

# TP 11

## Traitement de l'image

Vous compléterez le fichier réponse **TP11-Fichier réponse.py**

⌚ **Explication** ⌚ Ce TP 11 - Traitement de l'image est une application du Cours 10 - Tableaux et images. Il faut donc s'y référer en cas de besoin.

### I - Création d'images

1

**Exercice**



**Dessins avec Python**

- 1) Dessiner et afficher les images suivantes de taille  $180 \times 360$  :

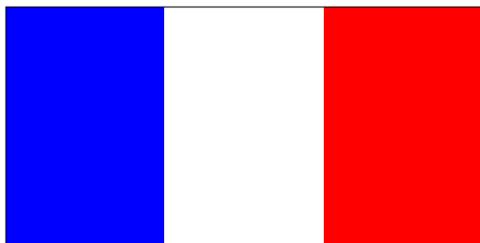


FIGURE 11.1 – France



FIGURE 11.2 – Allemagne

- 2) Dessiner et afficher les images suivantes de taille  $256 \times 256$  :

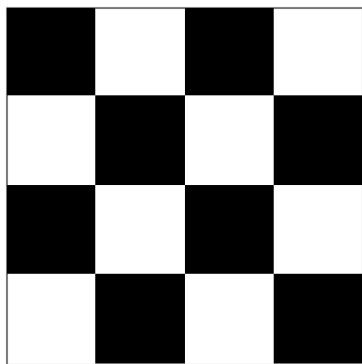


FIGURE 11.3 – Damier

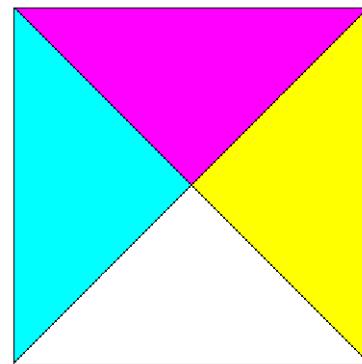


FIGURE 11.4 – Triangles

## II - Traitement de base de l'image

2

**Exercice**

### Traitement colorimétrique de l'image

Les résultats des tests sont affichés à la page suivante

- 1) a) Écrire une fonction

```
negatif(img),
```

qui prend en paramètre une image `img`, et **renvoie** l'image négative obtenue en remplaçant les composantes de chaque pixel de l'image d'entrée par leurs complémentaires à 255.

- b) Tester la fonction en **affichant** le résultat obtenu avec l'image carapuce.

- 2) a) Écrire une fonction

```
noir_blanç(img),
```

qui prend en paramètre une image `img`, et **renvoie** l'image en noir et blanc obtenue en remplaçant chaque pixel  $(r, g, b)$  de l'image d'entrée par le pixel  $(m, m, m)$  où :

$$m = \lfloor 0.2126r + 0.7152g + 0.0722b \rfloor,$$

la notation  $\lfloor \quad \rfloor$  désignant la partie entière.

- b) Tester la fonction en **affichant** le résultat obtenu avec l'image carapuce.

- 3) a) Écrire une fonction

```
rgb(img),
```

qui prend en paramètre une image `img`, et **renvoie** le tuple formé des images rouge, verte et bleue de l'image d'entrée. L'image rouge est obtenue en remplaçant chaque pixel  $(r, g, b)$  de l'image d'entrée par le pixel  $(r, 0, 0)$ .

- b) Tester la fonction en **affichant** les résultats obtenus avec l'image carapuce.

