
Exercices en Python

Exercices Cours/diaporama 1

Math Théorie

Exercices en Python (voir syllabus d'exercices sur Teams)

Vous réalisez les programmes comme si Python n'implémentait pas les fonctions demandées !

N'utilisez pas chatGPT ou autre pour ce genre de programme au pire utilisez-le pour corriger vos fautes ou commenter votre code

Exercice 1

En Python, créer un programme qui génère une matrice aléatoire dont les éléments sont des entiers et un autre où ce sont des réels. La taille de la matrice est entrée au clavier. Le résultat est affiché proprement à l'écran

Exercice 2

En Python, créer un programme qui génère une matrice unité, une matrice diagonale, une matrice triangulaire, une matrice creuse, une matrice nulle. La taille de la matrice est entrée au clavier

Exercice 3

En Python, créer un programme qui additionne/soustrait deux matrices après avoir vérifié que l'addition est possible. Les matrices sont entrées au clavier élément par élément.

Exercice 4

En Python, créer un programme qui crée une matrice aléatoire, qui calcule ensuite son opposée, affiche les deux matrices et qui vérifie que leur somme fait bien … (à compléter par vous-même)

Exercice 5

En Python, créer un programme qui exécute le produit d'une matrice aléatoire par un scalaire entré au clavier. La matrice de départ, le scalaire et le produit sont affichés à l'écran.

Exercice 6

En Python, créer un programme qui exécute le produit de deux matrices. Les matrices sont entrées au clavier élément par élément. Leur compatibilité est vérifiée avant d'introduire tous les éléments. Les matrices de départ et le produit sont affichés à l'écran.

Exercice 7

En Python, créer un programme qui calcule et affiche la somme des lignes d'une matrice aléatoire de deux manières différentes.

Exercice 8

Idem pour la somme des colonnes

Exercice 9

En Python, créer un programme qui calcule et affiche la $n^{ème}$ puissance d'une matrice carrée aléatoire de manière « économique »

Exercice 10

En Python, créer un programme qui exécute le produit de Hadamard de deux matrices. Les matrices sont entrées au clavier élément par élément. Leur compatibilité est vérifiée avant d'introduire tous les éléments. Les matrices de départ et le produit sont affichés à l'écran.

Exercice 11

En Python, créer un programme qui calcule et affiche la transposée d'une matrice aléatoire. La matrice de départ et sa transposée sont affichées à l'écran.

Diaporama 2 Python

math-ex1-1.py (en téléchargement sur Teams)

math-ex1-2.py (en téléchargement sur Teams)

Exercices Cours/diaporama 2 et 3

Matrice en Python avec et sans Numpy

Enoncé exercice 1.3 :

Premiers avec NumPy: utilisation des arrays pour manipuler des matrices
Ecrire un pgm en Python utilisant Numpy qui :

- Affiche les Arrays

A= ([4.1, 2.0, 0],
[4.6, 1, 6],
[2, 8, 3])

B= ([1, 1, 0],
[1.0, 1, 1],
[2, 2, 2])

- Calcule A+B et l'affiche (A quoi correspond cette addition?)

Solution possible Exercice 1.3 : `math-ex1-3.py` (en téléchargement sur Teams)

Enoncé exercice 1.4 :

- Création et affichage de la 'matrice' 3x3 suivante avec des éléments de type float:

  M=([4.1, 2.0, 0],
  [4.6, 1, 6],
  [2, 8, 3])

- Affichage de l'élément m_{23} de cette ‘matrice’
- Affichage de la 3ème ligne (quelle que soit la matrice introduite)
- Affichage de la première colonne (quelle que soit la matrice introduite)

- Création et Affichage d'une 'matrice' 3x3 de type entier ne contenant que des 1

- Création et Affichage d'une 'matrice' unité diagonale 5x5 éléments type float

- Réarrangement et affichage de la liste linéaire suivante en ‘matrice’ 3x2:

- A = ([2, 4, 6, 12, 24, 36])

Solution possible Exercice 1.4 : `math-ex1-4.py` (en téléchargement sur Teams)

Enoncé exercice 1.5 :

Ecrire un pgm en Python utilisant Numpy qui :

- Affiche les Arrays suivantes:

```
A= ([4.1, 2.0, 0],
    [4.6, 1,   6],
    [2,   8,   3])
B= ([1,   1,   0],
    [1.0, 1,   1],
    [2,   2,   2])
C= ([1,   2],
    [0,   1],
    [3,   1],)
```

- Calcule A x B et l'affiche (produit d'Hadamard)
- Calcule A@B et l'affiche (produit matriciel)
- Calcule le produit matriciel de A par B (mais sans utiliser l'opérateur @, en créant votre propre algorithme), de A par C et de C par A en indiquant si ce n'est pas possible pourquoi.

Solution possible Exercice 1.5 : `math-ex1-5.py` (en téléchargement sur Teams)

Enoncé exercice 1.6 :

Ecrire un pgm en Python utilisant Numpy qui :

- Propose l'introduction des éléments de 2 matrices quelconques (en demandant avant l'introduction leur dimensior
- Calcule et affiche le produit matriciel de A par B et de B par A en indiquant si ce n'est pas possible pourquoi.
- Calcule la somme de A+B en indiquant si ce n'est pas possible pourquoi.
- Calcule et affiche la transposée de A en indiquant son nombre de lignes et de colonnes

Solution possible Exercice 1.6 : `math-ex1-6.py` (en téléchargement sur Teams)

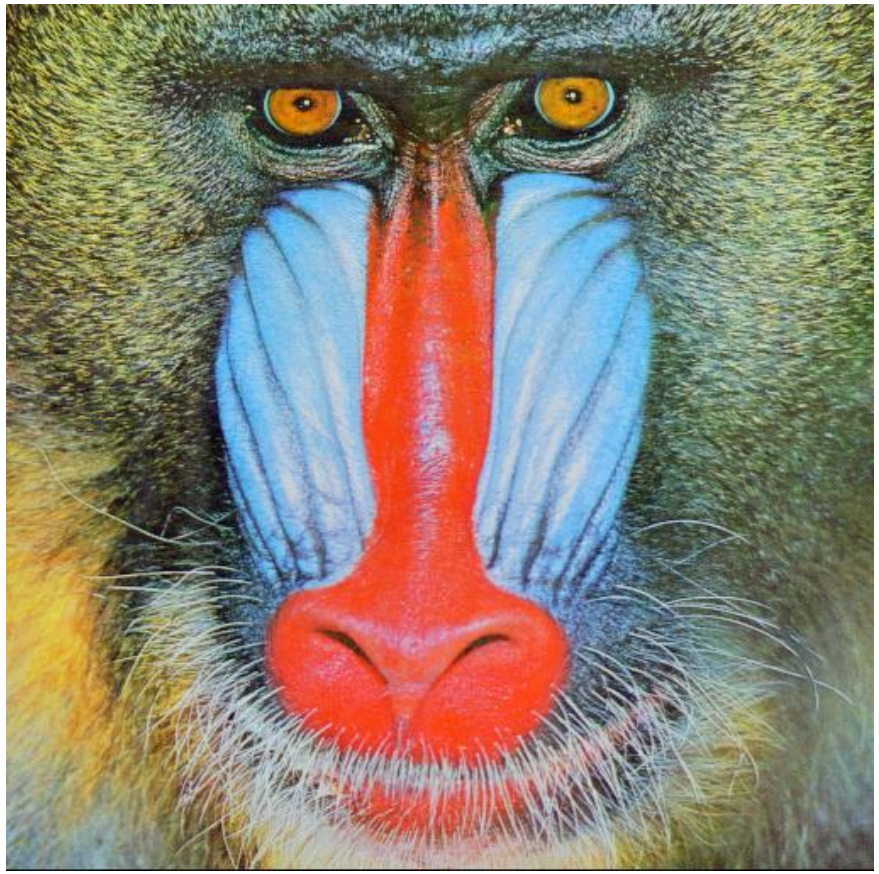
Exercices Cours/diaporama 4

Traitement d’images 1

Le but de ces exercices est à la fois de présenter quelques méthodes de manipulation d'images sans utiliser les fonctions toutes faites d'un module et de le faire d'une façon un peu plus mathématique que d'ordinaire en utilisant numpy et la manipulation mathématique des arrays.

