1 Listes

Pour chacune des fonctions écrites ci-dessous, on pourra utiliser `Python Tutor` *pour une meilleure visualisation.*

Exercice 1.1 (Affichage)

1. (a) `6` (b) `200` (c) `18` (d) une erreur
2. La liste `L` devient `[200,12,14,18,6,19]`

Exercice 1.2 (Seuil) La fonction suivante convient :

```
def SeuilListe(L,val):
  '''
  L est une liste de valeurs de type numérique et val est de type numérique
  Renvoie une liste dans laquelle les éléments de L supérieurs ou égaux à val
  ont été remplacés par 0
  '''
  for i in range(len(L)):
    if L[i] >= val:
      L[i] = 0
  return L
```

Exercice 1.3 (Plus petit et plus grand élément) Les fonctions suivantes conviennent :

```
def PlusPetitElement(L):
  '''
  L est une liste de valeurs de
  type numérique
  Renvoie la plus petite valeur de L
  '''
  minimum = L[0]
  for i in range(1,len(L)):
    if L[i] < minimum:
      minimum = L[i]
  return minimum
```

```
def PlusGrandElement(L):
  '''
  L est une liste de valeurs de
  type numérique
  Renvoie la plus grande valeur de L
  '''
  maximum = L[0]
  for i in range(1,len(L)):
    if L[i] > maximum:
      maximum = L[i]
  return maximum
```

Exercice 1.4 (Echange) La fonction suivante convient :

```
def EchangeElement(L,i,j):
  '''
  L est une liste et i et j sont de type int
  Renvoie une liste dans laquelle les éléments d'indice i et j, s'ils existent,
  ont été permutés
  '''
  if(0 <= i < len(L) and 0 <= j < len(L)):
    L[i], L[j] = L[j], L[i]
  return L
```

Exercice 1.5 (Listes palindromes) La fonction suivante convient :

```
def EstPalindrone(L):
  '''
  L est une liste de valeurs de type numérique
  Renvoie True si L est une liste palindrome et False sinon
  '''
  i = 0
  j = len(L)-1
  while(i < j and L[i] == L|j]):
    i = i + 1
    j = j - 1
  return i >= j
```

Exercice 1.6 (Comptage) La fonction suivante convient :

```
def ComptageListe(L,val):
    '''
    L est une liste de valeurs de type numérique et val est de type numérique
    Renvoie le nombre de fois où val apparaît dans la liste L
    '''
    nb = 0
    for i in range(len(L)):
        if L[i] == val:
            nb = nb + 1
    return nb
```

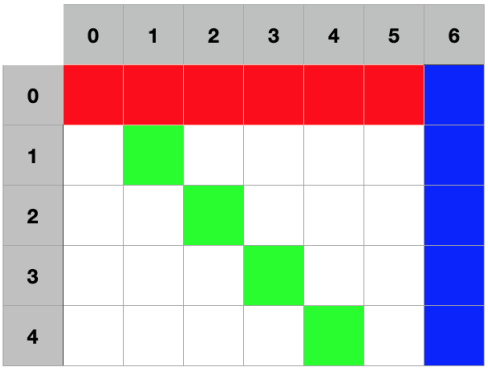
2 Listes de listes et images numériques

Exercice 2.1 (Préliminaire : petit point sur le codage de la couleur)

Couleur	Noir	Rouge	Vert	Bleu	Cyan	Jaune	Magenta	Blanc
Triplet	$(0,0,0)$	$(255,0,0)$	$(0,255,0)$	$(0,0,255)$	$(0,255,255)$	$(255,255,0)$	$(255,0,255)$	$(255,255,255)$

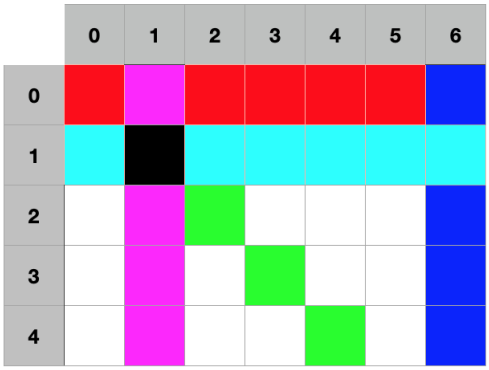
Exercice 2.2 (Création d'une image couleur pixel par pixel)