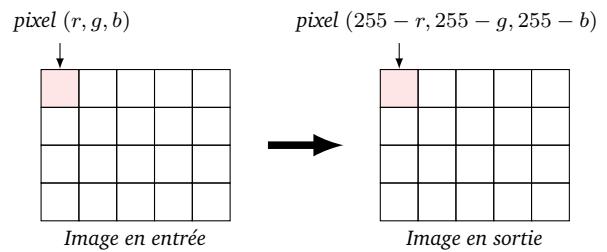


FIGURE 11.5 – Négatif d'une image

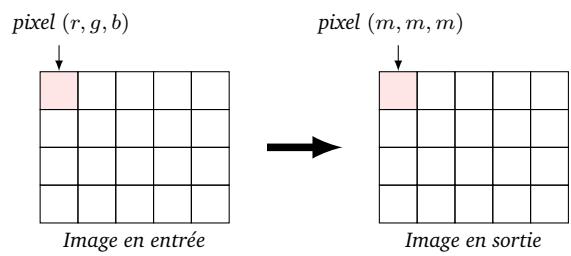


FIGURE 11.6 – Noir et blanc d'une image

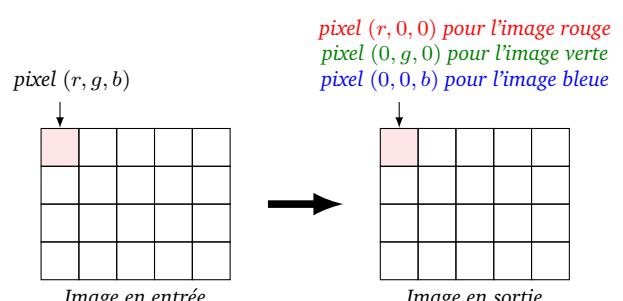


FIGURE 11.7 – Images rouge, verte et bleue d'une image

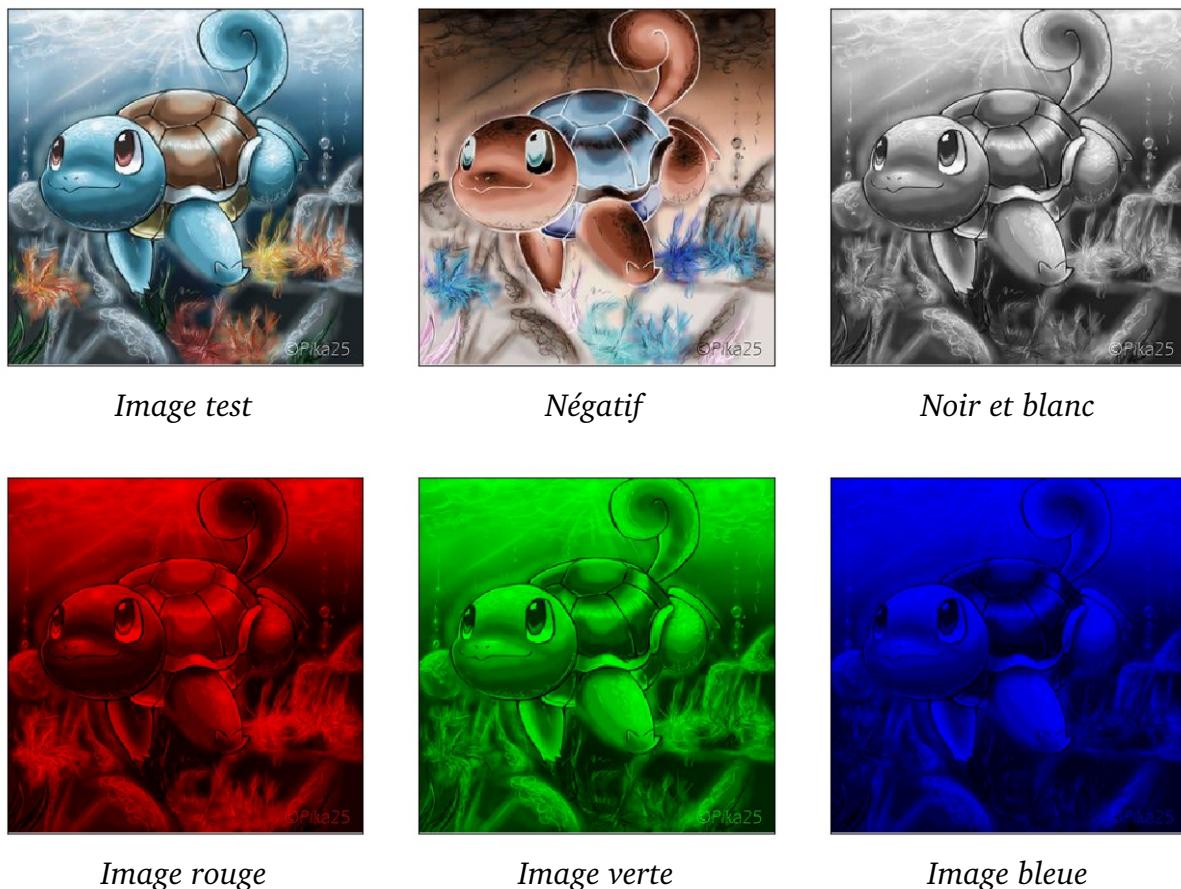


FIGURE 11.8 – Résultats des tests avec l'image carapuce

**3****Exercice****Traitements positionnels de l'image**

- a)** Écrire une fonction

```
sym_vert(img),
```

qui prend en paramètre une image `img`, et **renvoie** l'image obtenue par symétrie d'axe vertical passant par le milieu de l'image d'entrée.

- b)** Tester la fonction en **affichant** les résultats obtenus avec l'image carapuce.

- a)** Écrire une fonction

```
sym_hori(img),
```

qui prend en paramètre une image `img`, et **renvoie** l'image obtenue par symétrie d'axe horizontal passant par le milieu de l'image d'entrée.

- b)** Tester la fonction en **affichant** les résultats obtenus avec l'image carapuce.

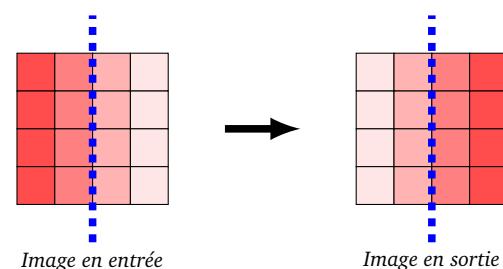


FIGURE 11.9 – Symétrie verticale

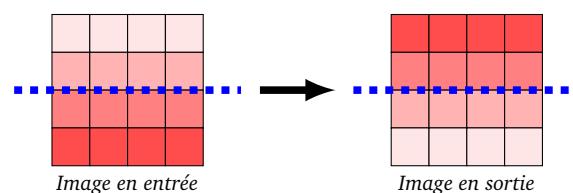


FIGURE 11.10 – Symétrie horizontale

## 3) a) Écrire une fonction

```
rotation(img),
```

qui prend en paramètre une image `img`, et **renvoie** l'image obtenue par rotation horaire d'angle  $\frac{\pi}{2}$  de l'image d'entrée.

- b) Tester la fonction en **affichant** les résultats obtenus avec l'image carapuce.

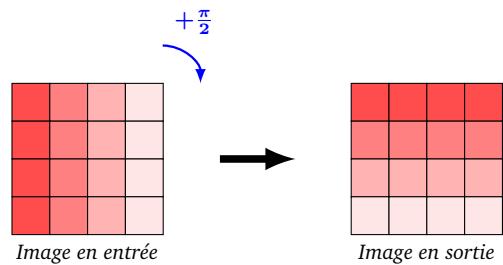


FIGURE 11.11 – Rotation horaire de  $\frac{\pi}{2}$

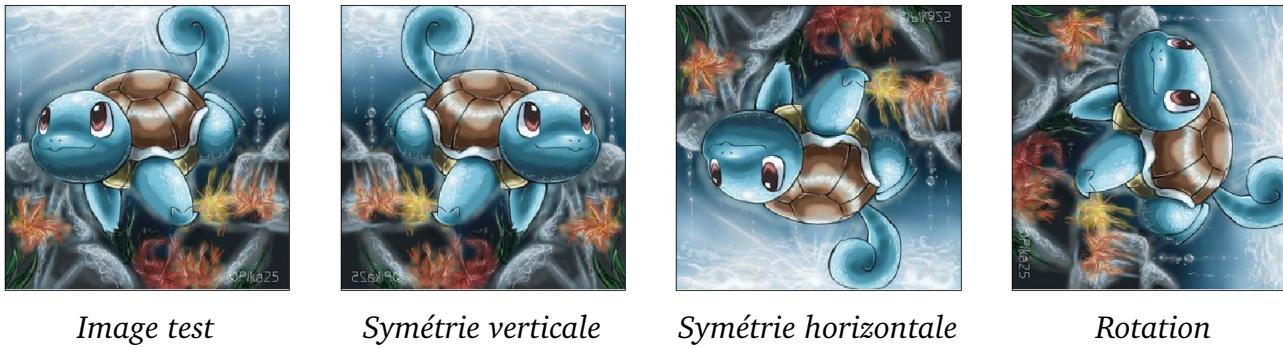


FIGURE 11.12 – Résultats des tests avec l'image carapuce

### III - Photoshoper avec Python

**4**

**Exercice**



#### Batman vs Superman

L'objectif est de créer une affiche pour le film *Batman vs Superman* sorti dans nos salles en 2016. Plus précisément, l'objectif est d'effectuer l'assemblage d'images suivant :

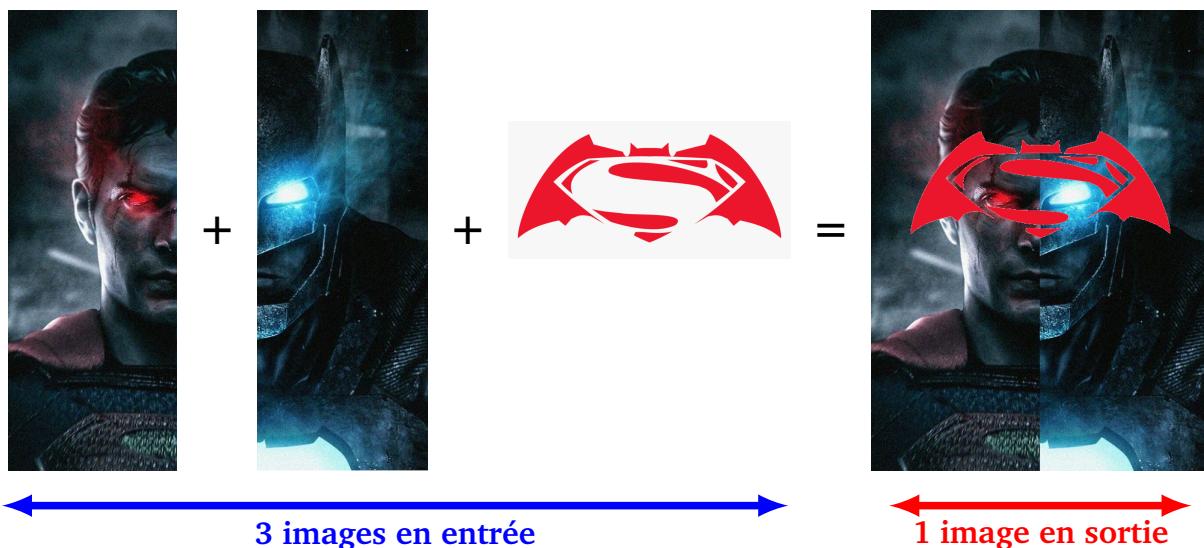


FIGURE 11.13 – Batman vs Superman

1) Écrire une fonction

```
fusion(img1, img2),
```

qui prend en paramètres deux images `img1` et `img2` de même hauteur, et **renvoie** l'image obtenue en juxtaposant les deux images d'entrée comme suit :

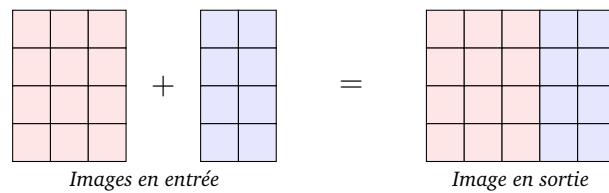


FIGURE 11.14 – Fusion de deux images

## 2) Écrire une fonction

```
incrustation(img1, img2, i0, j0),
```

qui prend en paramètres deux images `img1`, `img2` et deux entiers `i0`, `j0`, et renvoie l'image obtenue en incrustant l'image `img1` sur l'image `img2` en position  $(i_0, j_0)$  par le procédé de construction suivant :