PIL ou pillow nous permettra d'ouvrir les images directement dans un tableau numpy, mais nous n'utiliserons pas ses fonctionnalités de traitement d'images, nous le ferons en codant nous même les fonctions.

Lenna ou Lena est une image test standard largement utilisée dans le domaine du traitement de l'image depuis 1973. Il s'agit d'une partie de la photo du mannequin suédois Lena, prise par le photographe Dwight Hooker, tirée de la page centrale du magazine Playboy de novembre 1972. (https://fr.wikipedia.org/wiki/Lenna)
Cette image pose problème au mouvement féministe qui suggère de ne plus l'employer. Aussi pouvez-vous utiliser l'image 4-02-03 issue d'une banque standard, même si elle n'est pas aussi classique pour comparer les effets d'un algorithme de traitement d'image.



Ces images sont disponibles en différents formats (jpg, png,bmp,gif, tif) et en différentes tailles (220 x 220 pixels ou 512 x 512), en couleur et en noir et blanc (BW) dans le répertoire image

Exercice 2.1 (déjà fait ex2-1.py)

L'exercice 2.1 vous donne la syntaxe d'ouverture d'une image et de sauvetage de la nouvelle image. En fait, on ouvre l'image dans un tableau numpy nommé image, on récupère ses dimensions pour les afficher, on la transforme (dans cet exercice, on la recopie simplement) et on la sauve sous un autre nom/format.

On peut aussi facilement récupèrer les dimensions de l'image avec la fonction shape et stocker dans nb_lignes le nombre de lignes de notre tableau et dans nb_colonnes le nombre de colonnes pour les afficher.

Cet exercice est déjà fait (ex2-1.py), il doit vous permettre de vérifier que votre environnement est correct (bibliothèque PIL ou Pillow)

Note:

dans les exercices, nous laisserons le soin à PIL/Pillow d'encoder/décoder les formats d'image sans nous en préoccuper, ce qui nous intéresse, ce sera les modifications/traitements sur l'image. Vous pourriez bien sûr ouvrir le fichier via python, mais il faudra alors parfaitement connaître la structure des fichiers graphiques ce qui sort du cadre de ce cours.

Exercice 2.2 (déjà fait ex2-2.py)

Chaque pixel est représenté dans le tableau numpy par un triplet (r,v,b) où r, v et b sont des nombres entre 0 et 255 (8 bits de profondeur) représentant la "proportion" de rouge, vert et bleu pour afficher chaque pixel.
Un print du tableau affichera la valeur des pixels sous forme de triplets

Dans cet exercice on ouvre une minuscule image d'un damier de couleur rouge, vert et bleu de 3x2 pixels et on affiche le tableau correspondant. Dans ce format, l'ouverture de 6x2.bmp permet de voir clairement la structure.

Les pixels sont repérés dans le tableau par la ligne et la colonne comme dans une matrice. Autrement dit, pour récupérer le pixel tout en bas à gauche de l'image de Lenna, on utilisera image[511,0] puisque l'image est de dimension 512 par 512, que la première ligne porte le numéro « 0 », donc la dernière le numéro 511 et que le tableau de réception de l'image s'appelle image.

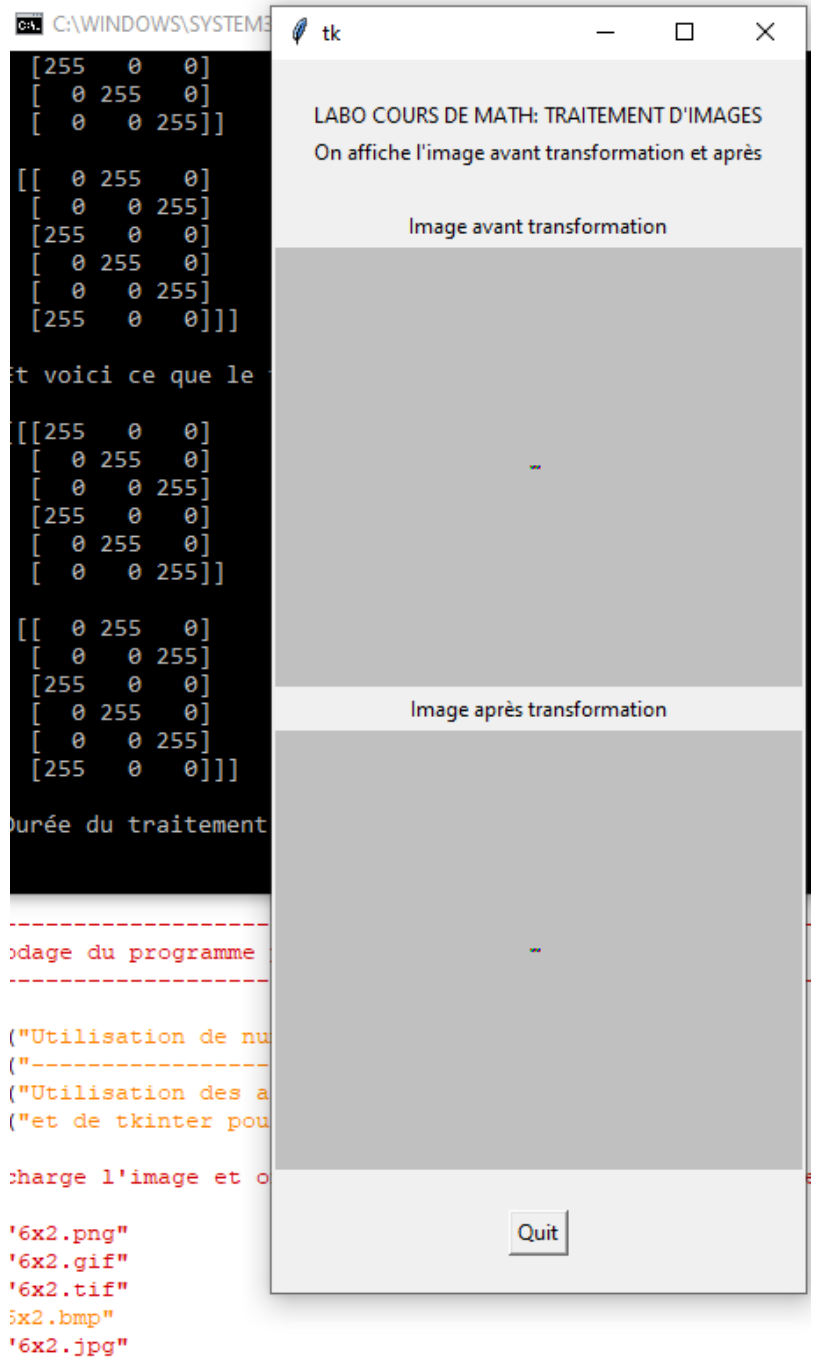
On peut aussi facilement récupèrer les dimensions de l'image avec la fonction shape et stocker dans nb_lignes le nombre de lignes de notre tableau et dans nb_colonnes le nombre de colonnes.

