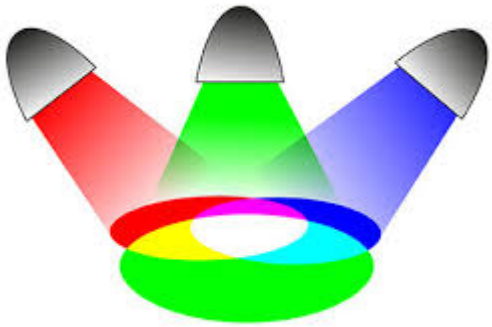
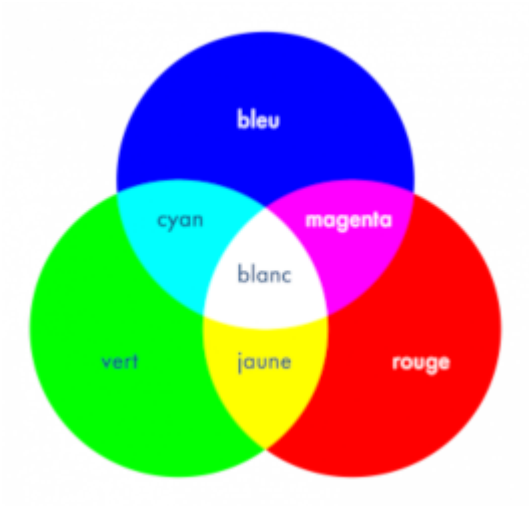
Couleurs, pixels et image numérique

Les couleurs

Selon le principe du mélange de couleurs additif, les couleurs rouge, vert et bleu sont utilisées pour obtenir les autres couleurs. On les appelle les trois couleurs primaires additives. La figure ci-dessous contient trois disques un pour chaque couleur présentant la même luminosité mise au maximum.
En l'absence de couleur, le résultat est noir, ce qui revient à une luminosité nulle pour chaque composante rouge, verte, et bleue. La combinaison de deux couleurs produit une nouvelle couleur.





Exercice 1

En observant la figure ci-dessus,

1. Indiquer quelles sont les deux couleurs primaires qui donnent :
   - le jaune
   - le magenta
   - le cyan
2. Indiquer la couleur que l'on obtient en ajoutant les trois couleurs primaires.

Pour tester le réglage des intensités des trois couleurs primaires, nous allons utiliser un module de Python qui permet de dessiner des grilles de carrés dont on pourra choisir la couleur en précisant l'intensité des trois couleurs primaires Red, Green et Blue.

Exercice 2

1. Exécuter la cellule ci-dessous pour pouvoir utiliser l'objet `BlockGrid`

Entrée[ ]:
```
from ipythonblocks import BlockGrid
```

2. Exécuter la cellule ci-dessous

Entrée[ ]:
```
R = 0
G = 0
B = 0
BlockGrid(8, 8, block_size=30, fill=(R, G, B))
```

3. Que se passe-t-il ?

Votre réponse : *double cliquer dans la cellule pour écrire votre réponse*

4. Dans le code ci-dessous, changer les valeurs contenues dans les variables `R`, `G` et `B` pour spécifier qu'on veut une intensité maximale (255) pour le rouge, et une intensité minimale (0) pour le vert et le bleu.
   Puis exécuter de nouveau le code.

Entrée[ ]:
```
R = 0
G = 0
B = 0
BlockGrid(8, 8, block_size=30, fill=(R, G, B))
```

7. En s'aidant de la figure sur l'addition des couleurs, modifier à nouveau les valeurs des variables `R`, `G` et `B` pour obtenir les couleurs jaune, magenta puis cyan.

Entrée[ ]:
```
# pour le jaune
R = 0
G = 0
B = 0
BlockGrid(8, 8, block_size=30, fill=(R, G, B))
```

Entrée[ ]:
```
# pour le mangenta
R = 0
G = 0
B = 0
BlockGrid(8, 8, block_size=30, fill=(R, G, B))
```

Entrée[ ]:
```
# pour le cyan
R = 0
G = 0
B = 0
BlockGrid(8, 8, block_size=30, fill=(R, G, B))
```

Exercice 3

1. Sachant que l'intensité de chaque couleur est donnée par un nombre compris entre 0 et 255, quel est le nombre de couleurs qu'on peut obtenir par le principe du mélange des couleurs ?