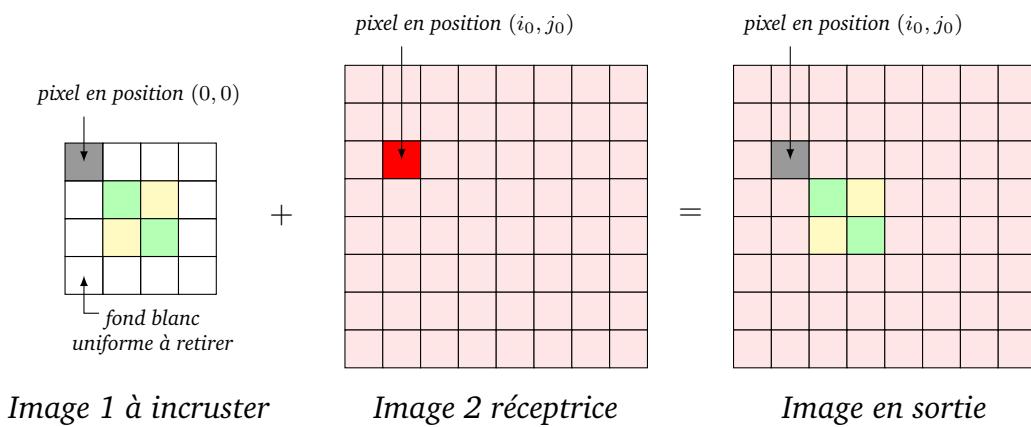


FIGURE 11.15 – Incrustation d'une image sur une autre

## 3) En déduire alors la création de l'affiche voulue.

## 4) (HS - Léo B. aka le maître Pokemon)

Léo B. ne sort jamais sans ses fidèles compagnons Pikachu et Bulbizarre. Pour éviter qu'il ne se sente seul, réaliser le photomontage suivant en veillant à ce qu'il soit le plus réaliste possible :

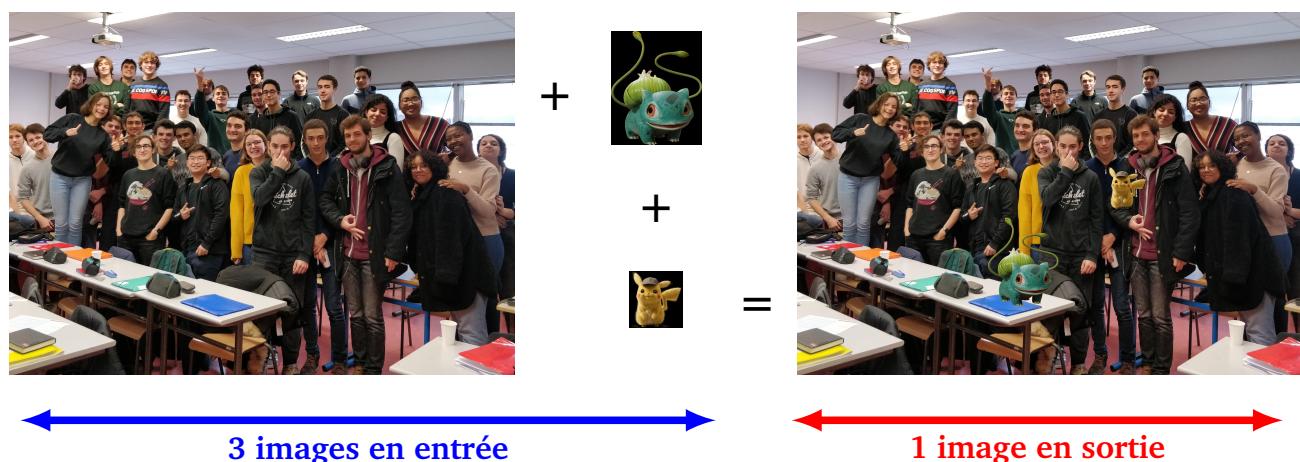


FIGURE 11.16 – Léo B. aka le maître Pokemon

## IV - Transformation du cliché Photomaton

5

Exercice

### Transformation du cliché Photomaton

En 1997, dans la revue *Pour la Science*, Jean-Paul Delahaye et Philippe Mathieu introduisent la TRANSFORMATION DU CLICHÉ PHOTOMATON qui permet d'obtenir quatre miniatures d'une image d'entrée (de largeur et hauteur paires) par le procédé de construction suivant :

- on découpe l'image d'entrée en paquets carrés de quatre pixels,
- pour chaque paquet carré de quatre pixels, on utilise celui en haut à gauche pour l'image réduite en haut à gauche, celui en haut à droite pour l'image réduite en haut à droite, etc...

