SNT — Photographie numérique

Mode d'emploi

- Un fichier par exercice, nommé `NOM_Prenom_ex05.py`, rangé dans ton dossier SNT. Les réponses écrites vont en commentaire `# ...` ou dans le traitement de texte.
- Version courte : seuls les exercices essentiels sont reproduits, avec les mêmes numéros que sur le site (où les 45 exercices sont proposés).

Partie P1 — De la lumière au fichier

Comment un appareil photo numérique transforme une scène en nombres

L'appareil photo numérique s'inspire de l'œil humain : il capte la lumière d'une scène, puis l'enregistre pour la restituer le plus fidèlement possible. Dans ce thème, tu vas comprendre comment une photo est fabriquée, puis tu vas la **modifier avec Python**.

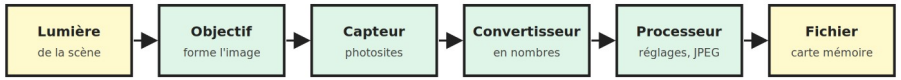
Cours · Quelques repères historiques

Regarde la vidéo « Découvrons l'histoire de la photographie », puis réponds à l'exercice 1.

- Vidéo : Découvrons l'histoire de la photographie : https://youtu.be/CtwuQynmzIA

Date	Événement
vers 1826	Nicéphore Niépce obtient la plus ancienne photographie conservée
1839	Daguerre présente le daguerréotype
1888	Kodak met la photographie à la portée du grand public (appareil à pellicule)
1975	Premier prototype d'appareil photo numérique (Kodak) : une image de 0,01 mégapixel
vers 2000	Premiers téléphones équipés d'un appareil photo

Cours · Du monde réel au fichier



Les étapes de fabrication d'une photo numérique.

- Le **capteur** est une grille de minuscules cellules, les **photosites**. Chacune mesure la quantité de lumière reçue.
- Devant chaque photosite se trouve un filtre **rouge, vert ou bleu** : l'appareil reconstitue ensuite les trois couleurs de chaque point de l'image.
- Le **convertisseur** transforme chaque mesure en **nombre**. Une photo est donc un très grand tableau de nombres.
- Le **processeur** applique des réglages, puis enregistre le fichier (le plus souvent en JPEG).

Œil humain	Appareil photo numérique
cornée et cristallin	objectif
iris (la pupille se ferme ou s'ouvre)	diaphragme
rétine (cônes sensibles au rouge, au vert, au bleu)	capteur (photosites rouges, verts, bleus)
nerf optique et cerveau	convertisseur et processeur

À retenir

Une photo numérique est un **tableau de nombres** : chaque case du tableau, un **pixel**, contient la couleur d'un point de la scène.

RVB signifie **Rouge, Vert, Bleu** : les trois couleurs utilisées pour coder chaque pixel.

Exercices · Premières questions

Exercice 1 — Petite histoire de la photographie 🌶

Regarde la vidéo, puis réponds en quelques lignes.

1. Compare la vitesse d'évolution des techniques de la photographie argentique et celles de la photographie numérique.
2. Que signifie RVB ?
3. Quel est l'ordre de grandeur du nombre de photos prises chaque jour dans le monde ?

Réponses dans ton compte rendu (exercice 1)

Exercice 3 — Du réel au fichier 🌶🌶

1. Remets dans l'ordre les étapes : enregistrement du fichier JPEG · mesure de la lumière par les photosites · lumière de la scène · conversion des mesures en nombres · passage dans l'objectif · réglages par le processeur.
2. Que mesure un photosite : une couleur ou une quantité de lumière ? Comment l'appareil obtient-il alors des photos en couleurs ?

Réponses dans ton compte rendu (exercice 3)

À retenir · Synthèse

Ce que je sais faire

- Situer quelques repères de l'histoire de la photographie.
- Décrire la chaîne qui transforme la lumière en fichier.
- Expliquer qu'une photo numérique est un tableau de nombres.

Partie P2 — Les pixels

Définition, résolution, mégapixels : ce que veulent dire les chiffres d'une photo

Une image numérique est une grille de petits carrés de couleur uniforme, les **pixels** (de l'anglais *picture element*). De près, on les voit ; de loin, l'œil les fond en une image continue.

Cours · Définition et résolution

Grandeur	Ce que c'est	Unité
Définition	le nombre de pixels de l'image : largeur × hauteur	des pixels (px) ; en mégapixels (Mpx) pour un grand nombre : 1 Mpx = 1 000 000 px
Résolution	la finesse de l'image **une fois imprimée ou affichée** : nombre de pixels par pouce	ppp (pixels par pouce), ou dpi en anglais

Un **pouce** mesure 2,54 cm. Pour une longueur donnée, la résolution se calcule ainsi :

$$\text{résolution (ppp)} = \frac{\text{nombre de pixels}}{\text{longueur en pouces}}$$

avec :

$$\text{longueur en pouces} = \frac{\text{longueur en cm}}{2{,}54}$$

Exemple : une photo de 3000 pixels de large imprimée sur 15 cm.

