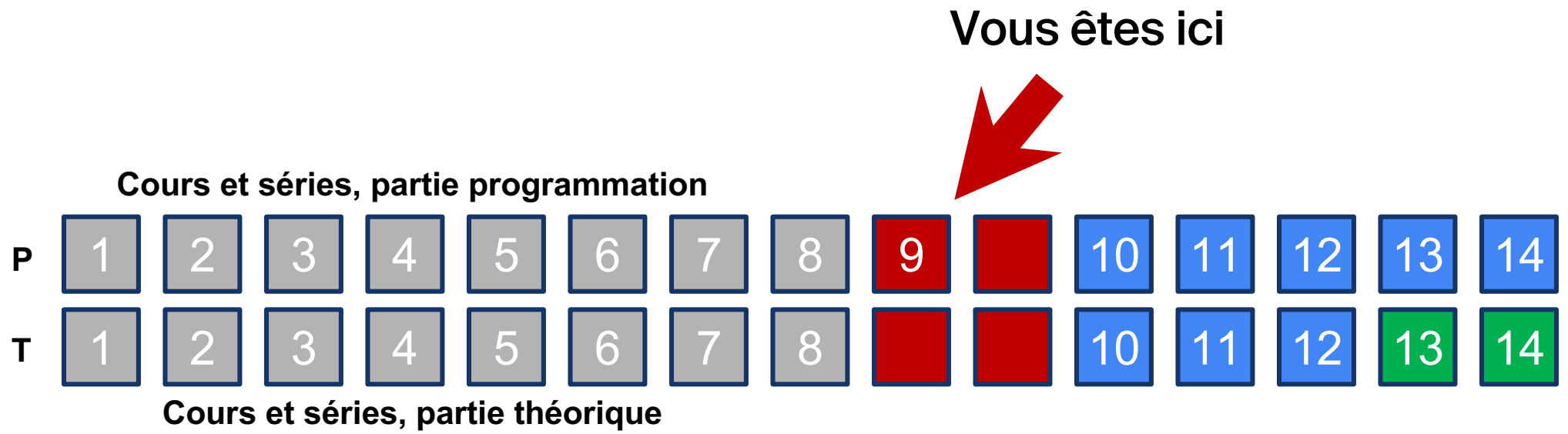
Information, Calcul et Communication

CS-119(k) ICC – Programmation
Semaine 9

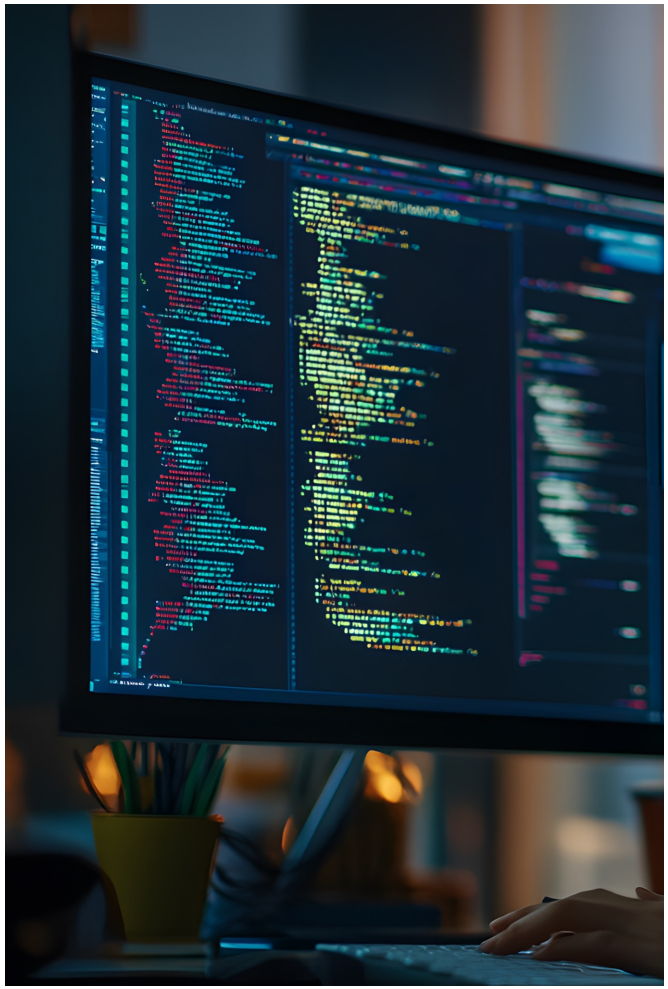
Rafael Pires
rafael.pires@epfl.ch

Lausanne, 14.04.2025

Planning


Vous êtes ici
Cours et séries, partie programmation
P
1 2 3 4 5 6 7 8 9 10 11 12 13 14
T
1 2 3 4 5 6 7 8 10 11 12 13 14
Cours et séries, partie théorique