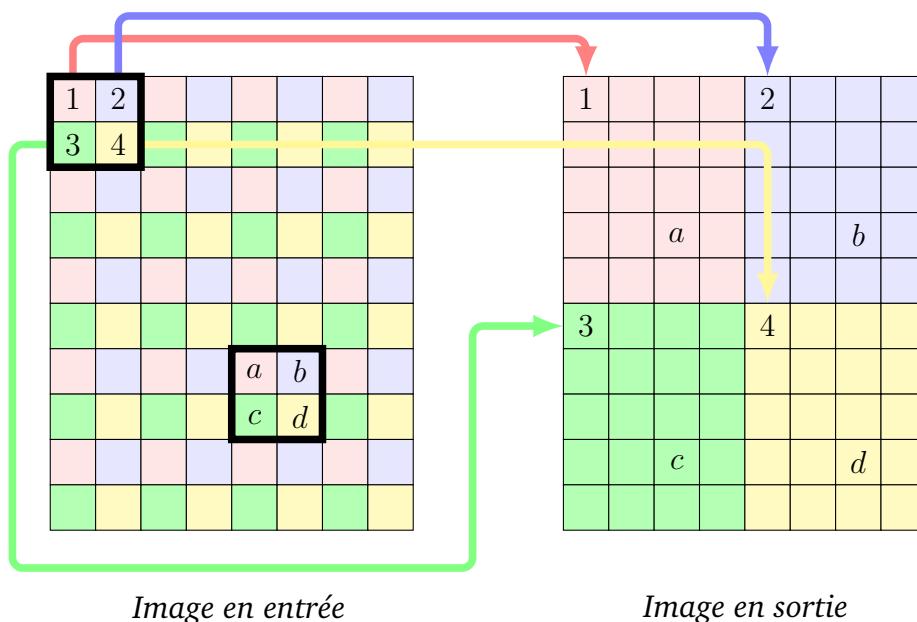


FIGURE 11.17 – Principe de la transformation du cliché Photomaton

- 1) a) À partir des coordonnées  $(i, j)$  d'un pixel de l'image d'entrée, exprimer ses coordonnées  $(i', j')$  dans l'image de sortie.  
*On pourra distinguer quatre cas suivant les parités de  $i$  et  $j$ .*
- b) À l'aide de la commande  $\%$ , exprimer  $i'$  (resp.  $j'$ ) en fonction de  $i$  et  $j$  dans une seule formule.
- c) En déduire une fonction

```
photomaton(img),
```

qui prend en paramètre une image `img`, et **renvoie** l'image obtenue par transformation du cliché Photomaton de l'image d'entrée.

- d) Tester la fonction en **affichant** le résultat obtenu avec l'image carapuce.



FIGURE 11.18 – Transformation du cliché Photomaton de carapuce

- 2) En appliquant de manière itérée la fonction photomaton(img) à n'importe quelle image d'entrée, il est toujours possible de retomber sur celle-ci !  
 Vérifier cette affirmation en écrivant un script permettant de visualiser les applications itérées de la fonction photomaton(img) sur l'image carapuce jusqu'à retomber-dessus.  
 Combien d'itérations a-t-on dû effectuer ?
- 3) a) Il est possible de démontrer que pour une image de taille  $p \times q$  (où  $p$  et  $q$  sont des entiers pairs), le nombre minimal  $n \in \mathbb{N}^*$  d'itérations à effectuer pour retomber sur cette image est donné par la formule :

$$n = \min \left\{ j \in \mathbb{N}^* \mid p - 1 \text{ et } q - 1 \text{ divisent } 2^j - 1 \right\} \quad (\star)$$

À partir de la formule ( $\star$ ), écrire une fonction

`periode(p,q),`

qui prend en paramètres deux entiers (pairs)  $p$  et  $q$ , et **renvoie** le nombre minimal d'itérations à effectuer pour retomber sur une image d'entrée de taille  $p \times q$ .

- b) Tester la fonction créée avec l'image carapuce.