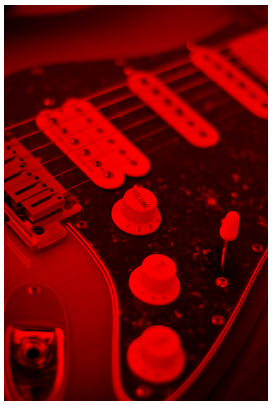
Objectif : Réaliser une production informatique, au format pdf, qui devra contenir les programmes, des explications, des schémas et des exemples choisis pour relever 3 défis sur les 5 proposés.
Lire les aides en fin de projet pour vous aider à réaliser les défis 1 à 4.

Défi 1 : Filtre Rouge

Appliquer un filtre ne laissant passer que la couleur rouge d'une image

Défi 2 : Transformer une image couleur (RGB) en niveaux de gris

Transformer une image en couleur en calculant la moyenne des 3 composantes RVB de chaque pixel et stocker cette (même) valeur dans chacune des composantes.
Comparer les résultats obtenus avec ce calcul de moyenne pondérée:
moyenne = 0,2126 × *R* + 0,7152 × *V* + 0,0722 × *B*.


Défi 3 : Effet de pixellisation

La pixellisation est souvent utilisée pour flouter une image, en partie ou dans sa totalité. La zone pixellisée fait apparaître des blocs carrés contenant N × N pixels de couleur uniforme. On observe la même chose en agrandissant une image jusqu'au niveau du pixel, sauf que là, chaque carré représente 1 pixel.

Méthode :

- Balayer l'image non pas pixel par pixel, mais par blocs de N×N pixels à l'aide d'une double boucle (commencer avec N=20 pixels pour tester).
- Pour chaque bloc, calculer la moyenne des composantes R, V, et B des pixels.
- Remplacer les composantes R, V, et B de chacun des pixels par la valeur moyenne (une moyenne de valeurs par composante afin de respecter la teinte moyenne du bloc).

Exemple avec des niveaux de gris :

	0	1	2	3
0	172	127	103	109
1	134	85	65	67
2	120	78	56	41
3	120	61	49	40

extrait (16 pixels) d'une image en niveaux de gris

Traitement d'un premier bloc 2x2 (en vert) : on prend la moyenne des valeurs : (172 + 127 + 134 + 85)/4 = 130
On donne cette valeur aux 4 pixels de l'image initiale.
On passe ensuite au second bloc (en rouge) et on opère de même.
Les 4 pixels du bloc rouge prennent la valeur de la moyenne (86).
De même pour les blocs bleus et jaunes (dont les pixels prennent respectivement la valeur 95 et 47).
On obtient ainsi la nouvelle image ci-contre.

	0	1	2	3
0	130	130	86	86
1	130	130	86	86
2	95	95	47	47
3	95	95	47	47

Défi 4 : Effet Warhol

Transformer une photo couleur en noir et blanc : Pour cela, choisir un seuil permettant de définir si le pixel sera noir OU blanc (il n'y aura pas de niveaux de gris)
Appliquer ensuite un effet de couleur et de duplication pour obtenir une nouvelle image composée de 4 images en mosaïques comme sur l'exemple de droite.

Défi 5 : Stéganographie

Le principe de la stéganographie consiste à cacher une image dans une autre image.

Objectifs

1. Écrire un programme permettant de cacher une image (au format png) dans une autre image (au format png).
2. Écrire un programme permettant à un tiers de récupérer l'image cachée (*qui s'affichera lors de l'exécution du programme*).

Méthode employée pour cacher l'image

L'essentiel de l'information définissant une image se trouve dans les bits de poids forts des composantes **RGB**.
Il faut donc être capable de concevoir une image dont les composantes **RGB** de chaque pixel seront constitués par :

les quatre bits de poids forts seront les quatre bits de poids forts d'une des deux images de départ

les quatre bits de poids faibles seront les quatre bits de poids fort de l'autre image de départ.

Pour garder les 4 bits ' de poids forts' on pourra utiliser (vu en SNT : réseau local d'internet) un masque de valeur décimale : 240 soit valeur binaire : 11110000. Puisqu'ainsi l'opération « **1001**1111 AND 11110000 » donne en binaire **1001**0000.