Si on effectue une manipulation sur le fichier, on sauvera l'original dans image_entree.png (pour que l'original Lenna.png reste toujours dispo pour les autres scripts) et l'image modifiée dans

Exercice 2.2 suite

Vous remarquerez que l'on a rajouter la **bibliothèque time** qui nous servira à mesurer le temps mis pour réaliser nos algorithme de traitement d'image en calculant la différence entre l'heure de début du script et l'heure de fin.

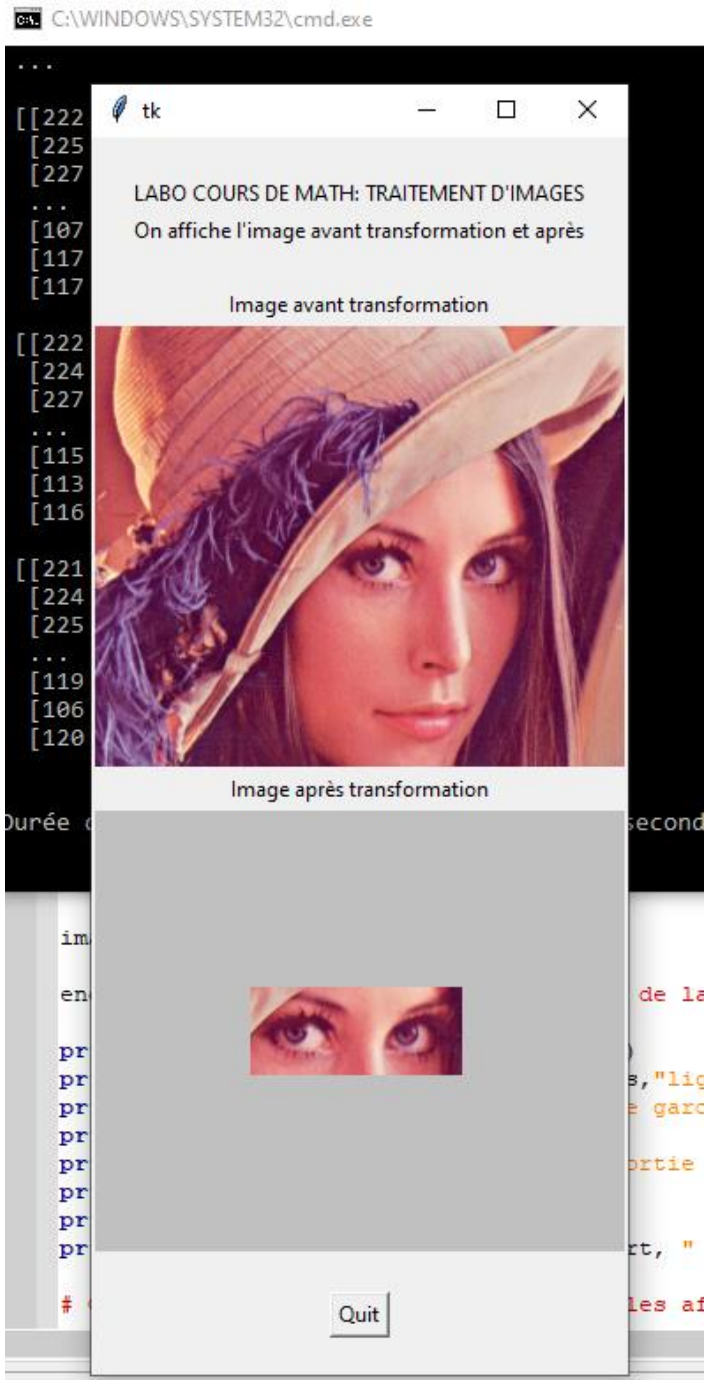
On a aussi ajouté la **bibliothèque tkinter** afin d'afficher l'image de départ et l'image du résultat du traitement, cela vous évite d'avoir à les ouvrir pour comparer.
Si vous gardez les noms « image_entree.png » et « image_sortie.png » , vous ne devrez rien changer dans la boucle tkinter, l'affichage se fera automatiquement dans une fenêtre graphique en popup.
L'affichage des tableaux et textes continuera à se faire dans la console.



Exercice 2.3 récupérer une partie d'une image.

Pour cela il suffit de ne conserver qu'une partie de notre tableau de données.
C'est très facile avec un tableau numpy : image[2:6,3:9] permet de récupérer la zone entre la ligne 2 et 5 et la colonne 3 et 8 (en python on ne prend pas la dernière valeur).

Si on souhaite récupérer le regard de Lenna512, on peut récupérer la zone entre les lignes 240 et 290 et les colonnes 240 et 360.



Exercice 2.4 image miroir

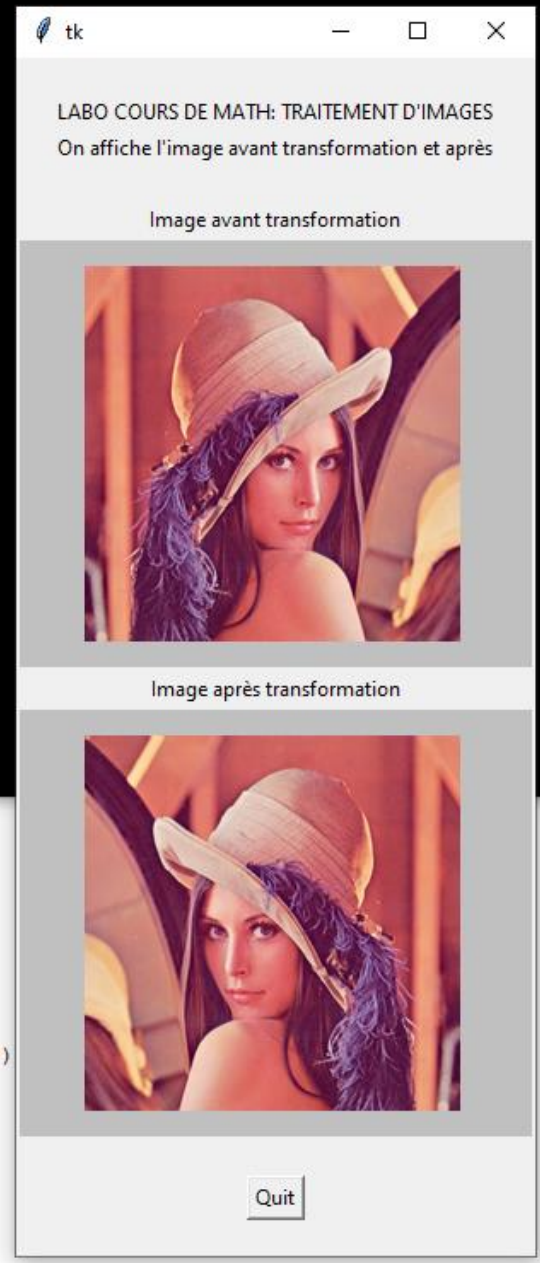
Pour retourner une image (inverser gauche-droite: la colonne 0 devient la 511, la 1 devient la 510, la 2, 509), il suffit simplement que sur chaque ligne, on mette dans le pixel situé à la colonne col la valeur du pixel de la colonne (nb_colonnes-1)-col de l'image originale.