Votre réponse : *double cliquer dans la cellule pour écrire votre réponse*

2. Faire des essais en remplaçant `R` , `G` et `B` par des valeurs numériques.

Entrée[ ]:
```
BlockGrid(8, 8, block_size=30, fill=(R, G, B))
```

3. Choisir la même valeur pour `R` , `G` et `B` (sauf 0 et 255). Quelle couleur obtient-on ?

Entrée[ ]:
```
BlockGrid(8, 8, block_size=30, fill=(R, G, B))
```

Votre réponse : *double cliquer dans la cellule pour écrire votre réponse*

Les pixels

Un pixel est un petit élément d'une surface de visualisation dans une image numérique. Chaque pixel de votre écran est constitué de trois sous-pixels (rouge, vert et bleu). C'est de cette façon que fonctionnent tous les écrans d'ordinateur et de smartphone. Ainsi une image est composée de petits points appelés pixel. Un pixel est composé de trois parties : une partie rouge, une partie verte et une partie bleue. À chaque pixel, on associe donc 3 couleurs : le rouge, le vert et le bleu. On parle du canal rouge, du canal vert et du canal bleu d'un pixel (on parle de système RVB ou RGB en anglais). La valeur de l'intensité lumineuse associée à chaque canal de chaque pixel d'une image est comprise entre 0 et 255 (on a 256 valeurs possibles). On codera donc un pixel à l'aide d'un triplet de valeurs, la première valeur donnant l'intensité du canal rouge, la deuxième valeur donnant l'intensité du canal vert et la troisième valeur donnant l'intensité du canal bleu.

Exercice 4

1. Pour pouvoir agir sur les carrés de la grille, il faut lui donner un nom : nous choisirons la variable `grille` pour y stocker une grille noire de dimension 8×8.
   Pour afficher le contenu de la variable `grille` , il suffira d'écrire `grille` dans une cellule du notebook puis de l'exécuter.
   Exécuter les cellules ci-dessous

Entrée[ ]:
```
grille = BlockGrid(8, 8, block_size=30)
```

Entrée[ ]:
```
grille
```

2. On accède à un carré de la grille en précisant ses coordonnées dans la grille entre crochets.
   La première coordonnée est le numéro de ligne, la seconde le numéro de colonne.
   Attention : les numéros débutent à 0 et ici finissent à 7 puisque notre grille est de dimension 8×8. Pour modifier la couleur d'un carré, on utilise la commande :
   `grille[ligne, colonne] = (R, G, B)`

Compléter le code ci-dessous pour modifier la couleur du carré de coordonnées `(0, 0)` en rouge.

Entrée[ ]:
```
# version modifiée
grille[0, 0] = (..., ..., ...)
grille  # pour voir
```

4. Modifier les valeurs de `ligne`, `colonne`, `R`, `G` et `B` avec des valeurs possibles.

Entrée[ ]:
```
grille[ligne, colonne] = (R, G, B)
grille  # pour voir
```

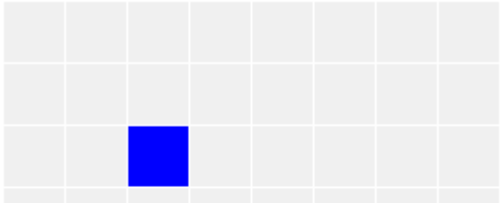
Exercice 5

1. Colorier le quadrillage représentant la grille qui s'affichera si on saisit le code donné ci-dessous.
   Vous pouvez écrire le nom exact de chaque couleur si vous ne possédez pas de crayon ayant la couleur précise.

```
grille = BlockGrid(8, 8, block_size=30, fill=(220, 220, 220))

grille[2, 2] = (0, 0, 255)
grille[5, 5] = (255, 0, 0)
grille[0, 0] = (255, 255, 0)
grille[7, 0] = ( 0, 255, 0)
grille[0, 7] = (0, 255, 255)
grille[7, 7] = (255, 0, 255)

grille
```



Entrée[ ]:

Les images

La grille contient 64 carrés : pour indiquer la couleur de chaque carré (ou pixel) il faut écrire 64 lignes de code !!!!
Voyons sur l'exemple ci-dessous comment on peut procéder en considérant l'image comme un tableau de 8 lignes et 8 colonnes.

Dans le code ci-dessous :

- la variable `N` contient le triplet de valeurs de R, G, B pour avoir du noir
- la variable `P` contient le triplet de valeur de R, G, B pour avoir du rose
- la variable `matrice` contient une liste de listes contenant les triplets de couleurs de chaque carré de gauche à droite et de bas en haut. La disposition proposée, sous forme d'une matrice, n'est pas obligatoire mais utile pour visualiser ce que l'on fait.

```
N = (0, 0, 0)
P = (255, 105, 180)
matrice = [
    [N, N, N, N, N, N, N, N],
    [N, P, P, N, P, P, N, N],
    [P, P, P, P, P, P, P, N],
    [P, P, P, P, P, P, P, N],
    [N, P, P, P, P, P, N, N],
    [N, N, P, P, P, N, N, N],
    [N, N, N, P, N, N, N, N],
    [N, N, N, N, N, N, N, N]
    ]
```

Exercice 6

1. Exécuter le code ci-dessous pour pouvoir utiliser le contenu de mat1.

Entrée[ ]:
```
N = (0, 0, 0)
P = (255, 105, 180)
matrice = [
    [N, N, N, N, N, N, N, N, N],
    [N, N, P, P, N, P, P, N, N],
    [N, P, P, P, P, P, P, P, N],
    [N, P, P, P, P, P, P, P, N],
    [N, N, P, P, P, P, P, N, N],
    [N, N, N, P, P, P, N, N, N],
    [N, N, N, N, P, N, N, N, N],
    [N, N, N, N, N, N, N, N, N]
    ]
```

2. Pour pouvoir associer à chaque carré la couleur correspondante, on a écrit une fonction nommée `dessiner` et qu'il n'est pas nécessaire de comprendre pour l'utiliser.
   Exécuter la cellule pour pouvoir utiliser cette fonction.

