Les nageurs

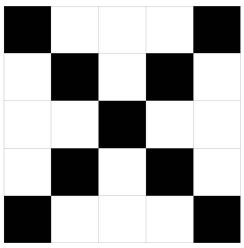
Dessin des nageurs

La première partie consiste à définir une fonction qui dessine le nageur dans sa position initiale.
Commencer par créer un script Python qui commence par les quatre lignes suivantes :

```
from kandinsky import *
from random import *
black = color(0,0,0)
white = color(255,255,255)
```

Introduction : Prise en main de la technique du dessin

L'objectif est d'afficher cette croix de taille 5x5 pixels sur l'écran à une position $(x; y)$. On va donc écrire une fonction `draw_cross(x,y)` qui dessinera cette croix dont le pixel en haut à gauche sera le pixel d'abscisse `x` et d'ordonnée `y`.



1. Etant donné que le dessin n'est fait que de deux couleurs (blanc et noir), nous allons attribuer la valeur 0 ou 1 à chaque pixel en prenant comme convention qu'un pixel noir sera codé par 1 et un pixel blanc par 0.
   Sur une feuille de papier, remplir un tableau à 5 lignes et 5 colonnes par la valeur de chaque pixel (0 ou 1) selon leur couleur.
2. Pour tracer le dessin, nous allons le découper en cinq lignes de cinq pixels. Chaque ligne du tableau de la question précédente va être représentée par une liste de 5 valeurs (0 ou 1) qui seront appelées `L1`, `L2`, `L3`, `L4` et `L5`.
   Dans l'éditeur Python, définir ces 5 listes.

```
L1=[1,0,0,0,1]
L2=[0,1,0,1,0]
L3=[0,0,1,0,0]
```

```
L4=[0,1,0,1,0]
L5=[1,0,0,0,1]
```

3. La fonction `set_pixel(i,j,black)` de Python permet d'afficher le pixel $(i;j)$ en noir. La fonction `set_pixel(i,j,white)` permet d'afficher le pixel $(i;j)$ en blanc.
   Ecrire une nouvelle fonction `draw1(x,y)` qui dessine la première ligne `L1` de 5 pixels à la position $(x;y)$ en utilisant la fonction `set_pixel` de Python. Cette fonction parcourra la liste `L1` et affichera le pixel concerné en noir si sa valeur est 1 ou en blanc si sa valeur est 0.

```
def draw1(x,y):
  for i in range (5):
    if L1[i]==1:
      set_pixel(x+i,y,black)
    else:
      set_pixel(x+i,y,white)
```

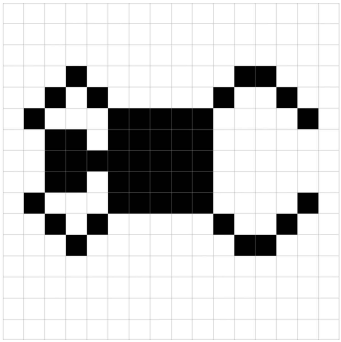
4. Définir une liste de listes, nommée L, qui contiendra les 5 listes `L1`, `L2`, `L3`, `L4` et `L5` : `L=[L1,L2,L3,L4,L5]`.
   Renommer la fonction `draw1` en `draw_cross` en la modifiant de sorte qu'elle dessine la croix en entier. Cela consistera à ajouter une deuxième boucle `for` qui parcourra la liste `L` de façon à tracer la croix en la dessinant ligne par ligne.

```
def draw_cross(x,y):
  for i in range (5):
    for j in range (5):
     if L[j][i]==1:
       set_pixel(x+i,y+j,black)
     else:
       set_pixel(x+i,y+j,white)
```

Dessin du nageur

Par analogie avec le dessin de la croix, vous allez représenter le dessin d'une position du nageur.

1. Définir la liste de listes `p1` par analogie avec la liste `L` de la question précédente. Elle contient 16 listes de 16 valeurs (0 ou 1).
2. Définir la fonction `draw_swimmer(x,y,p)` qui dessine le nageur défini par la liste `pos` en $(x;y)$. Vous la testerez avec la liste `p1` définie à la question précédente en écrivant par exemple `draw_swimmer(100,100,p1)`.



```
from kandinsky import*
from random import*
black = color(0,0,0)
white = color(255,255,255)

p1=[[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0],[0,0,1,0,1,0
    ,0,0,0,0,1,0,0,1,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,1,0],[0,0,1,1,0,1,1,1,1
    ,1,0,0,0,0,0,0],[0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[0,0,1,1,0,1,1,1,1,1,0,0
    ,0,0,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,1,0],[0,0,1,0,1,0,0,0,0,0,1,0,0,1,0
    ,0],[0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]

def draw_swimmer(x,y,p):
  for i in range (16):
    for j in range (16):
     if p[j][i]==1:
       set_pixel(x+i,y+j,black)
     else:
       set_pixel(x+i,y+j,white)
```