Exemple:
On balaye toutes les lignes et
pour la colonne 0 (col=0) on va mettre la valeur du pixel de la colonne 512-1-0=511
Pour la colonne 1 on mettra la valeur du pixel 512-1-1=510 etc..
Jusqu'à la colonne 511 avec la valeur du pixel 512-1-511=0
Et on passe à la ligne suivante.

Pour faire une image upside/down, remplacer ligne par colonne et colonne par ligne dans l'algorithme!

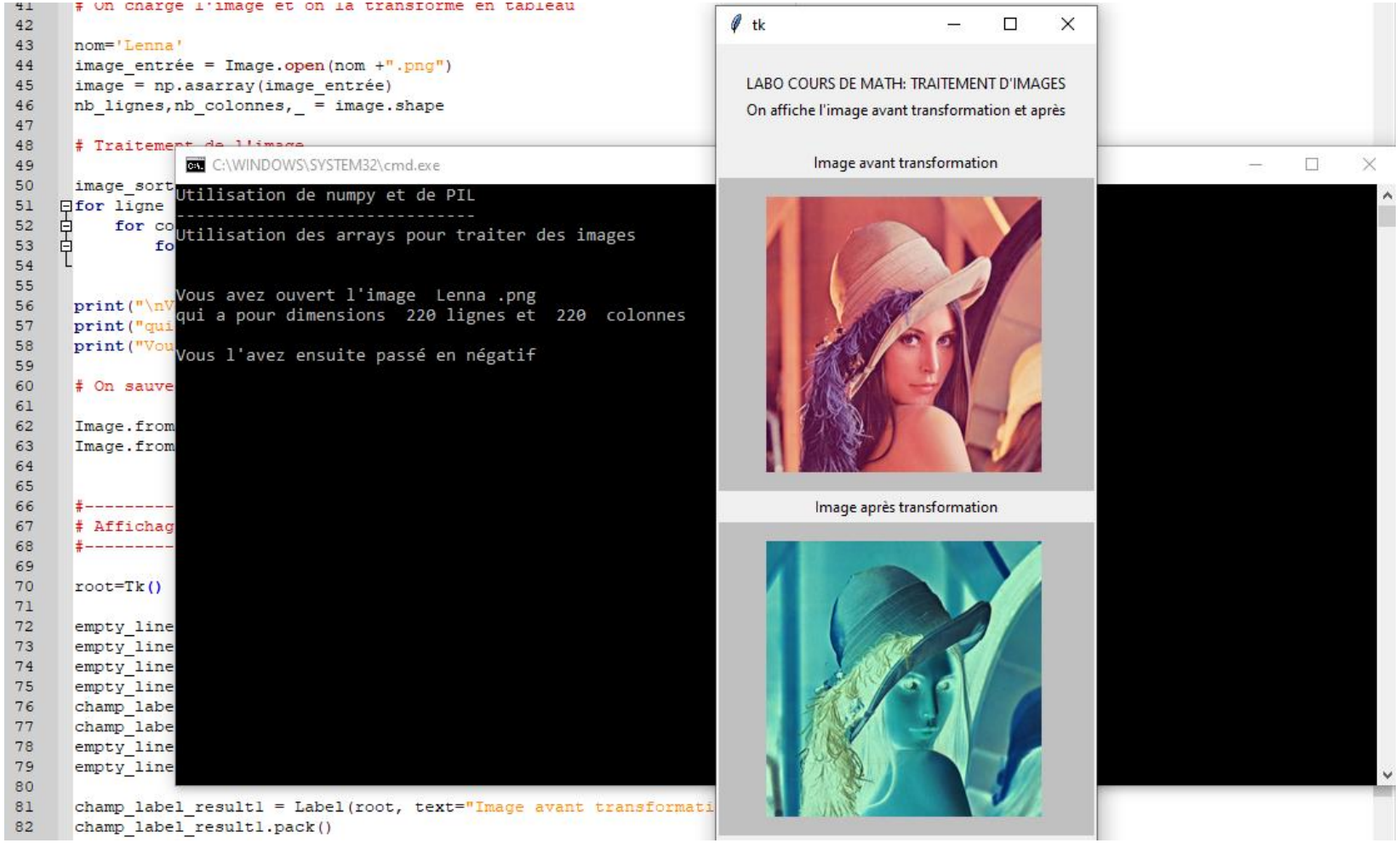
```
19
20  #-----------------------
21  # Importation des biblic
22  #-----------------------
23
24  import time
25  from tkinter import *
26  from PIL import Image, I
27  import numpy as np
28
29  #-----------------------
30  # encodage des fonctions
31  #-----------------------
32
33  #-----------------------
34  # encodage du programme
35  #-----------------------
36
37  print("Utilisation de nu
38  print("-----------------
39  print("Utilisation des a
40
41  # On charge l'image et c
42
43  nom='Lenna'
44  image_entree = Image.ope
45  image = np.asarray(image
46  nb_lignes,nb_colonnes,_
47
48  # Traitement de l'image
49
50  start = time.time()
51
52  image_sortie = np.copy(image)
53  for ligne in range(nb_lignes):
54      for col in range(nb_colonnes):
55          image_sortie[ligne,col] = image[ligne,nb_colonnes-1-col]
56
57  end = time.time()       # stocke l'heure de la fin du traitement
58
59  print("\nVous avez ouvert l'image ", nom, ".png")
60  print("qui a pour dimensions ", nb_lignes,"lignes et ", nb_colonnes, " colonnes\n")
61  print("Vous l'avez ensuite inversée gauche/droite (effet miroir)\n")
62  print ("Durée du traitement: ",end - start, " seconde\n")
63
64  # On sauvegarde les images pour pouvoir les afficher
65
66  Image.fromarray(image).save("image_entree.png")
67  Image.fromarray(image_sortie).save("image_sortie.png")
```

```
C:\WINDOWS\SYSTEM32\cmd.exe
Utilisation de numpy et de PIL
------------------------------
Utilisation des arrays pour traiter des images


Vous avez ouvert l'image  Lenna .png
qui a pour dimensions  220 lignes et  220  colonnes

Vous l'avez ensuite inversée gauche/droite (effet miroir)

Durée du traitement:  0.05296754837036133  seconde
```



Exercice 2.5 image en négatif

Pour inverser les couleurs (faire un négatif) , il suffit de remplacer chaque couleur (le rouge, le vert et le bleu) de valeur v par la valeur 255 – v (le complément à 255).



Exercice 2.6 isoler une des 3 couleurs

A partir d'une image, on peut isoler une seule composante de couleur, le rouge par exemple.
On peut aussi créer 3 images différentes dans lesquelles on ne garde à chaque fois que la composante rouge, verte ou bleue de chaque pixel. L'image originale étant en fait la superposition des trois.