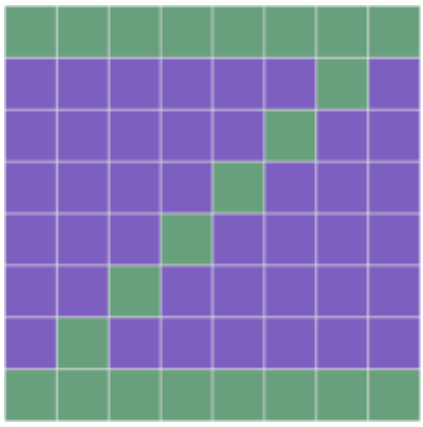
Entrée[ ]:
```
def dessiner(mat):
    nb_lignes = len(mat)
    nb_colonnes = len(mat[0])
    grille = BlockGrid(nb_colonnes, nb_lignes, block_size=30)
    for ligne in range(nb_lignes):
        for colonne in range(nb_colonnes):
            grille[ligne, colonne] = mat[ligne][colonne]
    return grille
```

3. Exécuter la cellule qui vous permettra de voir l'image construite à partir du contenu de la variable `matrice` .

Entrée[ ]:
```
dessiner(matrice)
```

4. Remplacer les `...` dans le code ci-dessous pour afficher la première lettre de votre prénom avec la couleur de votre choix et le fond de votre choix.
   Faites d'abord un dessin sur un quadrillage de 8 sur 8.

Par exemple, pour la lettre Z on a :



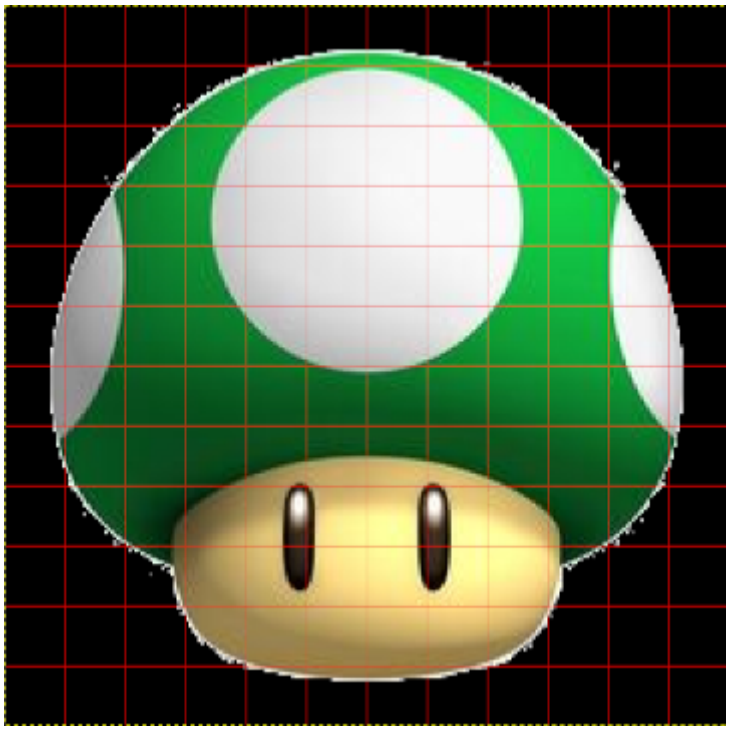
Entrée[ ]:
```
# la variable F prend pour  valeur le triplet pour avoir la couleur du fond
F = (..., ..., ...)
# la variable L prend pour valeur le triplet pour avoir la couleur de la let
L = (..., ..., ...)
matrice = [
    [..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ...]
    ]

dessiner(matrice)
```

Exercice 7

Pour représenter une image donnée, on dessine une grille par dessus et on utilise le principe suivant : la couleur d'un carré doit correspondre à la couleur dominante du carré sur l'image initiale. On dit qu'on pixelise l'image. Cette règle peut être un peu adaptée pour obtenir un meilleur rendu.

1. Pixeliser l'image de Toad, en utilisant une grille de 12 sur 12.



2. Compléter le code ci-dessous qui va permettre d'afficher l'image de Toad pixellisé.

Entrée[ ]:

```
# pour le blanc
b = (..., ..., ...) # remplacer les ...
# pour le noir
n = (..., ..., ...)
# pour le vert
v = (..., ..., ...)
# pour le brun
br = (110, 44, 0)
# pour le chair
c = (250, 219, 216)

matrice = [
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ..., ...]
]


dessiner(matrice)
```

Nathalie Maïer
Inspirée par une activité utilisant un sensehat de Stéphane Renouf.

Entrée[ ]: