Mémo Python (1/2)

- Bonnes pratiques :
  - Utiliser un vrai éditeur de code Python (spyder, jupyter-notebook, etc.)
  - Se placer dans un dossier dédié
  - Toujours écrire dans un script (*fichier.py*)
  - Surveiller le workspace pour voir quelles sont et surtout la taille des variables
  - Consulter l'aide des fonctions `(help(function))`
- Modules :
  - Accéder aux fonctions d'autres modules :

```
import numpy as np #toutes les fonctions,
     possibilité de renommer le module
from signal import convolve
   #import d'une seule fonction
import subfolder.my_module
  #import de ./subfolder/my_module.py
```

- Fonctions :
  - Peuvent être écrites dans le script (avant le code appelant, comme en C)

```
def min_max(T):
    min_ = min(T)
    max_ = max(T)
    return min_, max_
```

- Exécution (*spyder*) :

- Tout le script : **F5** ou **<Run>**

- Par section : **ctrl+enter** ou **<Run section>**

```
h, w, c = img.shape

#%% Affichage
plt.figure(), plt.imshow(img)

#%% Vectorisation
img_vect = img(:)
```

- Par sélection : **F9**

```
h, w, c = img.shape

#Affichage
plt.figure(), plt.imshow(img)
```

- Principales différences avec Matlab :
  - Indices [ ], à partir de 0 : tableau de taille `l`

```
tab[0] #premier élément
tab[l-1] = tab[-1] #dernier élément
```

  - Vecteur d'échantillonnage :

```
range(0,100) #0, 1, ... , 99
```

  - Opération terme à terme par défaut :

```
a = np.array([1,2,3,4])
a*a #array([1,4,9,16])
```

Mémo Python (2/2)

- Commandes de bases :

```python
#Modules utiles
import numpy as np #tableau, opérateurs maths
import matplotlib.pyplot as plt #image, affichage
import skimage  #par ex. espace couleur ycbcr
from scipy import signal #par ex. convolution
import cv2 #opencv, par ex. vidéo, imwrite

#Manipulation d'image
img = plt.imread('path/img.png') #chargement
h, w, c = img.shape  #image couleur
print(h)

img_vect = img.ravel()  #Vectorisation
G = img[:,:,1] #accès dimension 2e canal = vert

#Mise à zéro
img = np.zeros((h,w,c)) #ones() existe aussi
img = np.copy(img*0)

#Sous-échantillonnage
img_se = img[1:h:4, 1:w:4] #ou img[::4,::4]

#Création d'un vecteur/d'une matrice
mat = [[1,1],[2,2],[3,3]]  #liste de taille 3x2
mat = np.array(mat)  #np.array

#Produits vecteurs/matriciels
vect = np.array(np.matrix(range(1,11,2))).T
           #range(début,fin,pas)  #T = transposée
vect_5_1 = vect*vect #terme à terme
vect_5_5 = np.dot(vect, vect.T) #prod. matriciel
vect_1_1 = np.dot(vect.T, vect) #prod. matriciel

#Somme sur matrice nD
sum_G = np.sum(G**2) #somme de tous les termes^2
sum_G = np.mean(img, axis=2) #conversion niv. gris
```

```python
#Seuillage sur une matrice
mask = G > 100 #mask = carte binaire (hxw)
#Équivalent à faire :
mask = np.zeros((h,w))
for i in range(0,h):
   for j in range(0,w):
      if (G[i,j]>100):
          mask[i,j] = 1

G[mask==0] = 0 #Mise à zéro des pixels
                  #de G où mask est nul (0)
G = G*mask #Équivalent à multiplication terme à terme

#changement d'espace couleur
img_ycbcr = skimage.color.rgb2ycbcr(img)

#convolution image couleur (même filtrage sur R,G,B)
filter_ = np.ones([7,7,1])/49
img_f = signal.convolve(img, filter_, mode='same')

#Affichage
img_L = np.mean(img, axis=2)
plt.figure()
plt.subplot(121) #Affichage multiples 1x2
plt.plt(img_L[0,:])
plt.title('Profil de la première ligne de L')
plt.subplot(122)
plt.plot(img[0,:,0],'ro') #superposition par défaut
plt.plot(img[0,:,1],'g+')
plt.plot(img[0,:,2], color=[0,0,1])
plt.title('Profil RGB de la première ligne')
plt.xlabel('x'), plt.ylabel('Intensité')
plt.show()
```

Autres fonctions utiles : `np.squeeze, np.tile, plt.ginput, …`

Liens vers les docs : Tuto général Python
Tuto numpy, matplotlib, scipy
Tuto scikit-image