Pour isoler une couleur, il suffit simplement de garder la valeur de la couleur souhaitée et de remplacer les autres par 0. Pour cela, le plus simple est de multiplier le tuple contenant les valeurs des 3 couleurs par un tableau (1,0,0) si vous voulez garder le rouge, (0,1,0) si vous voulez garder le vert et (0,0,1) si vous voulez garder le bleu (le triplet affiche les couleurs dans l'ordre Rouge (R), Vert (V) et Bleu (B)).

Le même algorithme peut servir à modifier la colorimétrie d'une image. On multiplie chaque couleur non plus par 1 ou 0 mais par un réel compris entre 0 et 1. Ainsi (0.8, 1, 0.95) diminuerait le rouge de 20%, garde le vert tel quel et diminue le bleu de 5%. Attention, il faudra régler le problème de multiplication entre integer et float et remettre le résultat final sous forme d'integer compris entre 0 et 255!

```
#-------------------------------------------------------------------------
# Importation des bibliothèques
#-------
import
from t
from P
import

#-----
# enco
#-----

print(
print(
print(

# On c

nom='L
image_
image
nb_lig
nb_col

# Trai

np.arr
image_
for li
    for col in range(nb_colonnes):
        image_sortie[ligne,col] = image[ligne,col]*[0,1,0]

print("\nVous avez ouvert l'image ", nom, ".png")
print("qui a pour dimensions ", nb_lignes,"lignes et ", nb_colonnes,
print("Vous avez ensuite isolé la composante Verte de l'image\n")

# On sauvegarde les images pour pouvoir les afficher

Image.fromarray(image).save("image_entree.png")
Image.fromarray(image_sortie).save("image_sortie.png")

#-------------------------------------------------------------------------
# Affichage dans tkinter
#-------------------------------------------------------------------------
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
Utilisation de numpy et de PIL
------------------------------
Utilisation des arrays pour traiter des images


Vous avez ouvert l'image  Lenna .png
qui a pour dimensions  220 lignes et  220  colonnes

Vous avez ensuite isolé la composante Verte de l'image
```

tk

LABO COURS DE MATH: TRAITEMENT D'IMAGES

On affiche l'image avant transformation et après

Image avant transformation

Image après transformation

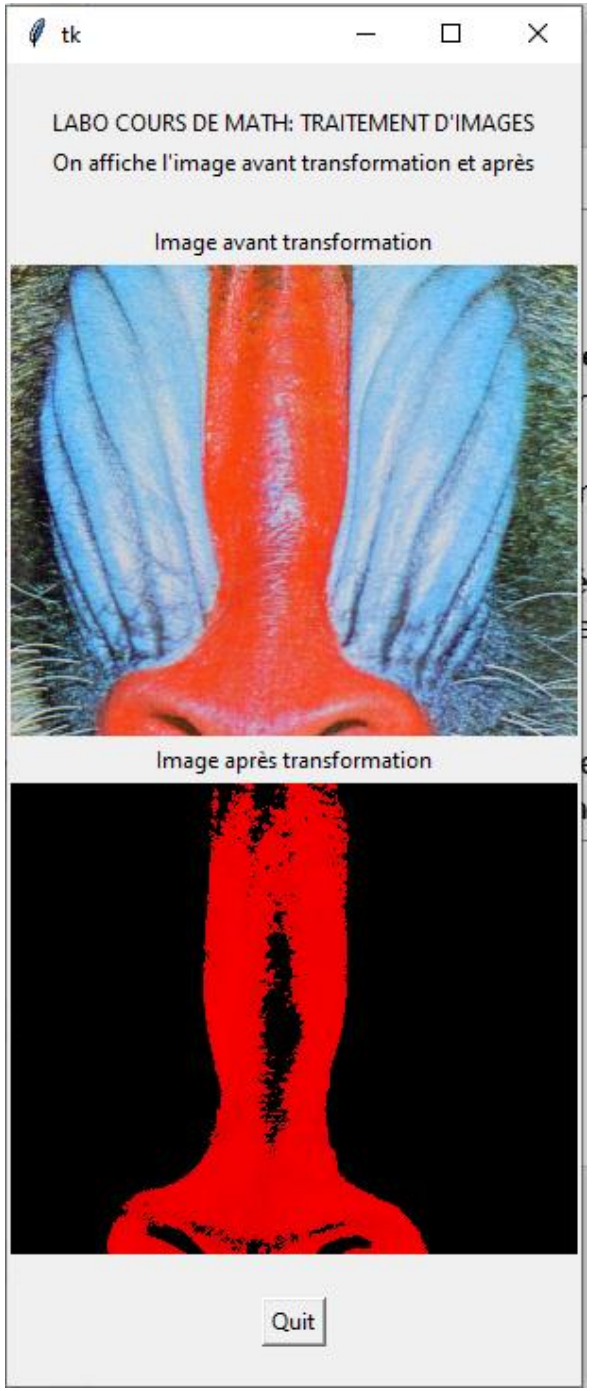
Quit

Exercice 2.6b isoler une des 3 couleurs à partir d'un seuil

Souvent dans l'analyse scientifique d'une image, on désire extraire de l'image les zones correspondant à un seuil minimal pour une couleur donnée.

Imaginer une petite modification de votre script qui fasse passer en noir les zones dont la composante rouge est inférieure à un certain seuil et conserve (ou force à 255 selon votre envie) les zones dont la composante rouge dépasse ce seuil.
Fixez le seuil à par exemple 220 et appliquez le filtre à l'image du singe.
Vous isolerez très facilement son nez.

Ce principe est la base de nombreux algorithme de sélection de zone et de mesure (par exemple sélection d'une lésion dans une radiographie permettant sa mesure)



Exercice 2.7 Mettre en nuance de gris

On pourrait imaginer faire la somme des valeurs rouge, vert et bleu et diviser par 3.
Mais dans ce cas, le gris correspondant à un rouge pur, un vert pur ou un bleu pur serait identique et donc l'image manquerait de rendu.

On peut trouver sur Wikipédia la formule suivante pour mettre une photo en nuance de gris :
Pour chaque pixel, on remplace la couleur (r,g,b) par la couleur (luminance, luminance, luminance) où