Précédemment, dans... ICC-P



Données

- Types de base en Python (immuables) : `int`, `float`, `str`, `bool`
- Types modifiables
  - Listes
  - Sets
  - Dictionnaires
  - Dataclasses
- Fichiers texte (encodage, modélisation, lecture, écriture)

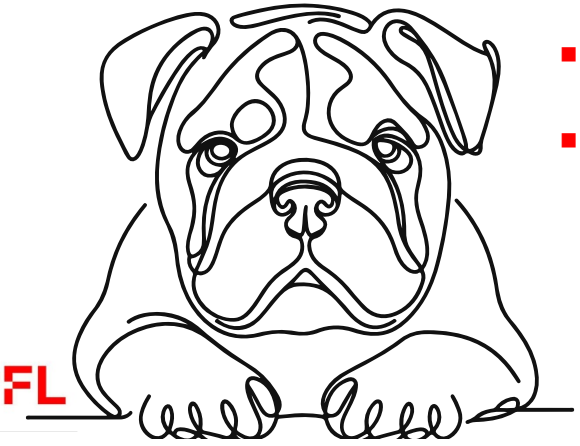
Traitement

- Méthodes, fonctions et slicing pour calculer des valeurs dérivées
- Branchements pour exécuter du code selon la valeur d'une expression booléenne
- Boucles pour exécuter du code plusieurs fois
- Fonctions

Miniprojet

Représentation des données : Images vectorielles

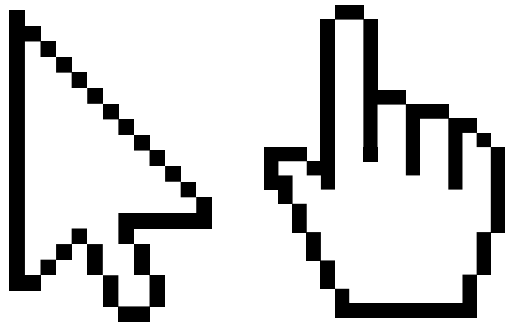
- Représentée par des formes géométriques (lignes, courbes, polygones)
- Pas de pixels : l'image est recalculée à chaque taille
- Avantage : s'adapte parfaitement au zoom
- Inconvénient : peu adaptée aux photos réalistes
- Exemples : SVG, PDF, EPS
- Logiciels : Illustrator, Inkscape, TikZ (LaTeX)