Longueur en pouces : $\frac{15}{2{,}54} \approx 5{,}91.$ Résolution : $\frac{3000}{5{,}91} \approx 508$ ppp.

Vigilance

La **définition** appartient à l'image (elle ne change pas quand on imprime). La **résolution** dépend de la taille du tirage : plus on agrandit, plus elle diminue.

Pour un tirage de qualité, on vise environ **300 ppp**. En dessous, les pixels commencent à se voir.

Voir les pixels

La **loupe à pixels** permet de zoomer dans une image, de voir les pixels un par un et de lire leur couleur.

- Ouvrir la loupe à pixels : <loupe-pixels.html>

Exercices · Manipuler les pixels

Exercice 4 — Définition ou résolution ? 🌶

1. Qu'est-ce que la définition d'une image ? Dans quelle unité s'exprime-t-elle ?
2. Qu'est-ce que la résolution d'une image ? Dans quelle unité s'exprime-t-elle ?

Réponses dans ton compte rendu (exercice 4)

Exercice 5 — À la loupe 🌶

Ouvre la loupe à pixels et choisis la photo « Ardèche ».

1. Augmente le zoom. À partir de quel zoom distingues-tu nettement les pixels ?
2. Place le curseur sur un pixel du ciel. Note ses coordonnées $(x\,;y)$ et ses trois valeurs R, V, B.
3. Compare avec un pixel de l'oiseau ou de l'eau. Quelles valeurs ont changé ?

Réponses dans ton compte rendu (exercice 5)

Exercice 6 — Compter en mégapixels 🌶🌶

1. Un appareil produit des photos de 4000 pixels sur 3000. Quelle est leur définition en mégapixels ?
2. La photo « Ardèche » fait 1024 pixels sur 680. Combien de pixels contient-elle ? Est-ce plus ou moins qu'un mégapixel ?

Réponses dans ton compte rendu (exercice 6)

Exercice 8 — Le poids d'une image 🌶🌶

Dans l'exercice précédent, la profondeur de couleur est de **24 bits**, soit 3 **octets** par pixel (1 octet = 8 bits).

1. Calcule le poids de l'image **sans compression** : nombre de pixels × 3 octets. Exprime-le en mégaoctets (1 Mo = 1 000 000 octets).
2. Le fichier PNG pèse 2,81 Mo. Combien de fois est-il plus léger que l'image brute ? Que peut-on en conclure sur le format PNG ?

Réponses dans ton compte rendu (exercice 8)

À retenir · Synthèse

Ce que je sais faire

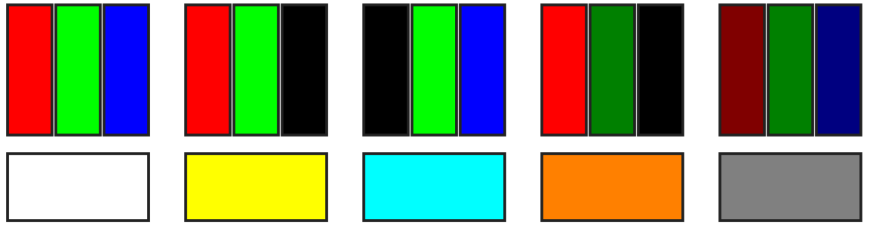
- Distinguer **définition** (nombre de pixels) et **résolution** (pixels par pouce).
- Calculer une définition en pixels ou en mégapixels.
- Calculer la résolution d'un tirage, ou la taille maximale d'un tirage à 300 ppp.
- Estimer le poids brut d'une image : pixels × 3 octets.

Partie P3 — Le codage des couleurs

Rouge, vert, bleu : 16 millions de couleurs avec trois nombres

Sur un écran, chaque pixel est formé de **trois sous-pixels** : un rouge, un vert, un bleu. En réglant l'intensité de chacun, on fabrique toutes les couleurs. À distance normale, l'œil ne distingue pas les sous-pixels : il voit une couleur unique.

Cours · Le triplet (R, V, B)



Les trois sous-pixels de quelques couleurs. Sous chaque couleur, le triplet (R, V, B).

- L'intensité de chaque sous-pixel est un nombre **entier de 0 à 255**, soit **256 valeurs** (un octet = 8 bits).
- Une couleur est donc un **triplet** $(R\,;V\,;B)$: la quantité de rouge, de vert et de bleu.
- En tout : $256\times256\times256=256^3$ couleurs possibles.

Couleur	Triplet (R, V, B)
noir	(0, 0, 0)
blanc	(255, 255, 255)
rouge · vert · bleu	(255, 0, 0) · (0, 255, 0) · (0, 0, 255)
jaune	(255, 255, 0)
rose (magenta)	(255, 0, 255)
turquoise (cyan)	(0, 255, 255)
gris	R = V = B, par exemple (128, 128, 128)

Mélanger les couleurs

Le **mélangeur RVB** permet de régler les trois curseurs et de voir la couleur obtenue, son triplet et son code hexadécimal.

