EXERCICES

Pour vérifier ses acquis

manuel p. 118

1 Vrai ou faux ?

a. Faux (rouge, vert, bleu.)

b. Faux (entre 0 et 255.)

c. Vrai

d. Vrai

e. Faux (Elle prend moins de place.)

f. Vrai

2 QCM

a. Le code couleur pour le vert est (0,255,0).

b. La définition d'une image numérique est exprimée en pixels.

c. Pour enregistrer une image dans un format qui prend le moins de place possible, il faut choisir l'extension Jpg.

d. Les filtres des photosites sont répartis en carré de quatre : deux verts, un rouge, un bleu.

3 Qui suis-je ?

a. Les petits carrés monochromes formant une image numérique sont les **pixels**.

b. Les données comme la date, la géolocalisation ou les réglages de l'appareil sont enregistrées dans un fichier au format **EXIF**.

Pour s'entraîner

manuel p. 118-119

4 Qualité des images imprimées

• Recopier et compléter le tableau en indiquant la qualité de la photo (bonne, moyenne ou mauvaise) selon sa définition et ses dimensions.

Il faut diviser la définition par l'aire, en cm^2, de la photographie pour estimer le nombre de pixels par cm^2.

Dimensions / Définition	11 × 15 (en cm)	15 × 20 (en cm)	20 × 27 (en cm)
10 mégapixels	Bonne (environ 60 606 pixels/cm²)	Bonne (environ 33 333 pixels/cm²)	Bonne (environ 18 519 pixels/cm²)
5 mégapixels	Bonne (environ 30 303 pixels/cm²)	Bonne (environ 16 666 pixels/cm²)	Moyenne (environ 9 259 pixels/cm²)
3 mégapixels	Bonne (environ 18 181 pixels/cm²)	Moyenne (10 000 pixels/cm²)	Moyenne (5 556 pixels/cm²)
800 000 pixels	Moyenne (4 848 pixels/cm²)	Mauvaise (2 667 pixels/cm²)	Mauvaise (1 481 pixels/cm²)

5 Choisir le bon capteur

1. Calculer la définition de chaque capteur (arrondir au million de pixels). Quelle est la différence entre le capteur B et le capteur C ?

Définition du capteur A :

5 908 × 4 400 = 26 millions de pixels (environ).

Définition des capteurs B et C : 23 millions de pixels (environ).

Les capteurs B et C n'ont pas la même dimension.

2. Sachant que plus un photosite est petit, moins il est sensible à la lumière, quel est le capteur permettant le meilleur rendu parmi les trois ?

Le capteur le plus grand est celui qui aura le meilleur rendu : le capteur B.

3. Léa souhaite acheter un appareil qui lui permette de réaliser des photos qu'elle imprimera en grand format pour en faire des affiches. Tom souhaite pouvoir faire de jolies photos par temps couvert, lorsque la luminosité est faible. Quels sont les appareils les plus adaptés aux besoins respectifs de Léa et de Tom ?

Léa a besoin d'une plus grande définition : elle doit choisir le capteur A. Tom a besoin d'un grand capteur : il doit choisir le capteur B.

6 Données EXIF

• Associer les images à leur fichier de métadonnées.

Image A → Fichier 2

Image B → Fichier 3

Image C → Fichier 1

7 Étapes de la prise de vue

• Classer dans un ordre cohérent les principales étapes permettant la construction d'une image numérique finale.

b. Mise au point, stabilisation.

d. Capture des valeurs R, V, B.

e. Enregistrement au format Raw.

a. Enregistrement des données EXIF.

c. Correction de la saturation.

f. Enregistrement au format Jpeg.

8 Profondeur de couleur

1. Quelle est la mémoire requise pour enregistrer une image dont la définition est de 2,4 Mpx en résolution *color* (8 bpp) ? Par combien faut-il multiplier ce résultat pour obtenir la mémoire nécessaire en *highcolor* ? en *truecolor* ?

En *color*, il faut : 2 400 000 × 8 = 19 200 000 bits (environ 19 milliards de bits).

Il faut le double en *highcolor* et le triple en *truecolor*.

2. Combien d'images de 2,4 Mpx avec une profondeur de couleur de 8 bits peut-on stocker dans une carte mémoire de 16 milliards de bits (16 gigabits) ?

16 000 000 000/19 200 000 = environ 833 photos.

Pour s'entraîner à programmer

manuel p. 120-121

9 Tests de luminosité

1. Qu'affiche la fonction `testLuminosite` pour les valeurs suivantes ?

a. R = 50, V = 20 et B = 10

La fonction affiche "C'est un pixel foncé" car R+V+B=80<150.

b. R = 80, V = 30 et B = 120

La fonction affiche "Ce n'est pas un pixel foncé" car R+V+B=230>150.

2. On souhaite repérer les pixels clairs, c'est-à-dire ceux dont la somme des valeurs est supérieure à 500. Remplacer les lignes 4 et 5 par une instruction conditionnelle faisant afficher si le pixel est clair. On indiquera si un pixel n'est ni clair ni foncé.

```
1  def testLuminosite(R,V,B):
2    if R+V+B<150:
3      print("C'est un pixel foncé")
4    elif R+V+B>500:
5      print("C'est un pixel clair")
6    else:
7      print("Ce n'est ni un pixel clair, ni un pixel foncé")
```

10 Calcul du nombre de bits nécessaires

1. Quel est le type de la variable `nombreCouleursSouhaitees` ?

`nombreCouleursSouhaitees` est de type entier (`int`) : c'est l'instruction `int` qui l'impose.

2. À quelle condition la boucle `while` de la ligne 3 s'arrête-t-elle ?

La boucle s'arrête dès que la valeur de la variable `nombreCouleurSouhaitees` est plus petite que ou égale à `2**nbBit` (c'est-à-dire 2 puissance le nombre de bits courant).