1. On obtient l'image ci-contre.
2. Les couleurs utilisées dans cette image sont le rouge, le bleu, le vert et le blanc.
3. Le dernier pixel de la première ligne est bleu car dans la fonction `DessineImage` on colorie d'abord la première ligne puis ensuite la dernière colonne. Le dernier pixel de la première ligne étant à la fois sur la première ligne et la dernière colonne, il a la dernière couleur affectée qui efface la précédente, soit ici le bleu.
4. Les pixels verts sont sur la diagonale : les numéros de ligne et de colonne sont égaux.



5. Il s'agit de produire l'image ci-contre.

   Pour cela on ajoute les instructions suivantes dans la fonction `DessineImage` à la suite des autres :




```
for x in range(7) : #La deuxième ligne en cyan
    img[1][x] = [0,255,255]
for y in range(5) : #La 3ème colonne en magenta
    img[y][2] = [255,0,255]
img[1][2] = [0,0,0] # Le pixel (2,1) est noir
```

Exercice 2.3 (Filtres couleurs)

1. Pour produire un filtre rouge on modifie la fonction précédente en lui donnant le nom de `FiltreRouge` et on change l'instruction interne aux boucles par l'instruction :

   ```
   new[y][x] = [img[y][x][0],0,0]
   ```

2. Pour produire un filtre vert on modifie la fonction précédente en lui donnant le nom de `FiltreVert` et on change l'instruction interne aux boucles par l'instruction :

   ```
   new[y][x] = [0,img[y][x][1],0]
   ```

3. Pour échanger les quantités de rouge et de bleu l'instruction interne aux boucles devient :

   ```
   new[y][x] = [img[y][x][2],img[y][x][1],img[y][x][0]]
   ```

Exercice 2.4 (Filtres couleurs secondaires)

```
def SeuilCouleur(img, seuilR, seuilV, seuilB) :
    ''' Produit une image couleur modifiée à partir de la matrice
        img d'une image couleur de trois valeurs entières comprises entre 0 et 255.
        seuilR est le seuil pour le rouge, seuilV pour le vert, seuilB pour le bleu.
        Ces seuils indiquent quand les trois intensités des pixels valent 255 ou 0
    '''

    hauteur = len(img) # nombre de lignes (hauteur de l'image)
    largeur = len(img[0]) # nombre de colonnes (largeur de l'image)
    new = [[0 for j in range(largeur)] for i in range(hauteur)]
    for y in range(hauteur) :
       for x in range(largeur) :
          if img[y][x][0] >= seuilR : # seuil sur le rouge
             r = 255
          else :
             r = 0
          if img[y][x][1] >= seuilR : # seuil sur le vert
             v = 255
          else :
             v = 0
          if img[y][x][2] >= seuilR : # seuil sur le bleu
             b = 255
          else :
             b = 0
          new[y][x] = [r,v,b]
    return new
```

Exercice 2.5 (Symétries)

1. (a) Le tableau complété est le suivant :

y\x	0	1	2	3
0	(0,0,255)	(0,255,0)	(255,0,0)	(255,255,255)
1	(0,255,255)	(255,255,0)	(255,0,255)	(0,0,0)

   (b) Le pixel noir est ligne 1 colonne 3 dans l'image symétrique. Le pixel vert est ligne 0 colonne 1 dans l'image symétrique.

   (c) Par cette symétrie verticale les pixels symétriques sont situés sur la même ligne mais pas sur la même colonne que ceux de l'image d'origine.

2. La fonction suivante convient :

```
def SymétrieHorizontale(img) :
   '''Produit une image par symétrie horizontale à partir d'une
        image donnée sous forme matricielle (liste de lignes).
        Les pixels d'une même colonne sont inversés par rapport
        à l'image d'origine.
        '''
    hauteur = len(img) # nombre de lignes (hauteur de l'image)
    largeur = len(img[0]) # nombre de colonnes (largeur de l'image)
    new = [[0 for x in range(largeur)] for y in range(hauteur)]
    for y in range(hauteur) :
       for x in range(largeur) :
          new[y][x] = img[hauteur-y-1][x]
    return new
```