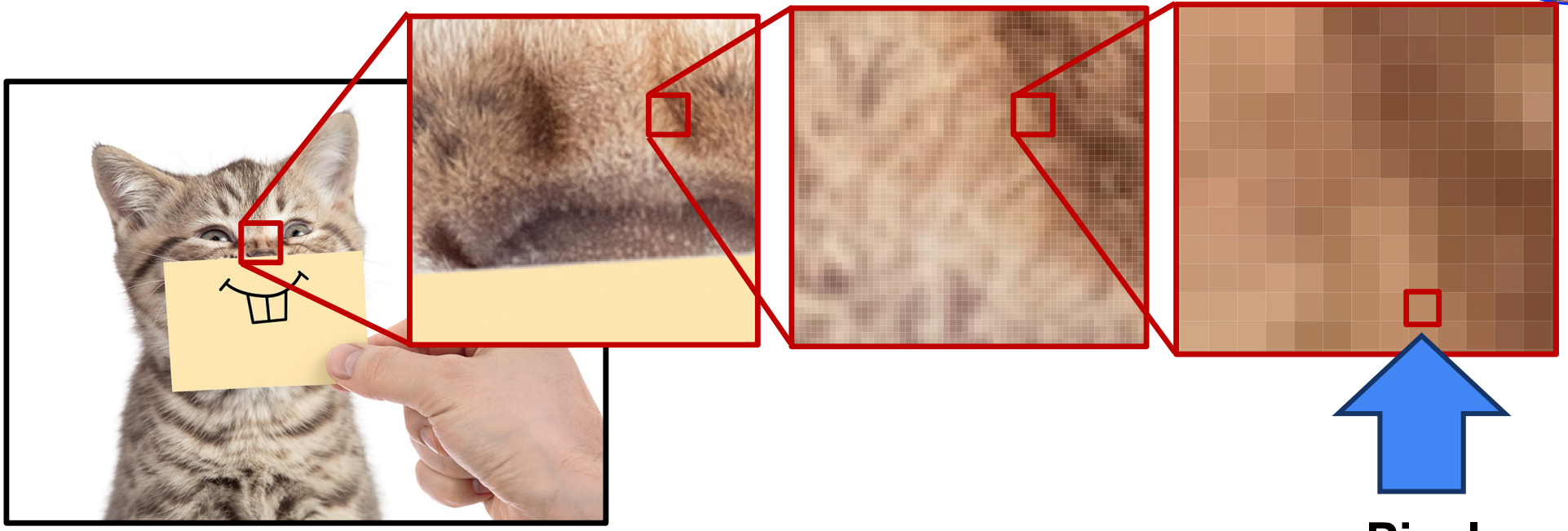
EPFL

Représentation des données : Images matricielles

- Représentée comme une grille de pixels
- Chaque pixel contient une couleur
- Avantage : idéale pour les photos
- Inconvénient : perd en qualité si on zoome (pixellisation)
- Exemples : PNG, JPEG, TIFF
- Logiciels : Photoshop, GIMP, Pillow (Python)



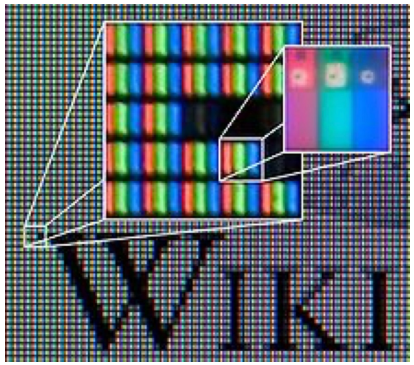
Représentation des données : Images matricielles

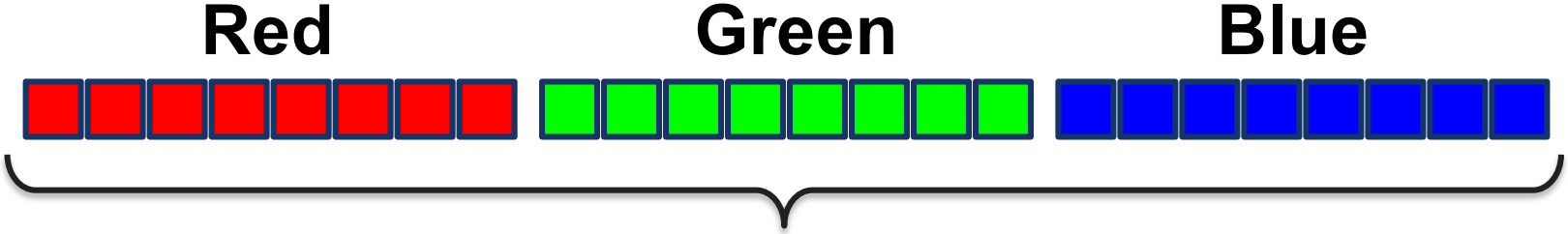


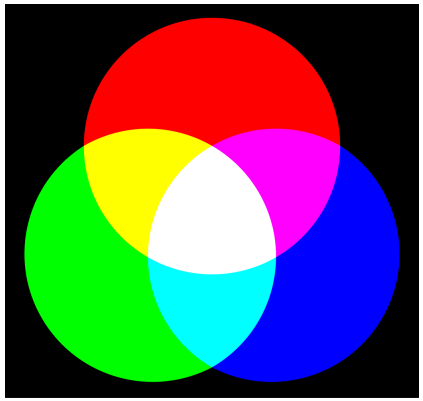
Pixel

Point de l'image qui a une seule couleur

RGB







Noir = (0, 0, 0)
Blanc = (255, 255, 255)
Bleu = (0, 0, 255)

En niveaux de gris, une seule composante

- 8 bits

1 pixel = 24 bits = 3 x 8 bits = 3 nombres entre 0-255
2^{24} couleurs possibles = 16'777'216

Bibliothèques

Bibliothèques : environnements virtuels

- Créer un environnement isolé pour ton projet Python.
- Chaque **venv** a ses propres packages installés avec **pip**, indépendamment du système ou des autres projets.
- Tu évites les effets de bord entre projets.

- Créer :

```
python3 -m venv venv
```

- Activer :

```
source venv/bin/activate
```

- Désactiver :

```
deactivate
```

Bibliothèques

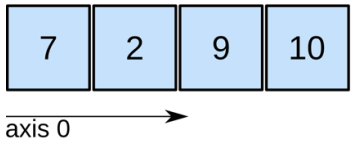
NumPy

- Une librairie Python pour les tableaux (arrays) et le calcul numérique.
- Ultra rapide : les opérations sont faites en C sous le capot.
- Permet de manipuler facilement des vecteurs, matrices, images, etc.
- Très utile pour : data science, machine learning, traitement d'images, simulations...

Bibliothèques

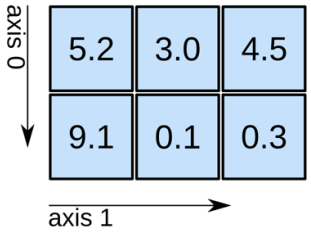
NumPy

1D array



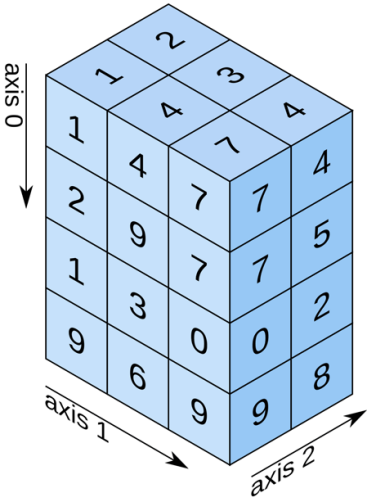
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)

Miniprojet

- Date limite : 23.05
- 2ème partie et logistique : après les vacances
- 10% de la note
- Possible de travailler par deux ou tout.e seul.e

- Partie 1
  1. Convertir une image couleur en niveaux de gris
  2. Dessiner un rectangle autour d'une zone d'intérêt
  3. Comparer des morceaux d'images
  4. Trouver automatiquement où un motif se cache dans une image

- Partie 2

À venir

NumPy : slice

Slicing 2D : `img[a:b, c:d]`

- **Objectif** : extraire ou modifier une sous-région.

```
coin = img[0:10, 0:10]  # lire le coin en haut à gauche
img[50:60, 50:60] = 0   # noircir une zone
```

NumPy : soustractions d'array

- Objectif : comparer deux images ou matrices.

```
diff = img1 - img2
```

- Même forme requise : `img1.shape == img2.shape`
- Peut donner des valeurs négatives

NumPy : `np.abs` et `np.average`

- **`np.abs`** : valeur absolue des éléments

```
diff = np.abs(img1 - img2)
np.abs(np.array([-3, 0, 2])) # donne [3, 0, 2]
```

```
zone1 = img1[20:30, 20:30]
zone2 = img2[20:30, 20:30]
diff = np.abs(zone1 - zone2)
moyenne = np.average(diff)
```

- **`np.average`** : moyenne sur tout ou partie d'un array

```
avg = np.average(img)
avg = np.average(img[0:10, 0:10])
```

Création d'images en niveaux de gris

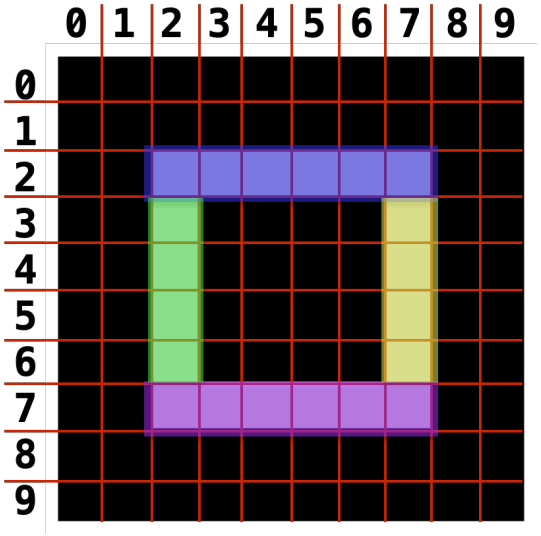

```
from miniprojectutils import *

img = new_image_gray(10, 10)
print(img)
save_image(img, "black.png")

for row in range(10):
  for col in range(10):
    img[row, col] = (row + 1) * 20

print(img)
save_image(img, "gradient.png")
```

Dessiner des formes


```
img = new_image_grey(10, 10)

for col in range(2, 8):
    img[2, col] = 255

for row in range(3, 7):
    img[row, 2] = 255
    img[row, 7] = 255

for col in range(2, 8):
    img[7, col] = 255

print(img)
save_image(img, "box.png")
```

```
img[2, 2:8] = 255
img[3:7, 2] = 255
img[3:7, 7] = 255
img[7, 2:8] = 255
```

```
img[[2,7], 2:8] = 255
img[3:7, [2,7]] = 255
```



Création d'images couleur


```
img = new_image_rgb(250, 250)

print(img[125, 125])

for row in range(250):
  for col in range(250):
    img[row, col] = [row, 0, col]

save_image(img, "output.png")
```

Résumé Cours 9 – ICC-P

- Représentation d'images (vectorielles et matricielles)
- Les images en Python sont (pour notre cas) représentés par structures bidimensionnelles (niveaux de gris) ou tridimensionnelles de NumPy.
- Quelques opérations en NumPy : slicing, soustractions, `np.abs` et `nb.average`.
- On peut lire et écrire des pixels individuels, qui sont soit un nombre unique entre 0 et 255 (niveaux de gris), soit trois nombres (RGB).

rafael.pires@epfl.ch

Merci