luminance = 0.2126 * r + 0.7152 * v + 0.0722 * b

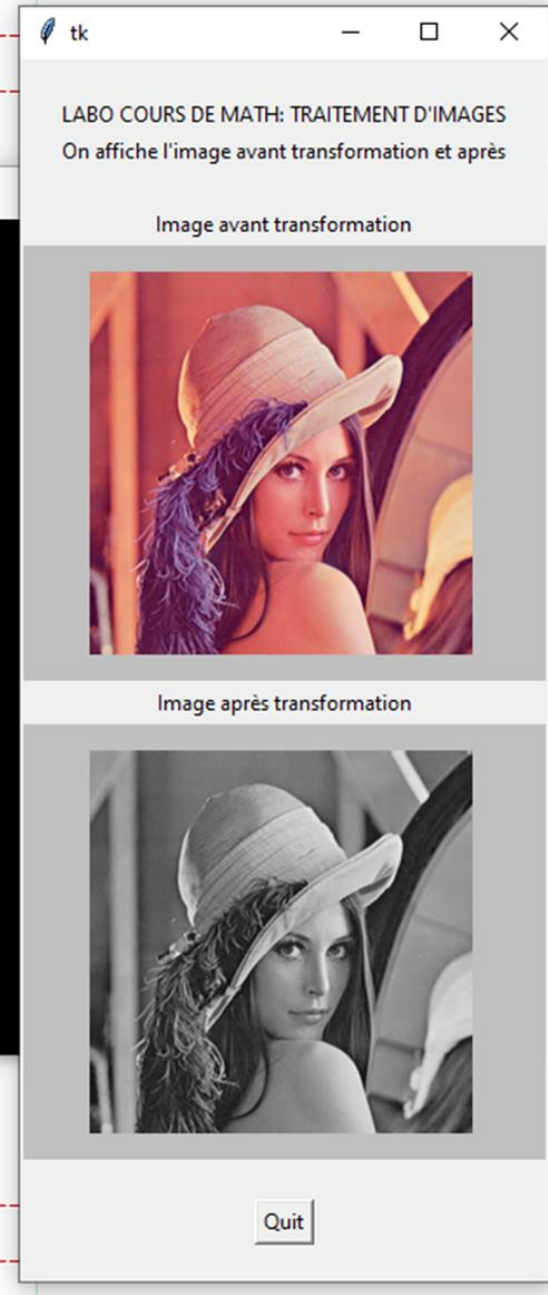
Dans ce cas un rouge pur (255,0,0) est transformé en (54 ,54 ,54)
un vert pur (0, 255,0) est transformé en (182 , 182 , 182)
un bleu pur (0,0, 255) est transformé en (18 ,18 ,18)

(il faut bien sur normaliser le résultat comme un entier compris entre 0 et 255!)

```
31  #-------------------------------------------------------------------------
32
33  #-------------------------------------------------------------------------
34  # encodage du programme principal
35  #-------------------------------------------------------------------------
36
37  print("Utilisation de numpy et de PIL")
38  pr
39  pr
40
41  #
42
43  no
44  im
45  im
46  nb
47  nb
48
49  #
50
51  #
52  #
53  #
54
55  st
56
57  im
58  fo
59
60
61
62
63  en
64
65  pr
66  pr
67  pr
68  pr
69
70  # On sauvegarde les images pour pouvoir les afficher
71
72  Image.fromarray(image).save("image_entree.png")
73  Image.fromarray(image_sortie).save("image_sortie.png")
74
75  #-------------------------------------------------------------------------
76  # Affichage dans tkinter
77  #-------------------------------------------------------------------------
78
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
Utilisation de numpy et de PIL
------------------------------
Utilisation des arrays pour traiter des images


Vous avez ouvert l'image  Lenna .png
qui a pour dimensions  220 lignes et  220  colonnes

Vous avez transformé cette image en noir et blanc

Durée du traitement:  1.9797797203063965  seconde
```



Exercice 2.8 Modifier la luminosité

Pour augmenter la luminosité d'une image, il suffit d'ajouter (ou soustraire) à toutes les valeurs un même nombre sans dépasser la valeur maximale qui est de 255 si on ajoute (ou minimale qui est 0 si on soustrait).

On pourra soit le faire avec une condition, soit utiliser la fonction min.

On peut aussi écrire en une ligne la modification en utilise la fonction:

np.where(condition,valeur_si_vraie,valeur_si_fausse)

où la condition écrite sur un tableau t de valeurs s'applique sur chaque valeur directement.

Par exemple, np.where(t<30,0,t-30) permet de créer un tableau dans lequel toutes les valeurs de t inféreures strictement à 30 sont remplacée par 0 et celles au dessus de 30, on soustrait 30. Cela permettrait de réduire la luminosité de 30 si on l'applique à une image.

Exercice 2.9: mixage de deux images

On peut, à partir de deux images, créer un mélange des deux.
Pour cela, on commence par choisir dans quelle proportion on veut les mélanger (par exemple 60% de la première et donc 40% de la seconde).
Ensuite, il suffit de prendre comme valeur, pour chaque couleur de chaque pixel, 60% de la valeur de la première image + 40% de la valeur de la seconde image en veillant à ne pas dépasser 255 et à ne conserver que la valeur entière.

Ainsi, si par exemple le premier pixel de la première image est de couleur (10,20,30) et le premier pixel de la seconde image est de couleur (100,100,100) alors l'image mélangée sera de couleur (0.6*10 + 0.4*100, 0.6*20 + 0.4*100, 0.6 * 30 + 0.4*100) c'est à dire (46, 52, 58).

C'est assez facile puisque multiplier un tableau numpy par un nombre multiplie chacun de ses termes.

Seul détail technique, il faut penser à transformer le résultat final en tableau à valeur entière inférieure à 255 (en appliquant np.asarray(..., dtype=np.uint8) au résultat par exemple).

Exercice 2.10: Fond vert

Le principe est un peu le même qu'avec le fondu de deux images, si ce n'est que dans ce cas:

si le pixel de l'image 1 d'avant plan est vert, on le remplace totalement par le pixel équivalent de l'image 2 de fond souhaitée,

sinon on garde sa valeur initiale.

Exercice 2.11 Modifier le contraste

Pour chaque pixel de couleur (r,v,b), on peut définir son intensité par la moyenne des valeurs des 3 couleurs c'est à dire i=(r+v+b)/3. On a ainsi, si on parcourt tous les pixels de l'image, une intensité minimale i_min et une maximale i_max. L'idée, pour augmenter le contraste, est redimensionner la plage des intensités entre 0 et 255 de manière linéaire. Autrement dit, on veut que le pixel d'intensité i_min devienne d'intensité 0 et celui d'intensité i_max devienne d'intensité 255 et entre les deux, on modifie les valeurs de manière linéaire. Pour cela, un pixel d'intensité i à l'origine sera modifié en un pixel d'intensité normalisée i_n = 255 * (i-i_min)/((i_max-i_min)*i) en multipliant chaque couleur par cette valeur. On fera bien attention à ce que le résultat soit un entier inférieur à 255.

Exercice 2.12 Redimensionner une image

Supposons que nous voulions redimensionner notre image qui est de dimension a0 lignes et b0 colonnes en une nouvelle dimension a1 lignes et b1 colonnes.
On peut noter alors les ratios des transformations selon les lignes et les colonnes :
ratio_lignes = a0/a1 et ratio_colonnes = b0/b1.

Pour remplir notre nouvelle image, il suffit alors de remplir le pixel situé à la ligne ligne et la colonne col avec les couleurs du pixel de l'image de départ situé à la ligne int(ligne*ratio_lignes) et la colonne int(col*ratio_colonnes).

Remarque : Si on fait ainsi, lors d'un agrandissement, plusieurs pixels de l'image de départ seront recopiés donnant une impression de gros pixels. Il existe beaucoup de façon d'empêcher ce phénomène en lissant les couleurs par différentes méthodes mais nous ne nous y intéresserons pas ici.

Ecrivez le script suivant pour qu'il affiche une image en sortie de dimension 220 lignes et 220 colonnes de Lenna à partir de celle de 512x512 et comparer avec Lenna220.png.

Exercices Cours/diaporama 5

Math théorie Transversion /Gauss-Jordan Inversion de matrice

Ex 2.1 Enoncé

Écrivez un programme qui inverse deux lignes d'une matrice et qui sera appelé par l'instruction nom-fonction(A,i,j) où A est le nom de la matrice et i et j le numéro des lignes à échanger

Ex 2.2 Enoncé

Écrivez un programme qui soustrait d'une ligne k un multiple d'une autre ligne d'une matrice (transvection: $L_k = L_k – alpha.L_i$) et qui sera appelé par l'instruction nom-fonction(A,k,i,alpha) où A est le nom de la matrice et k et i le numéro des lignes à soustraire et alpha le facteur de multiplication de la $i^{ème}$ ligne avant soustraction

Ex 2.3 Enoncé

Écrivez un programme qui calcule le déterminant d’une matrice par la méthode du pivot de Gauss-Jordan (vous aurez besoin des programmes des exercices 1 et 2 sous forme de fonction)

Ex 2.4 Enoncé

Ecrivez un programme qui inverse une matrice par la méthode du pivot de Gauss-Jordan (vous aurez besoin des programmes des exercices 1 et 2 sous forme de fonction)

(Vous trouverez une description complète de l'algorithme en pseudo-code sur la diapo suivante)

(On cherche le max de la première colonne comme premier pivot et si il ne se trouve pas en a_{11}, on l'y met en échangeant la ligne du max avec la première. On divise la ligne par la valeur du max pour obtenir 1, puis on s'occupe de a_{12} et ainsi de suite on descend jusqu'au bout de la colonne, puis on passe à la colonne suivante on cherche le max, on le met par échange sur la diagonale et on s'occupe des valeurs du dessous et du dessus dans la colonne 2 et on passe à la colonne suivante etc.)

Soit une matrice A de dimensions n × m ;

L'algorithme de Gauss-Jordan en Pseudo-code est le suivant:

```
r = 0                              (r est l'indice de ligne du dernier pivot trouvé)
   Pour j de 1 jusqu'à m                (j décrit tous les indices de colonnes)
   |  Rechercher max(|A[i,j]|, r+1 ≤ i ≤ n). Noter k l'indice de ligne du maximum
   |                                (A[k,j] est le pivot)
   |  Si A[k,j]≠0 alors                 (A[k,j] désigne la valeur de la ligne k et de la colonne j)
   |  |  r=r+1                        (r désigne l'indice de la future ligne servant de pivot)
   |  |  Diviser la ligne k par A[k,j]      (On normalise la ligne de pivot de façon que le pivot prenne la valeur 1)
   |  |  Si k≠r alors
   |  |     |  Échanger les lignes k et r  (On place la ligne du pivot en position r)
   |  |  Fin Si
   |  |  Pour i de 1 jusqu'à n           (On simplifie les autres lignes)
   |  |  |  Si i≠r alors
   |  |  |  |  Soustraire à la ligne i la ligne r multipliée par A[i,j] (de façon à annuler A[i,j])
   |  |  |  Fin Si
   |  |  Fin Pour
   |  Fin Si
   Fin Pour
 Fin Gauss-Jordan
```

Ex 2.5 Enoncé

**Écrivez un programme qui calcule les solutions d’un système d’équation.
Le programme demandera combien il y a de variables et combien d'équations.
Si les deux chiffres sont différents, affichage d'un message d'erreur (lequel?) et arrêt ou bouclage du programme pour reposer la question.
Ensuite, boucle pour introduire le coefficient de chaque variable, équation par équation, mise en forme de la matrice correspondante et résolution du système (en utilisant pour l’inversion de matrice nécessaire le résultat de l’exercice précédent avec affichage du résultat**

Ex 2.6 Enoncé

Une bonne pizzeria est une pizzeria qui n'est pas trop chère !

5 de vos amis ont été dans la même pizzeria.

- Le premier était accompagné de trois autres amis, ils ont commandé 1 Margherita, 2 Quatre-saisons et 1 végétarienne et en ont eu au total pour **55 €**
- Le second qui a trois enfants est allé en famille avec Madame. Ses 2 filles ont choisi 1 pizza Margherita chacune, sa femme 1 Quatre-saisons, lui 1 végétarienne et son fils oh sacrilège 1 pizza Hawaïenne. Ils ont payé **65,5 €**
- Le troisième a payé au total **80 €** pour 1 Margherita, 2 Hawaïenne, 2 végétariennes et 1 Napolitaine ;
- Le quatrième a emporté 3 Napolitaine, 1 Hawaïenne, 2 Margherita, 1 végétarienne et 2 Quatre-saisons pour **117,5 €**
- Le cinquième avec ses deux frères a mangé 2 Napolitaines, 1 Margherita et 2 Hawaïennes pour un total de **63,5 €**

Pouvez-vous en utilisant le calcul matriciel retrouver le prix des différents types de pizza?

Réponses (pour vérification):

- La pizza végétarienne ? **14.5**
- La pizza Hawaï ? **13.5**
- La pizza Quatre-saisons ? **14.5**
- La pizza Margherita ? **11.5**
- La pizza Napolitaine ? **12.5**

Exercices Cours/diaporama 6

Convolution

Convolution avec scipy

Numpy.convolve ne s'applique qu'à des tableaux à une seule dimension (vecteur).
Pour pouvoir faire une convolution sur une matrice, on va utiliser la fonction convolve de Scipy :

```
from scipy.ndimage import convolve
convolve(data, kernel)
```

Doc: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve.html

On peut aussi utiliser la fonction

```
scipy.signal.convolve2d
```

qui fera le calcul plus rapidement (car elle est implémentée en C compilé) et possède d'autres options pour les pixels des bords.)
Doc: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html

Exercice de traitement d'images par convolution

1. Ecrire votre propre fonction/ programme qui fait la convolution entre une matrice de grande taille et un masque dont les noms sont passés en paramètre.

2. Tester le bon fonctionnement en l'utilisant sur la matrice et le masque de l'exemple page 10 et 11 du syllabus sur les tenseurs.

3. Idem mais avec la fonction convolve de Scipy. Le résultat est-il identique ? Pourquoi ?

4. Implanter un filtre de flou linéaire avec Scipy ou votre programme par convolution avec un masque de 3x3 puis de 10x10 et appliquez-le à une image 512x512

5. Implanter un filtre de flou Gaussien avec Scipy ou votre programme et appliquez-le à une image 512x512

6. Imaginez le masque d’un filtre de flou directionnel de gauche à droite (comme si la personne ne bougeait que dans cette direction) et testez-le

7. Imaginez un filtre de flou basé sur la médiane (filtre médian non linéaire)

8. Créez un filtre de piqué utilisant le masque proposé dans le syllabus

9. Programmez un filtre de détection de contour basé sur le gradient

10. Programmez un filtre de détection de contour basé sur le Laplacien avec ou sans pré-floutage

Exercices Cours/diaporama 7

Graphes 1-2-3

Algorithme de Dijkstra

Parcours en largeur

Parcours en profondeur

Labyrinthe

Exercices Cours/diaporama 11

AI – Reconnaissance de visages

Étapes concrètes du programme openCV Cascade de Haar

Pour détecter des visages de chats dans une image et dessiner des boîtes de délimitation autour d'eux, vous pouvez suivre les étapes ci-dessous