- Ouvrir le mélangeur RVB : <melangeur-rvb.html>

> **Le code hexadécimal**
>
> Sur le Web, une couleur s'écrit souvent #RRVVBB. Chaque nombre de 0 à 255 est écrit avec **deux chiffres hexadécimaux** (en base 16) : `FF` vaut 255, `80` vaut 128, `00` vaut 0.
>
> Par exemple, le turquoise (0, 255, 255) s'écrit `#00FFFF`.

Exercices · Explorer les couleurs

Exercice 10 — Combien de couleurs ? 🌶

1. Chaque sous-pixel a 256 nuances. Combien de couleurs différentes peuvent être fabriquées avec les trois sous-pixels ?
2. Un triplet utilise 3 octets, c'est-à-dire combien de bits ? Retrouve le résultat de la question 1 avec une puissance de 2.

Réponses dans ton compte rendu (exercice 10)

Exercice 12 — Les couleurs de base 🌶

1. Quel triplet donne le rouge pur ? le vert pur ? le bleu pur ?
2. Trouve, avec le mélangeur, les triplets du jaune, du rose (magenta) et du turquoise (cyan).

Réponses dans ton compte rendu (exercice 12)

Exercice 13 — Prévoir la couleur 🌶🌶

Pour chaque triplet, prévois la couleur, puis vérifie avec le mélangeur.

- (120, 0, 0)
- (0, 0, 120)
- (255, 128, 128)
- (255, 128, 0)

Réponses dans ton compte rendu (exercice 13)

Exercice 15 — Du triplet au code hexadécimal 🌶🌶🌶

Pour convertir un nombre de 0 à 255 en hexadécimal, on le divise par 16 : le quotient et le reste sont les deux chiffres (les valeurs 10 à 15 s'écrivent A, B, C, D, E, F).

1. Convertis 128 puis 255 en hexadécimal. En déduis le code de l'orange (255, 128, 0).
2. Le code #3399FF est celui d'un bleu clair. Retrouve son triplet (R, V, B). Astuce : 33 vaut $3 \times 16 + 3$.

Réponses dans ton compte rendu (exercice 15)

À retenir · Synthèse

Ce que je sais faire

- Décrire une couleur par un triplet (R, V, B) de nombres entre 0 et 255.
- Retrouver noir, blanc, gris, couleurs de base et mélanges.
- Calculer le nombre de couleurs possibles : $256^3 = 16\,777\,216$.
- Convertir un nombre de 0 à 255 en hexadécimal, et l'inverse.
- Calculer un niveau de gris.

Partie P4 — Pixel art avec Python

Dessiner pixel par pixel avec la bibliothèque Pillow

Tu as vu qu'une image est une grille de pixels. Avec Python, tu vas **colorier ces pixels un par un**, grâce à la bibliothèque **Pillow**. Tu réinvestis les boucles `for` et les conditions `if` du cours Python.

> **Fichiers du thème**
>
> Les photos utilisées dans la suite (Ardèche, Eiffel, Gargouille…) sont dans l'archive PhotoNumerique.zip. Décompresse-la dans ton dossier SNT : tes programmes doivent être **dans le même dossier** que les photos.

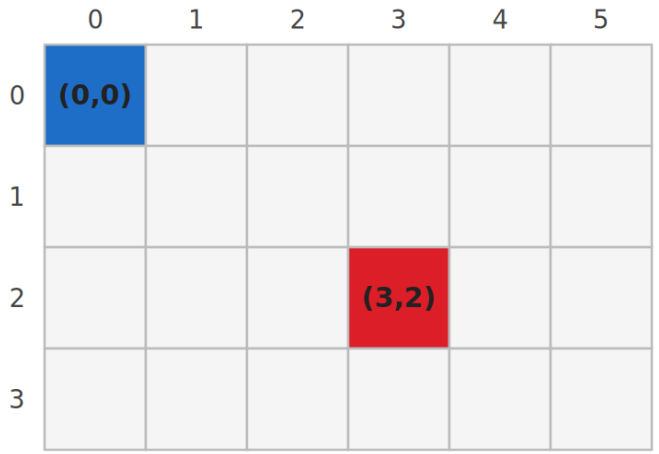
Méthode · Installer Pillow

À faire une seule fois par ordinateur. La bibliothèque s'appelle **Pillow**, mais on l'utilise avec le nom `PIL`.

1. Dans EduPython, menu *Outils → Outils → Installation d'un nouveau module*.
2. À la question « Votre choix : », tape 2 puis Entrée.
3. À la question « Nom du module : », tape `pillow` puis Entrée.
4. Attends la fin de l'installation, puis appuie sur une touche quand c'est demandé.