Autres positions du nageur

De façon à créer l'animation du nageur en train de nager, le nageur sera représenté par quatre positions successives différentes, la première étant celle que vous venez de dessiner.
Retrouvez dans le corrigé de l'exercice les listes `p1`, `p2`, `p3` et `p4` pour vous éviter de les écrire à la main !
Vous pourrez ensuite les tester en écrivant dans la console les instructions suivantes :

- `draw_swimmer(100,100,p2)`
- `draw_swimmer(100,100,p3)`
- `draw_swimmer(100,100,p4)`

```
p1=[[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0],[0,0,1,0,1,0
    ,0,0,0,0,1,0,0,1,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,1,0],[0,0,1,1,0,1,1,1,1
    ,1,0,0,0,0,0,0],[0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[0,0,1,1,0,1,1,1,1,1,0,0
    ,0,0,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,1,0],[0,0,1,0,1,0,0,0,0,0,1,0,0,1,0
    ,0],[0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]
p2=[[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],[0,0,0
    ,0,0,0,0,0,0,0,0,0,1,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0],[0,0,1,1,1,0
    ,0,0,0,0,1,0,0,0,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0],[1,0,1,1,0,1,1,1,1
    ,1,0,0,0,0,0,0],[0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[1,0,1,1,0,1,1,1,1,1,0,0
    ,0,0,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0],[0,0,1,1,1,0,0,0,0,0,1,0,0,0,0
    ,0],[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],[0
    ,0,0,0,0,0,0,0,0,0,0,1,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]
p3=[[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,1
    ,0,0,0,0,0,0,0,0,0,0,0,1,0],[0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0,1,0
    ,0,0,0,0,0,1,1,0,0,0],[0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0],[0,0,1,1,0,1,1,1,1
    ,1,0,0,0,0,0,0],[0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[0,0,1,1,0,1,1,1,1,1,0,0
    ,0,0,0,0],[0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0],[0,0,0,0,1,0,0,0,0,0,0,1,1,0,0
    ,0],[0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0],[0
```

```
    ,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]
p4=[[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0
    ,0,0,0,0,1,1,0,0,0,0,0,0,0],[0,0,0,0,0,1,1,0,0,0,0,0,0,0,0],[0,0,0,1,1,0
    ,0,0,0,0,0,0,0,0,0,0],[0,0,0,1,1,1,1,1,1,0,0,0,0,0,0],[1,1,0,1,1,1,1,1,1
    ,1,1,1,1,1,1,0],[1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[1,1,0,1,1,1,1,1,1,1,1
    ,1,1,1,0],[0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0],[0,0,0,1,1,0,0,0,0,0,0,0,0,0
    ,0],[0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0],[0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]
```

Dessin de la piscine

Maintenant, vous allez devoir définir une fonction qui dessinera la piscine.

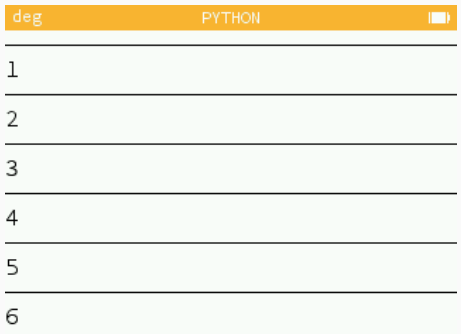
Numéros des nageurs

1. Dans la console Python, taper les instructions suivantes et observer le résultat de chaque instruction : `1+2`, `str(1)`, `str(1)+str(2)`.
   Que fait la fonction `str` ?
2. Dans la console Python, taper les instructions suivantes et observer le résultat de chaque instruction : `draw_string("Texte",0,0)`, `draw_string("Texte",100,0)`, `draw_string("Texte", 100, 100)`.
   Que fait la fonction `draw_string` ?
3. A partir de vos observations, créer la fonction `draw_pool()` qui dessine les numéros des nageurs à l'écran en respectant les règles suivantes :
   — Les numéros sont tous collés à gauche de l'écran, donc placés en $x = 0$;
   — Les numéros sont placés verticalement tous les 35 pixels ;
   — Le premier numéro est placé à une distance de 19 pixels du bord haut de l'écran (donc à $y = 19$).

```
from kandinsky import*
from random import*
black = color(0,0,0)
white = color(255,255,255)

def draw_pool():
  for i in range (6):
    draw_string(str(i+1),0,19+35*i)
```

```
    for j in range (320):
      set_pixel(j,35*i+10,black)
```



Bouées séparatrices

1. Quelle sont les instructions à écrire pour tracer une ligne horizontale noire continue qui traverse l'écran ? Vous utiliserez la fonction `set_pixel` (la largeur de l'écran est de 320 pixels).
2. Modifier la fonction `draw_string` pour qu'elle trace également une ligne représentant les bouées séparatrices entre chaque numéro. Le tracé respectera les règles suivantes :
   — Les lignes sont tracées tous les 35 pixels ;
   — La première ligne est placée à une distance de 10 pixels du haut de l'écran (donc à $y = 10$) ;
   — La longueur des lignes est de 320 pixels (soit la largeur de l'écran).