- Importez la bibliothèque OpenCV ou ssurez-vous de l'avoir déjà installée.
- Lisez l'image d'entrée en utilisant `cv2.imread()`. Spécifiez le chemin complet de l'image.
- Convertir l'image d'entrée en niveaux de gris:
  `image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`
- Lancer un objet classificateur en cascade Haar: `cat_cascade = cv2.CascadeClassifier()` pour la détection des visages de chats. Transmettre le chemin complet du fichier xml de la cascade de Haar. Vous pouvez utiliser le fichier haar cascade `haarcascade_frontalcatface.xml` pour détecter les visages de chats dans l'image.
- Détectez les visages de chat dans l'image d'entrée à l'aide de `cat_cascade.detectMultiScale()`. Cette fonction renvoie les coordonnées des visages de chat détectés au format (x,y,w,h).
- Dessinez les rectangles de délimitation autour des visages de chats détectés dans l'image originale à l'aide de `cv2.rectangle()`.
- Affichez l'image avec les rectangles de délimitation et un titre dessiné autour des visages de chat: `cv2.imshow()`

Fichier sur Teams: detection-chat.py

Exercices Cours/diaporama 12

AI – Reconnaissance de chiffres

Exemple avec sklearn

Exemple avec sklearn: Création du réseau et préparation de l'échantillon d'entraînement ¶

scikit-learn permet de créer, d'entraîner, et d'évaluer un réseau de neurones en quelques lignes de code.

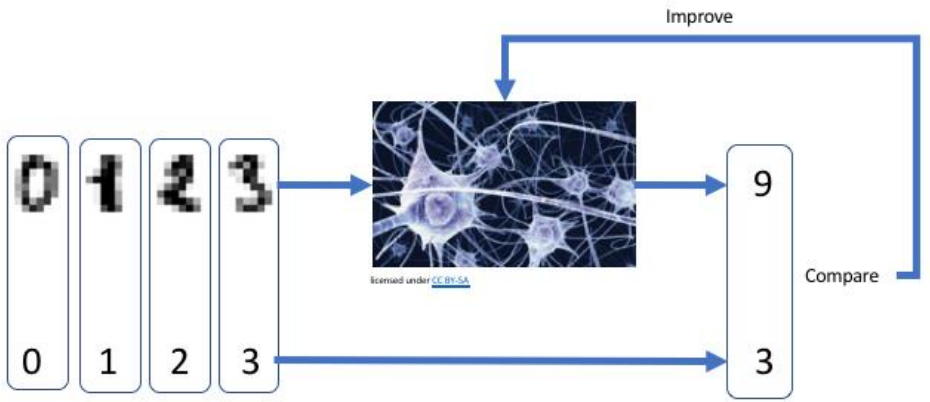
Nous allons créer un réseau extrêmement simple, avec seulement trois couches:

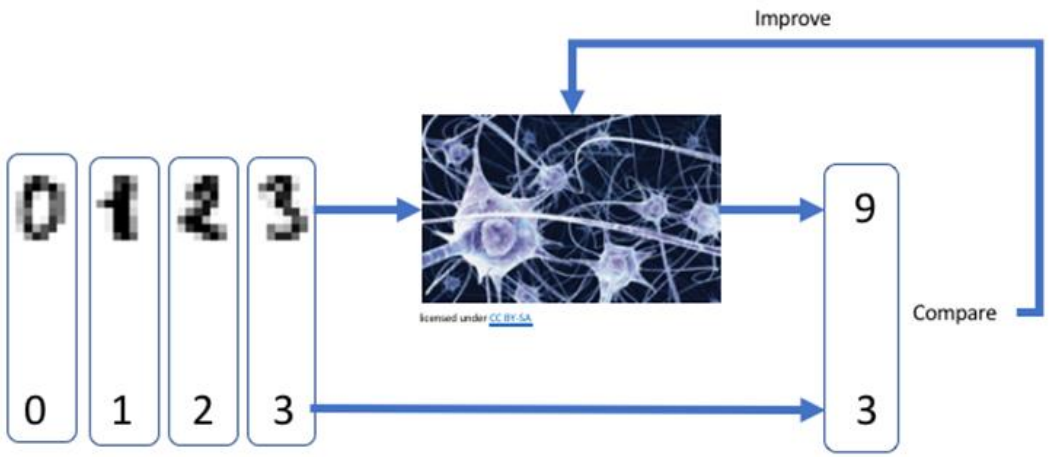
une couche d'entrée (input layer), avec un noeud par pixel dans les images 8x8, 64 noeuds, donc) (neurone qui ne fait rien, il se contente de prendre la valeur en entrée et de la transmettre aux neurones de la couche suivante).

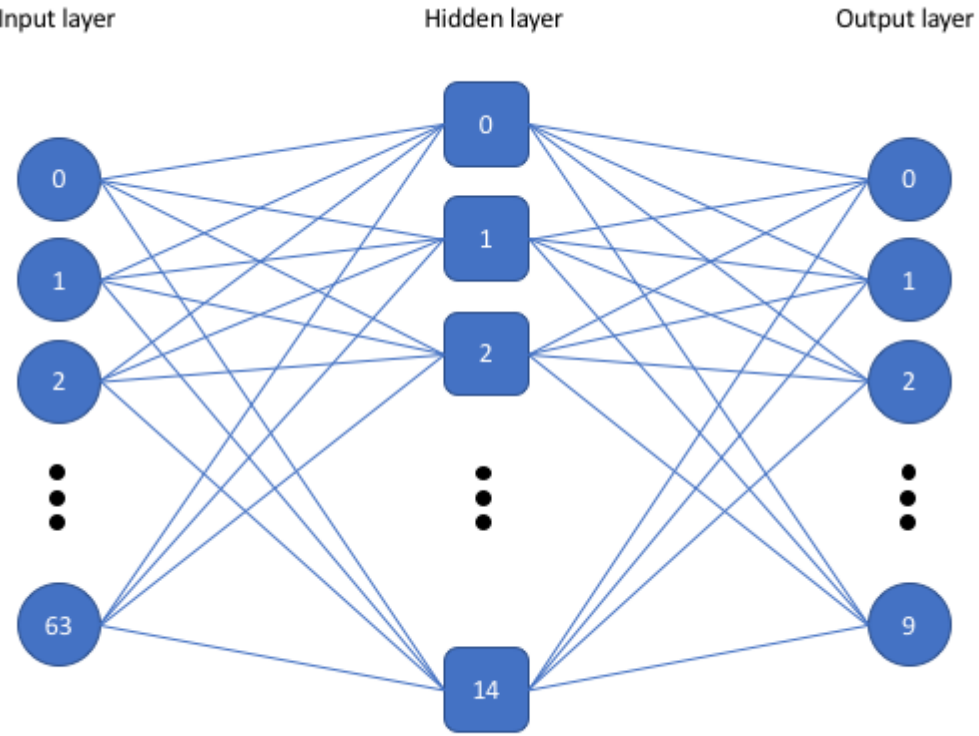
une couche cachée (hidden layer) avec 15 neurones.
Nous aurions pu choisir un autre nombre de neurones, et ajouter des couches avec différents nombres de neurones.

une couche de sortie (output layer) avec 10 neurones, correspondant à nos 10 classes de chiffres, de 0 à 9

On dit que ce type de réseau est dense , ce qui v est connecté avec tous les neurones des couches p




Improve
0
1
2
3
licensed under CC BY-SA
9
3
Compare


Input layer
Hidden layer
Output layer
0
1
2
63
0
1
2
14
0
1
2
9

Un échantillon des données du dataset représenté sous forme d’images