Cours · Repérer un pixel

Un pixel est repéré par deux coordonnées $(x\,;y)$. L'**origine** $(0\,;0)$ est le pixel **en haut à gauche**. L'axe des x va vers la **droite**, mais l'axe des y va vers le **bas**.



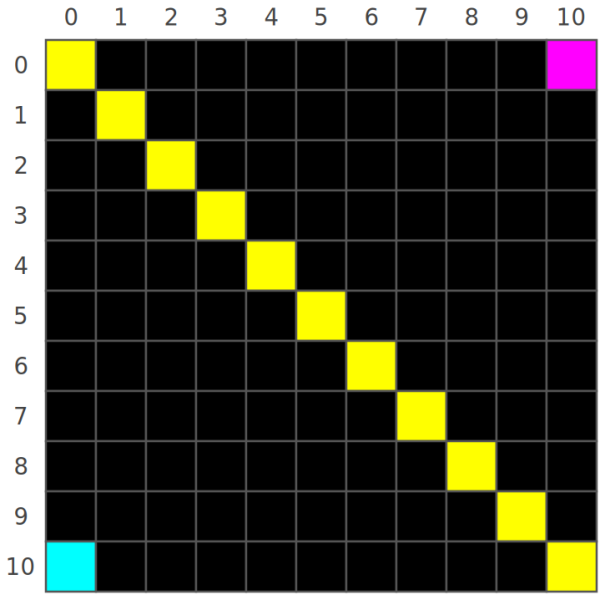
Coordonnées des pixels : y augmente vers le bas.

essai.py

```
from PIL import Image

image = Image.new("RGB", (11, 11))              # image 11 x 11, tout en noir
image.putpixel((0, 10), (0, 255, 255))          # pixel (0,10) en turquoise
image.putpixel((10, 0), (255, 0, 255))          # pixel (10,0) en rose
for i in range(11):
    image.putpixel((i, i), (255, 255, 0))       # diagonale jaune

image.resize((330, 330), Image.NEAREST).show()  # agrandit 30 fois pour la voir
image.save("Essai.png")
```



Le résultat de `essai.py`, agrandi.

Ligne	Effet
`Image.new("RGB", (11, 11))`	crée une image couleur de 11 pixels de large et 11 de haut, **noire** par défaut
`Image.new("RGB", (11, 11), (255, 255, 255))`	la même image, avec un **fond blanc**
`image.putpixel((x, y), (R, V, B))`	colorie le pixel $(x\,;y)$ avec le triplet $(R\,;V\,;B)$
`image.resize((330, 330), Image.NEAREST).show()`	affiche une version **agrandie** : sans elle, une image de 11 pixels est minuscule à l'écran
`image.save("Essai.png")`	enregistre l'image dans le dossier du programme

Vigilance

Les coordonnées vont de **0 à largeur − 1** : dans une image de 11 pixels de large, le dernier pixel est le numéro 10. Écrire `(11, 0)` provoque une erreur.

Les valeurs de R, V, B doivent être des **entiers** de 0 à 255. Un nombre décimal comme `127.5` provoque une erreur : utilise `//` ou `int(...)`.

Tracer des lignes et des rectangles

- **Horizontale** à la hauteur 5 : `for i in range(11): image.putpixel((i, 5), couleur)`.
- **Verticale** à 7 pixels du bord gauche : `putpixel((7, i), couleur)`.
- **Diagonale montante** : `putpixel((10 - i, i), couleur)`.
- **Segment** : donne deux valeurs à `range`, le début et la valeur qui suit la fin : `range(3, 8)` colorie les pixels 3, 4, 5, 6 et 7.

Pour un **rectangle**, on imbrique deux boucles. Par exemple, un rectangle bleu dont le coin supérieur gauche est le pixel $(1\,;2)$, de largeur 5 et de hauteur 3 :

rectangle.py

```
bleu = (0, 0, 255)
for x in range(1, 1 + 5):
    for y in range(2, 2 + 3):
        image.putpixel((x, y), bleu)
```

À retenir

Un pixel se colorie avec `image.putpixel((x, y), (R, V, B))`.

Les figures se tracent avec des **boucles** : une pour une ligne, deux imbriquées pour une surface.

On peut ranger une couleur dans une variable : `rouge = (255, 0, 0)`.

Exercices · Dessiner pixel par pixel

Exercice 17 — Recopier et comprendre

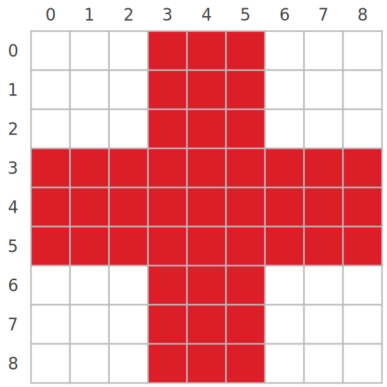
Recopie `essai.py`, exécute-le, puis réponds.

1. Sur quel bord de l'image se trouve le pixel turquoise ? le pixel rose ?
2. Quelle diagonale est tracée par la boucle : de quel coin à quel coin ?
3. Que faut-il changer pour obtenir un fond blanc ?

Fichier : NOM_Prenom_ex17.py

Exercice 18 — La croix rouge 🌶🌶

Dessine cette croix rouge sur fond blanc : une image de 9 × 9 pixels formée de cinq carrés de 3 × 3 pixels.

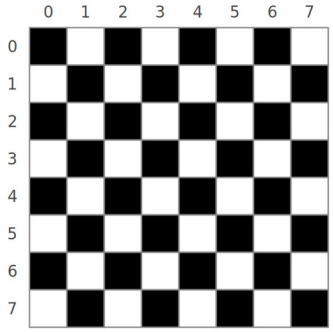


La croix à dessiner.

Fichier : NOM_Prenom_ex18.py

Exercice 20 — Le damier 🌶🌶

Dessine ce damier de 8 × 8 pixels : les pixels dont la somme $x + y$ est **paire** sont noirs, les autres blancs.



Le damier à dessiner.

Fichier : NOM_Prenom_ex20.py

À retenir · Synthèse

Ce que je sais faire

- Créer une image avec `Image.new` et colorier un pixel avec `putpixel`.
- Repérer un pixel : origine en haut à gauche, y vers le bas, coordonnées de 0 à largeur − 1.
- Tracer lignes, rectangles, damiers, dégradés et disques avec des boucles et des conditions.

Partie P5 — Filtrer les photos avec Python

Lire chaque pixel, calculer une nouvelle couleur, écrire le résultat

Un **filtre** est un petit calcul appliqué à **chaque pixel** d'une photo. Les applications de retouche font la même chose, en beaucoup plus perfectionné. Tu vas écrire tes propres filtres.

Cours · Lire, calculer, écrire

Tous les filtres de ce chapitre suivent le même modèle :

filtre.py

```
from PIL import Image

photo = Image.open("Ardeche.jpeg").convert("RGB")      # la
photo de départ
largeur, hauteur = photo.size                          # sa
définition
image = Image.new("RGB", (largeur, hauteur))           # une
image vide, de même taille

for x in range(largeur):
    for y in range(hauteur):
        R1, V1, B1 = photo.getpixel((x, y))            # 1.
lire le pixel d'origine
        R2 = V1                                        # 2.
calculer la nouvelle couleur
```

```
        V2 = B1
        B2 = R1
        image.putpixel((x, y), (R2, V2, B2))          # 3.
écrire le pixel dans la nouvelle image

image.show()
image.save("Ardeche_filtre.jpg")
```

Ligne	Effet
`photo = Image.open("Ardeche.jpeg")`	ouvre la photo, qui doit être **dans le même dossier** que le programme
`largeur, hauteur = photo.size`	range la définition de la photo dans deux variables
`R1, V1, B1 = photo.getpixel((x, y))`	lit les trois valeurs du pixel $(x\,;y)$
`image.putpixel((x, y), (R2, V2, B2))`	écrit le pixel modifié à la **même** place dans la nouvelle image

Vigilance

Le nom du fichier doit être **exact**, extension comprise : ici `Ardeche.jpeg` (avec deux e), pas `.jpg`.

`.convert("RGB")` garantit trois valeurs par pixel. Sans lui, certaines images (PNG avec transparence, images en niveaux de gris) renvoient 4 valeurs ou une seule, et `R1, V1, B1 = ...` provoque une erreur.

Les valeurs écrites doivent être des **entiers entre 0 et 255**. Pour un résultat décimal, utilise `//` ou `int(...)`. Pour un résultat qui peut dépasser, utilise `min(255, ...)` ou `max(0, ...)`.

Le programme parcourt environ 700 000 pixels : compte quelques secondes d'exécution.

À retenir

Un filtre = une **double boucle** sur tous les pixels + un calcul de $(R_2\,;V_2\,;B_2)$ à partir de $(R_1\,;V_1\,;B_1)$.

Écrire dans une **nouvelle image** évite de mélanger anciennes et nouvelles valeurs.

Exercices · Des filtres à écrire

Exercice 24 — Ouvrir une photo

Copie `photo.py`, télécharge `Ardeche.jpeg` dans le même dossier et exécute :

photo.py

```
from PIL import Image

photo = Image.open("Ardeche.jpeg").convert("RGB")
largeur, hauteur = photo.size
print(largeur, hauteur)
photo.show()
```

1. Quelle est la définition de la photo ?
2. Combien de pixels contient-elle ? Combien de mégapixels ?

Fichier : NOM_Prenom_ex24.py

Exercice 26 — Le négatif

Le **négatif** d'une photo s'obtient en remplaçant chaque intensité par son complément à 255 : $R_2 = 255 - R_1$, et de même pour V et B.

1. Écris le programme qui produit le négatif de la photo « Ardèche ».
2. Quel est le négatif du noir ? du blanc ? du rouge pur (255, 0, 0) ?
3. Que se passe-t-il si on applique le filtre deux fois de suite ? Justifie par le calcul.