3. Quelle est l'utilité de l'instruction de la ligne 4 ?

C'est un compteur, il permet d'augmenter de 1 la variable `nbBit` à chaque tour de boucle. Sans cela, la boucle ne se termine jamais.

4. L'utilisateur saisit la valeur 2.

a. La boucle `while` se termine-t-elle ?

La condition de la ligne 3 est 2>2**0, c'est-à-dire 2>1 est vérifiée. Puis 2>2**1 n'est pas vérifiée : on sort de la boucle.

b. Qu'affiche le programme ?

Le programme affiche "Il vous faut un nombre de bit égal à 1".

c. Reprendre les questions **a.** et **b.** si l'utilisateur saisit la valeur 40.

Le programme affiche: "Il vous faut un nombre de bit égal à 6".

11 Modification de la luminosité

1. Le programme suivant a pour but de changer la luminosité d'un pixel en ajoutant un nombre entier compris entre – 255 et 255 à chacune des valeurs rouge, vert et bleu. Compléter les lignes 5, 6 et 7.

```
1  R = int(input("Rouge ="))
2  V = int(input("Vert ="))
3  B = int(input("Bleu ="))
4  lum = int(input("Valeur à ajouter (entre -255 et 255) ="))
5  R = R + lum
6  V = V + lum
7  B = B + lum
8  print("Nouvelles valeurs:",R,V,B)
```

2. Les valeurs des variables *R, V, B* ne peuvent pas être négatives ou dépasser 255. Les valeurs trop grandes doivent donc être ramenées à 255 et les valeurs négatives à 0. Ajouter au programme précédent, avant la ligne 8, des instructions conditionnelles permettant cette correction.

```
1  R = int(input("Rouge ="))
2  V = int(input("Vert ="))
3  B = int(input("Bleu ="))
4  lum = int(input("Valeur à ajouter (entre -255 et 255) ="))
5  R = R + lum
6  V = V + lum
7  B = B + lum
8  if R > 255:
9    R = 255
10 if R<0:
11   R = 0
12 if V > 255:
13   V = 255
14 if V<0:
15   V = 0
16 if B > 255:
17   B = 255
18 if B<0:
19   B = 0
20 print("Nouvelles valeurs:",R,V,B)
```

⑫ Définition d'une image

1. Compléter la fonction suivante afin de calculer la définition d'une image selon sa résolution (en ppp) et ses dimensions en pouces.

```
1  def calculDefinition(resolution,largeur,longueur):
2    definitionLargeur = resolution*largeur # Question 1
3    definitionLongueur = resolution*longueur # Question 1
4    return definitionLargeur,definitionLongueur
```

2. Écrire un programme qui demande à l'utilisateur la saisie de la résolution et des dimensions de l'image et qui appelle la fonction précédente.

```
1  resolution = int(input("Quelle est la résolution ?"))
2  largeur = int(input("Quelle est la largeur ?"))
3  longueur = int(input("Quelle est la longueur ?"))
4  definition = calculDefinition(resolution,largeur,longueur)
5  print("La définition est",definition)
```

3. Donner la définition d'une image de 5 pouces sur 7 en 200 ppp.

Lancer le programme, il affiche : `"La définition est 1000,1400"`.

⑬ Filtre de repérage des rouges

• Traduire en programme Python l'algorithme suivant qui permet de mettre en valeur les pixels à dominante de rouge (le rouge est la valeur la plus importante dans le triplet R, V, B).

```
1  R = int(input("Rouge ="))
2  V = int(input("Vert ="))
3  B = int(input("Bleu ="))
4  if R>V et R>B:
5    R = 255
6    V = 0
7    B = 0
8  print(R,V,B)
```

+ Prolongement possible

À l'aide du module PIL, la mise en place de cet algorithme pour l'image 'grenouille.jpg' (à ajouter) est la suivante :

```
1  from PIL import Image
2  image = Image.open("grenouille.jpg")
3  image.show()
4  L, H = image.size
5  for x in range(L):
6    for y in range(H):
7      pixel = image.getpixel((x, y))
8      if max(pixel)==pixel[0]:
```

▸

```
9       image.putpixel((x,y),(255,0,0))
10  image.show()
11  image.save('grenouilleRouge.jpg', 'jpg')
```

EXERCICES COMPLÉMENTAIRES

1 Inverser les couleurs

Le filtre *Inverser les couleurs* sur Gimp permet de créer un négatif d'une image.

1. Pour une image en niveaux de gris, les teintes foncées et claires sont échangées. On rappelle qu'un pixel en niveaux de gris est codé entre 0 et 255.

a. Quel calcul faut-il effectuer pour déterminer la valeur « inverse » d'un niveau de gris *N* donné ?

b. Compléter le programme de passage en négatif d'un pixel en niveaux de gris.

```
1  N = int(input("Niveaux de gris du pixel ?"))
2  NInverse=…
3  print("Nouvelle valeur du pixel :",NInverse)
```

2. Comment faut-il modifier le programme précédent pour créer le négatif d'une image en couleurs RVB ?

CORRIGÉ

1.a. Quel calcul faut-il effectuer pour déterminer la valeur « inverse » d'un niveau de gris *N* donné ?

Il suffit de calculer 255 – N.

b. Compléter le programme de passage en négatif d'un pixel en niveaux de gris.

```
1  N = int(input("Niveau de gris du pixel ?"))
2  NInverse=255-N
3  print("Nouvelle valeur du pixel :",NInverse)
```

2. Comment faut-il modifier le programme précédent pour créer le négatif d'une image en couleurs RVB ?

```
1  R = int(input("Niveau de rouge du pixel ?"))
2  V = int(input("Niveau de vert du pixel ?"))
```

▸