vous le même résultat ?

4 Les matrices

4.1 Création de vecteurs

1. Créez une matrice de taille (12,4) contenant que des zéros. Affichez la et affichez sa taille.
2. Créez une matrice identité de dimension (16,16). Affichez la et affichez sa taille.
3. Créez et affichez la matrice $\begin{bmatrix} 1 & 31 \\ 51 & 12 \end{bmatrix}$
4. Affichez uniquement la ligne 0 de la matrice précédente. Affichez la colonne 1.

4.2 Lecture et manipulation d'une image

L'objectif de cet exercice et de lire une image, de lui ajouter un bord noir et de l'afficher. Nous allons procéder pour cela en différentes étapes, comme expliqué ci-dessous.

1. Il existe dans la librairie scipy deux images de test que vous pouvez utiliser. L'une d'elles peut être récupéré par le code suivant :

```
import scipy as sc
import scipy.misc
im = sc.misc.ascent()
```

2. À l'aide de la fonction plt.imshow(im,cmap='gray',vmin=0,vmax=255) de matplotlib affichez l'image précédente. L'option cmap='gray' permet de préciser que l'image est en noir et blanc. vmin = 0 et vmax = 255 permet de préciser que les pixels sont définis par des valeurs allant de 0 (noir) à 255 (blanc). N'oubliez pour la fonction plt.show() après plt.imshow .
3. La fonction plt.show permet d'exécuter toutes les instructions d'affichage demandées. Il est parfois nécessaire d'afficher plusieurs fenêtres d'affichage. Pour cela vous pouvez appeler la fonction plt.figure(n) avec n le numéro de la fenêtre à gérer. Lorsque plt.show sera appelé, toutes les fenêtres seront affichées. Affichez dans deux fenêtres en parallèle l'image précédente.
4. Nous allons ajouter un cadre noir à notre image. Pour cela, construisez un tableau numpy contenant uniquement des zéros dont les dimensions sont celles de l'image augmentées de 100 en ligne et colonne. Recopiez ensuite l'image à partir de la ligne 50, colonne 50 (sans utiliser de for ou de while).

5 Augmentation de contraste d'une image

L'objectif de cet exercice est d'augmenter le contraste d'une image, en appliquant aux niveaux de gris des pixels une transformation affine telle que le plus petit niveau de gris a pour valeur 0 et le plus grand 255, après transformation des niveaux de gris.

Nous utiliserons pour cela deux méthodes et comparerons l'efficacité de ces deux méthodes.

Mais auparavant, vous commencerez par charger l'image face_gris.png (fournie sur ecampus) en mémoire dans la variable `im` et vous l'afficherez à l'écran. La lecture d'une image sur disque peut se faire au moyen de la fonction `sc.ndimage.imread` de la librairie `scipy`. L'affichage peut se faire à l'aide de la fonction `plt.imshow(im,cmap='gray',vmin=0,vmax=255)` de matplotlib. L'option cmap='gray' permet de préciser que l'image est en noir et blanc. vmin = 0 et vmax = 255 permet de préciser que les pixels sont définis par des valeurs allant de 0 (noir) à 255 (blanc).

Première méthode : utilisation de boucles

1. À l'aide d'instruction if et for (en utilisant que python 3 de base) trouvez la valeur minimale et maximale des pixels de l'image. Nous les noterons p_{min} et p_{max} dans la suite du sujet.
2. À l'aide d'instructions if et for créer une nouvelle image dont les pixels correspondent à l'équation $255\frac{p_{ij} - p_{min}}{p_{max} - p_{min}}$ où p_{ij} est le pixel de la ligne i et de la colonne j. Cette opération consiste à ramener les valeurs effectives des pixels entre 0 et 255 et forcer ainsi l'utilisation de toute la plage de valeur possible. Pour cette question vous calculerez les nouveaux pixels, pixel par pixel, à l'aide de l'image d'origine. Pour simplifier l'initialisation, vous pouvez commencer avec une image noir avec l'instruction : im2 = np.zeros(im.shape).
3. Affichez cette image et mesurez le temps d'exécution de votre programme. Vous devriez voir l'image "cachée" du fichier face_gris.png .

Seconde méthode : utilisation de numpy Utilisez maintenant les fonctions numpy : `np.min` et `np.max` pour calculer la valeur minimale et maximale des pixels de l'image. Effectuez le même traitement que pour la méthode précédente, mais cette fois-ci sans boucle, en traitement l'ensemble de l'image en même temps grâce aux opérations mathématiques sur les array numpy.

Comparez les images obtenues avec les deux approches. Que pouvez-vous dire des temps d'exécutions dans les deux cas ?