Fichier : NOM_Prenom_ex26.py

Exercice 27 — Passer en niveaux de gris

1. Écris un programme qui donne à R_2, V_2 et B_2 la **moyenne** des trois intensités du pixel d'origine.
2. Écris une deuxième version avec la **moyenne pondérée** $0{,}299\,R_1 + 0{,}587\,V_1 + 0{,}114\,B_1$ (revois l'exercice sur le gris). Compare les deux images.

Fichier : NOM_Prenom_ex27.py

Exercice 28 — Noir et blanc pur 🌶🌶🌶

Un pixel devient **blanc** si son niveau de gris dépasse un **seuil**, et **noir** sinon.

1. Écris ce filtre avec un seuil de 128 (calcule d'abord le niveau de gris comme à l'exercice précédent).
2. Essaie les seuils 60 puis 200. Comment change l'image ?
3. Pourquoi cette image est-elle une image « à deux couleurs » ? Combien de couleurs différentes contient-elle ?

Fichier : NOM_Prenom_ex28.py

À retenir · Synthèse

Ce que je sais faire

- Écrire un filtre : lire un pixel, calculer, l'écrire dans une nouvelle image.
- Modifier les couleurs (permutation, négatif, gris, seuillage, éclairage) et déplacer les pixels (miroir).
- Fusionner deux images par une moyenne.
- Compter des pixels pour analyser une photo.

Partie P6 — Ce que cache une photo

Les métadonnées : appareil, date, lieu…

Un fichier photo ne contient pas seulement des pixels. Il contient aussi des **métadonnées** : des informations *sur* la photo, enregistrées automatiquement par l'appareil. Ce sont les données **EXIF**.

Cours · Les métadonnées d'une photo

Information	Exemple
l'appareil	marque et modèle (Nikon D3200)
la date et l'heure	de la prise de vue, de la dernière modification
les réglages	temps de pose, ouverture, sensibilité ISO, flash
le logiciel	un éditeur d'images, s'il a modifié la photo
le **lieu**	les coordonnées GPS, si la géolocalisation est activée

Sous Windows : clic droit sur le fichier → *Propriétés* → onglet *Détails*. Avec Python, Pillow les lit :

exif.py

```
from PIL import Image
from PIL.ExifTags import TAGS

photo = Image.open("photo_mystere.jpg")
exif = photo.getexif()
for cle, valeur in exif.items():
    print(TAGS.get(cle, cle), ":", valeur)
```

Un dictionnaire

Les métadonnées sont rangées dans un **dictionnaire** : une collection de couples *clé* → *valeur*. Par exemple `{1: "N", 2: (43.0, 56.0, 51.0)}` associe la clé 1 à la valeur `"N"`. On lit une valeur avec `dico[clé]`.

`for cle, valeur in exif.items():` parcourt tous les couples.

Les coordonnées GPS sont rangées à part. On les obtient ainsi :

gps.py

```
gps = exif.get_ifd(0x8825)
print(gps)
```

Clé	Contenu	Exemple
1	hémisphère de la latitude : `"N"` ou `"S"`	`"N"`
2	latitude en **DMS** : degrés, minutes, secondes	`(43.0, 56.0, 51.0)`
3	hémisphère de la longitude : `"E"` ou `"W"`	`"E"`
4	longitude en **DMS**	`(4.0, 32.0, 6.0)`

Vigilance

Dans les métadonnées, l'ouest s'écrit **`"W"`** (*West*, en anglais), pas `"O"`.

Beaucoup de sites de partage effacent les métadonnées des photos publiées, mais **pas tous** : un envoi par messagerie ou par courriel peut les conserver. Une capture d'écran ne les conserve pas.

Les métadonnées peuvent être **modifiées ou effacées** : leur absence ne prouve rien.

Exercices · Enquêter sur des photos

Exercice 35 — Ce que dit le fichier 🌶️

Ouvre les propriétés (onglet *Détails*) de `Eiffel.jpeg` et de `Gargouille1.jpeg`.

1. Quel appareil a pris ces photos ?
2. Repère la **date de prise de vue** et la date de **dernière modification**. Sont-elles identiques ?
3. Un logiciel d'édition est-il mentionné ? Qu'est-ce que cela indique ?

Réponses dans ton compte rendu (exercice 35)

Exercice 37 — Où a été prise la photo ? 🌶️🌶️🌶️

Avec `gps.py`, lis les coordonnées GPS de `photo_mystere.jpg`.

1. Écris un programme qui convertit la latitude et la longitude en **degrés décimaux**, avec le bon signe (ouest et sud sont négatifs), puis les affiche.
2. Ajoute une carte Folium centrée sur ce point, avec un marqueur.
3. Où la photo a-t-elle été prise ?

Fichier : NOM_Prenom_ex37.py

Exercice 38 — Photos et vie privée 🌶️🌶️

Un élève publie une photo prise dans sa chambre, sans se douter que l'appareil a enregistré les coordonnées GPS du fichier.

1. Que peut apprendre une personne malveillante en lisant les métadonnées ? Cite trois informations.
2. Donne deux bonnes pratiques pour se protéger.
3. Teste ce programme, qui enregistre la photo dans un nouveau fichier. Que constates-tu ?

propre.py

```
from PIL import Image

photo = Image.open("photo_mystere.jpg")
photo.save("propre.jpg")
print(len(Image.open("propre.jpg").getexif()))
```

Fichier : NOM_Prenom_ex38.py

À retenir · Synthèse

Ce que je sais faire

- Lire les métadonnées d'une photo, avec l'explorateur ou avec Python.
- Convertir des coordonnées GPS d'un fichier en degrés décimaux et les placer sur une carte.
- Expliquer les risques pour la vie privée et effacer les métadonnées.

Partie P7 — Vrai ou truqué ?

Retouche, montage, images générées : garder l'esprit critique

Tu sais maintenant modifier une photo pixel par pixel : tu as donc compris qu'on peut aussi **truquer** une image. Comment reconnaître une image modifiée, et que penser de ces modifications ?

Cours · Trois façons de modifier une image

Type	Ce que c'est	Exemples
Retouche	améliorer ou corriger sans changer le sens de la scène	recadrer, éclaircir, effacer les yeux rouges
Montage (manipulation)	ajouter, retirer ou déplacer des éléments, ce qui peut changer le sens de la scène	coller un objet, effacer une personne, mélanger deux photos
Image générée	image créée entièrement par un logiciel (intelligence artificielle),	une image obtenue à partir d'une phrase de

Type	Ce que c'est	Exemples
	sans scène réelle photographiée	description

Repérer un doute

- **La lumière** : les ombres et les reflets sont-ils cohérents entre les éléments de l'image ?
- **Les détails** : mains, dents, texte, motifs répétés, arrière-plan flou ou déformé.
- **La qualité** : bords trop nets, zones plus floues ou plus granuleuses que le reste.
- **L'origine** : qui a publié l'image ? Existe-t-elle ailleurs, plus ancienne ?

Vérifier

1. Chercher la **source** originale : qui est l'auteur, quand et où l'image a-t-elle été prise ?
2. Faire une **recherche d'image inversée** : on donne l'image à un moteur de recherche, qui retrouve où elle est déjà publiée.
3. **Croiser** avec d'autres sources fiables (médias, agences, sites officiels).
4. Regarder les **métadonnées**, en sachant qu'elles peuvent être absentes ou modifiées.

Vigilance

Aucun indice n'est une preuve à lui seul. Une image peut être authentique **et** sortie de son contexte : une vraie photo avec une fausse légende trompe autant qu'une image truquée.

Des étiquettes d'authenticité

Des industriels et des médias mettent en place des étiquettes qui indiquent l'origine et les modifications d'une image (par exemple la norme **C2PA**). Leur usage progresse mais reste **partiel** : beaucoup d'images n'en portent pas.

Tu approfondiras le sujet des images générées dans le thème consacré à l'intelligence artificielle.

Exercices · Cultiver son esprit critique

Exercice 39 — Retouche, montage ou image générée ? 🌶️

Classe chaque situation dans une des trois catégories du cours :

1. Recadrer et éclaircir une photo de classe.
2. Coller un requin dans une rue inondée.
3. Obtenir un paysage à partir d'une phrase décrivant la scène, sans jamais photographier ce paysage.
4. Supprimer l'effet « yeux rouges » causé par le flash.
5. Déplacer une personne d'une photo pour l'insérer sur une autre.

Réponses dans ton compte rendu (exercice 39)

Exercice 41 — Mener l'enquête 🌶️🌶️

Tu reçois dans un groupe de messagerie une photo spectaculaire : un requin nage dans une rue de ville inondée, avec la légende « Pris ce matin ! Partagez ! ».

1. Cite quatre vérifications à faire avant de la partager.
2. Explique comment fonctionne une recherche d'image inversée et à quoi elle sert ici.
3. Cette photo n'a plus aucune métadonnée. Prouve-t-elle qu'elle est truquée ?

Réponses dans ton compte rendu (exercice 41)

Exercice 42 — Acceptable ou trompeur ? 🌶️🌶️🌶️

Pour chaque situation, écris si la modification te semble **acceptable**, **à signaler**, ou **trompeuse**, et justifie en une ou deux phrases.

1. Un magazine affine la silhouette d'un mannequin sans le dire.
2. Un élève efface un touriste de ses photos de vacances pour son album personnel.
3. Un journal retire un poteau qui gênait sur une photo d'actualité, pour « faire plus propre ».

4. Une marque présente une photo d'avant / après retouchée pour vanter l'efficacité d'une crème.
5. Un photographe éclaircit légèrement une photo de paysage pour qu'elle soit plus lisible.

Réponses dans ton compte rendu (exercice 42)

À retenir · Synthèse

Ce que je sais faire

- Distinguer retouche, montage et image générée.
- Repérer des indices de modification et vérifier l'origine d'une image.
- Argumenter sur ce qui est acceptable ou trompeur.

Partie P8 — Projet : un polyptyque numérique

Inspiré des sérigraphies d'Andy Warhol : la même image, quatre filtres différents

En 1962, l'artiste américain **Andy Warhol** a créé le *Diptyque Marilyn* : une même image répétée, chaque fois avec des couleurs différentes. Tu vas créer ton **polyptyque numérique** : ta propre photo, répétée avec des filtres que tu inventes.

Cours · Assembler quatre images



image finale : 2 × largeur sur 2 × hauteur

L'image finale est deux fois plus large et deux fois plus haute que la photo.

polyptyque.py

```
from PIL import Image

photo = Image.open("mon_image.jpg").convert("RGB")
largeur, hauteur = photo.size
creation = Image.new("RGB", (2 * largeur, 2 * hauteur))

for x in range(largeur):
    for y in range(hauteur):
        R1, V1, B1 = photo.getpixel((x, y))

        creation.putpixel((x, y), (R1, V1, B1))
# zone 1 : image d'origine

        R2, V2, B2 = V1, B1, R1
# zone 2 : filtre n°1
        creation.putpixel((x + largeur, y), (R2, V2, B2))

        R3, V3, B3 = 255 - R1, 255 - V1, 255 - B1
# zone 3 : filtre n°2
        creation.putpixel((x, y + hauteur), (R3, V3, B3))

        g = (R1 + V1 + B1) // 3
# zone 4 : filtre n°3
        creation.putpixel((x + largeur, y + hauteur), (g, g,
g))

creation.show()
creation.save("NOM_Prenom_polyptyque.jpg")
```

Chaque pixel $(x\,;y)$ de la photo est lu **une fois**, puis écrit **quatre fois**, décalé de la largeur ou de la hauteur de la photo.

Des idées de filtres

Idée	Calcul
négatif	$(255-R_1\,;255-V_1\,;255-B_1)$
permutation des canaux	$(V_1\,;B_1\,;R_1)$
un seul canal	$(R_1\,;0\,;0)$ pour ne garder que le rouge

Idée	Calcul
niveaux de gris	$(g\,;g\,;g)$ avec $g=\dfrac{R_1+V_1+B_1}{3}$
deux couleurs, façon affiche	si $g>128$, une première couleur ; sinon une seconde, par exemple rose et bleu ciel
mélange avec un drapeau	moyenne de la photo et du drapeau (exercice de fusion)
pixellisation, flou	voir les exercices de pixellisation et de flou

Exercices · À toi de créer

Exercice 44 — Quatre fois la même image 🌶🌶

Recopie `polyptyque.py` avec l'image de ton choix (photo de la classe libre de droit, paysage, objet…), en donnant à `photo` le nom de ton fichier. Exécute-le.

1. Quelle est la définition de ton image finale, si la photo mesure 400 pixels sur 300 ?
2. Remplace le filtre de la zone 4 par un filtre de ton choix.

Fichier : NOM_Prenom_ex44.py

Exercice 45 — Mon polyptyque numérique 🌶🌶🌶🌶

Crée ton polyptyque : une image finale composée d'**au moins quatre images** juxtaposées, obtenues avec des **filtres que tu écris toi-même**.

- **Image de départ** : une image libre de droit, **sans visage reconnaissable**, dont tu indiques la source dans un commentaire.
- **Version de base** : quatre zones. En haut à gauche, l'image d'origine. Les trois autres reçoivent des filtres **différents**.
- **Version moyenne** : au moins un filtre mélange la photo avec **un drapeau que tu as créé toi-même** (pas le drapeau français vu en cours).
- **Version avancée** : plus de quatre zones, ou au moins un filtre que tu as **inventé** (pixellisation, flou, seuillage en deux couleurs, mélange…).

L'originalité, la difficulté des programmes et l'esthétique du polyptyque comptent dans l'évaluation. Dépose sur Pronote le programme `NOM_Prenom_polyptyque.py` et l'image finale.

Critère	Points
Le programme fonctionne, il est lisible et commenté (source de l'image indiquée)	5
Variété et originalité des filtres	6
Difficulté technique (drapeau, filtre inventé, plus de quatre zones)	5
Qualité esthétique de l'image finale	4
Total	**20**

Fichier : NOM_Prenom_polyptyque.py + NOM_Prenom_polyptyque.jpg

À retenir · Bilan du thème

Ce que tu sais faire maintenant

- Expliquer comment un appareil photo fabrique une image et la coder avec des pixels et des couleurs RVB.
- Calculer définition, résolution, poids et taille d'impression.
- Créer une image et écrire des filtres avec Python.
- Lire les métadonnées d'une photo et en mesurer les risques.
- Garder un regard critique sur les images.