EXERCICE 1

Partie A : Création d'une image

Dans cette partie on crée une image de taille 100×100 pixels qui contient un cercle jaune de centre et de rayon donné sur fond bleu.

1. Supposons que l'on dispose d'un tableau T qui représente une image. $[i, j]$ représente la position d'un pixel dans l'image.
   Définir une fonction `distance` qui calcule la distance (en pixels) entre deux pixels. On conviendra que le premier pixel a pour position $[i, j]$ et le deuxième $[k, l]$.
2. On note $[c1, c2]$ la positon du centre d'un cercle dans l'image et R son rayon.
   Compléter puis tester la fonction `soleil` qui prend en argument `c1, c2, R` qui crée une image représentant un disque jaune sur fond bleu de position $[c1, c2]$ et de rayon R.
   Remarque : le jaune s'obtient avec autant de rouge que de vert et pas du tout de bleu.

```
def soleil(c1, c2, R):
    sol = np.zeros((... , ... ,3), dtype=np.uint8)
    for i in range(...):
        for j in range(...):
            if ..........................:
                sol[i,j] = ....................
            else:
                sol[i,j] = ....................
    plt.imshow(sol)
    plt.show()
```

3. Ajouter les arguments `couleurSol` et `couleurCiel` à la fonction précédente et modifier le corps de celle-ci pour qu'elle affiche un cercle de couleur `couleurSol` sur fond `couleurCiel`. Tester.

Partie B : Animation

On peut animer ou (et) changer la couleur d'une image à l'aide d'une simple boucle.
Pour cela il suffit de remplacer dans le corps de la fonction `soleil`, `plt.show()` par `plt.pause(t)` où `t` représente le temps de pause entre deux appels successifs à la fonction `soleil`.
On peut prendre `t = 0.1` par exemple.
De plus, pour éviter que la fenêtre ne se fige à la fin de l'exécution de la boucle on ajoute `plt.show()` après et hors de la boucle.

Exemple :

```
for k in range(10):
    soleil(30, 40 - 2*k, 20 + k, [10*k,10*k,0], [0, 0, 10*k])

plt.show()
```

Attention : La position d'un pixel ne correspond aux coordonnées dans le repère affiché. En effet, le premier nombre de la positon représente le numéro de ligne du tableau. Il s'agit donc de l'ordonnée du pixel correspondant dans le repère affiché. Le deuxième nombre de la position est l'abscisse. D'autre part, à l'affichage il faut tenir compte du fait que l'axe des ordonnées est dirigé vers le bas.

1. Sans écrire le code de l'exemple précédent et en tenant compte de la remarque ci-dessus, essayer de prévoir ce qu'il va se passer lors de l'exécution de l'exemple ci-dessus.
2. Vérifiez vos affirmations en testant ce code.
3. Modifier le code pour que le soleil parte du bas de la fenêtre et aille vers le haut, sans changer son rayon ni sa couleur.
4. Modifier à nouveau le code pour que le soleil passe du orange au jaune en allant vers le haut.
   Remarque : pour passer du jaune à l'orange ou inversement il suffit d'augmenter ou de diminuer la quantité de rouge.
5. Modifier encore le code pour que la couleur de fond passe du bleu foncé au bleu clair.
6. Pour finir modifier la trajectoire du soleil pour simuler son parcours dans le ciel du lever au coucher.
   Remarque : on pourra choisir une trajectoire parabolique de sorte que le centre du soleil soit sur une courbe d'équation $y = a(x-b)^2+c$ où a, b et c sont à choisir judicieusement.

EXERCICE 2

Dans les exercices qui suivent on utilisera la trame suivante pour importer une image et la manipuler :

```
import numpy as np
import matplotlib.pyplot as plt

Image_enregistree = plt.imread("Chemin acces image/Nom image.jpg")

def fonction(im):
    resultat = np.copy(im)
    #Ici le travail sur l image
    plt.imsave("Chemin acces image/Nom nouvelle image.jpg" , resultat)
```

Il s'agit, dans cet exercice, de créer une image en noir et blanc à partir d'une image en couleur. Une méthode possible est de faire, pour chaque pixel, la moyenne des trois composantes R, V et B, puis si le nombre obtenu est inférieur à 128 on décide que le pixel étudié est noir, sinon il est blanc.

1. Ecrire une fonction `enNoirEtBlanc` qui prend en argument une liste `Couleur` qui représente une couleur et qui renvoie 0 ou 1 selon que le pixel résultant doit être noir ou bien blanc.
2. Ecrire une fonction `ConversionNB` qui prend en argument une image et qui la convertit en noir et blanc.

EXERCICE 3

Pour modifier le contraste d'une image il suffit, pour chaque composante couleur, de s'éloigner ou de se rapprocher de la valeur 128.
Plus précisément, si on veut augmenter le contraste, si un pixel a une composante inférieure à 128 on diminue cette valeur, sinon on l'augmente.

1. Ecrire une fonction `contraste20` qui prend en argument une liste de trois entiers qui représente une couleur et qui modifie leur valeur selon la règle ci-dessus en augmentant ou en diminuant la valeur de 20.

   La fonction doit renvoyer une liste correspondant à la couleur modifiée.

   Attention : pensez à gérer le fait de ne pas passer en dessous de 0 et de ne pas dépasser 255.

2. Tester sur des images en utilisant une fonction `conversionContraste` similaire à celle de l'exercice 2.

3. Dans cette question il s'agit de gérer le niveau de contraste. Pour cela on ajoute un paramètre `k` à la fonction `contraste` qui diminue ou augmente celui-ci.

   On utilisera la fonction affine suivante où x représente une composante couleur (comprise entre 0 et 255) et $f(x)$ la nouvelle composante avec :

   — $f(x) = k(x - 127,5) + 127,5$ dans le cas où $0 \leqslant f(x) \leqslant 255$.
   — Si on obtient $f(x) < 0$ ou $f(x) > 255$ avec la formule précédente, on met la composante à 0 ou à 255 selon les cas.

   De cette manière, si $k = 0$ le contraste est nul : on obtient une image uniformément grise. Si $k = 1$ l'image est inchangée, si $k < 1$ le contraste est atténué et si $k > 1$ le contraste est augmenté.

   (a) Ecrire une fonction `f` qui prend en argument x et k et qui renvoie la nouvelle valeur selon la règle ci-dessus.

   (b) Ecrire une fonction `contrastek` qui prend en argument une liste de trois entiers entre 0 et 255 et le paramètre `k` et qui renvoie la nouvelle composante selon la règle ci-dessus.

   (c) Modifier la fonction `conversionContraste`, en l'appelant `conversionContrastek` et en l'agrémentant du paramètre k.