Indications et précisions

Les fonctions ord() et chr() du langage Python vous seront utiles.
La méthode **.getdata()** permet de récupérer sous forme de liste l'intégralité des triplets des composantes couleurs des pixels de l'image.
Une troisième technique pour parcourir une liste : for _ _ _ in enumerate _ _ _ :

Aide 1 : Afficher une image à partir d'un fichier

```
# La librairie PIL doit être importée en début de programme
from PIL.Image import *
from PIL.ImageDraw import *
im = open("nom_du_fichier_image.jpg")     # "im" ou tout autre nom de variable
im.show()
```

Aide 2 : Sauvegarder une image dans un fichier

```
im.save("nom_du_fichier_à_générer.jpg")
```

Aide 3 : Récupérer les valeurs des 3 composantes Rouge, Vert et Bleu d'un pixel d'une image en utilisant la fonction **getpixel**

```
# On s'intéresse au pixel de coordonnées (10,42).
# L'origine est située en haut à gauche.
x = 10
y = 42
# Lecture des 3 composantes de la couleur du pixel de coordonnées (x,y) et affectation aux variables r, v et b
(r,v,b) = im.getpixel((x,y))
print (r, v, b)  # si l'on souhaite afficher les valeurs
```

Aide 4 : Modifier la couleur d'un pixel en utilisant la fonction **putpixel**

```
# Modifie la couleur du pixel en ligne 10 et colonne 42 en choisissant r=133, v=122 et b=191
x = 10
y = 42
(r,v,b) = (132,122,191)
im.putpixel((x,y),(r,v,b))
```

Aide 5 : Parcourir les pixels d'une image pour réaliser le traitement

```
# Récupération de la largeur et de la hauteur de l'image
(largeur, hauteur)= im.size
# Parcourir l'image, pixel par pixel avec 2 boucles for imbriquées
for x in range(largeur):
        for y in range(hauteur):
                # Ici on traite le pixel (x,y) de l'image
```

Aide 6 (effet Warhol) : Créer une image vide

```
taille = (640, 480)                          # Dimensions de l'image
couleur = (50, 255, 137)                     # Couleur du fond de l'image
im = Image.new("RGB", taille, couleur)       # Création de l'image
im.save("nom_du_fichier_à_générer.jpg")      # Sauvegarde de l'image
```

Aide 2 : Sauvegarder une image dans un fichier

```
im.save("nom_du_fichier_à_générer.jpg")
```

Aide 3 : Récupérer les valeurs des 3 composantes Rouge, Vert et Bleu d'un pixel d'une image en utilisant la fonction **getpixel**

```
# On s'intéresse au pixel de coordonnées (10,42).
# L'origine est située en haut à gauche.
x = 10
y = 42
# Lecture des 3 composantes de la couleur du pixel de coordonnées (x,y) et affectation aux variables r, v et b
(r,v,b) = im.getpixel((x,y))
print (r, v, b)  # si l'on souhaite afficher les valeurs
```

Aide 4 : Modifier la couleur d'un pixel en utilisant la fonction **putpixel**

```
# Modifie la couleur du pixel en ligne 10 et colonne 42 en choisissant r=133, v=122 et b=191
x = 10
y = 42
(r,v,b) = (132,122,191)
im.putpixel((x,y),(r,v,b))
```

Aide 5 : Parcourir les pixels d'une image pour réaliser le traitement

```
# Récupération de la largeur et de la hauteur de l'image
(largeur, hauteur)= im.size
# Parcourir l'image, pixel par pixel avec 2 boucles for imbriquées
for x in range(largeur):
        for y in range(hauteur):
                # Ici on traite le pixel (x,y) de l'image
```

Aide 6 (effet Warhol) : Créer une image vide

```
taille = (640, 480)                          # Dimensions de l'image
couleur = (50, 255, 137)                     # Couleur du fond de l'image
im = Image.new("RGB", taille, couleur)       # Création de l'image
im.save("nom_du_fichier_à_générer.jpg")      # Sauvegarde de l'image
```