Animation de la course aléatoire des nageurs

Maintenant vous allez définir la fonction `swim()` qui simule la course de natation.

Trajectoire et arrêt de la course

1. Créer une fonction `swim()` que l'on complètera au fur et à mesure de cette partie. La fonction `swim()` commencera par tracer le dessin de la piscine effectué à la partie précédente. Vous appelerez donc la fonction `draw_pool` dès le début dans le contenu de la fonction `swim()`.
   Note : Les nageurs évolueront horizontalement de la droite vers la gauche. La trajectoire selon x partira de la position $x = 300$ (tout à droite) et ira jusqu'à la position $x = 10$ qui marquera la fin de la course.
2. La position des nageurs sera sockée dans une liste appelée `pos`. La liste `pos` contient donc 6 valeurs entières comprises entre 10 et 300, la première étant la position du nageur 1, etc.
   Dans la fonction `swim()` définir la liste `pos` avec les valeurs de la position initiale des 6 nageurs.

3. Dessiner les six nageurs en position de départ en utilisant la fonction `draw_swimmer` définie à la partie 1. Le dessin des nageurs respectera les règles suivantes :
   — Les nageurs seront dessinés verticalement tous les 35 pixels ;
   — Le premier nageur en partant du haut sera à une distance de 20 pixels du haut de l'écran (soit $y = 20$).
   Le nageur ne sera pour l'instant dessiné qu'en suivant la liste `p1`. Nous ajouterons les dessins des listes `p2`, `p3`, et `p4` plus tard.
4. Etant donné que la course s'arrête lorsque tous les nageurs sont arrivés en position $x = 10$ et continue tant qu'au moins une des positions est supérieure à 10, nous allons utiliser une boucle `while` dont la condition d'arrêt sera la même que celle de la course.
   Ecrire la condition d'arrêt de la boucle `while` en utilisant la fonction `max(liste)` qui renvoie le plus grand nombre contenu dans une liste.

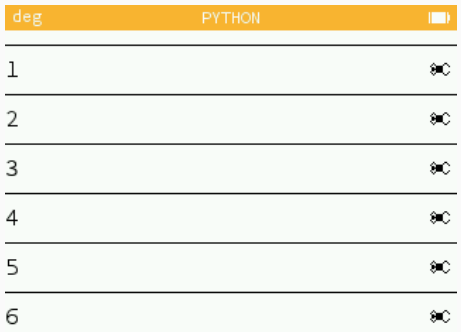
Déroulement de la course

1. A chaque itération de la boucle, un nageur tiré au hasard avance d'un pixel selon x. Utiliser la fonction `randint` pour simuler ce tirage au sort et la fonction `draw_swimmer` pour dessiner le nouveau nageur ayant avancé.
   Le nageur ne sera pour l'instant dessiné qu'en suivant la liste `p1`.
   Attention ! Le nageur ne doit être redessiné que s'il n'est pas déjà arrivé au bout de la course. N'oubliez pas que la position du nageur doit également être mise à jour dans la liste `pos`.
2. Créer une liste `nager` au début de la fonction en la définissant comme suit : `nager=[p1,p2,p3,p4]`.
   Modifier alors le 3e argument de la fonction `draw_swimmer` utilisé à la question précédente en remplaçant `p1` par `nager[pos[i]%4]`.
   Lancer alors `swim()` dans la console.
   A votre avis que fait `nager[pos[i]%4]` ?

Classement des nageurs

Il serait bien d'afficher au fur et à mesure de la course l'ordre d'arrivée des nageurs de sorte à réaliser un classement.

1. Créer une variable `rang` au début de la fonction `swim()`. Elle sera initialisée à 0 et incrémentée de 1 à chaque fois qu'un nageur arrivera au bout de la course.
2. A la fin de la fonction, écrire un test `if` qui ajoute 1 à la valeur de `rang` si le nageur qui vient de se déplacer est arrivé à la position d'arrivée.
3. Compléter ce test, en dessinant le rang d'arrivée sur la ligne du nageur qui vient d'arriver grâce à la fonction `draw_string(texte,x,y)`. Vous respecterez les règles suivantes :
   — Le numéro est dessiné horizontalement au pixel d'abscisse $x = 300$;
   — Les numéros sont dessinés verticalement tous les 35 pixels et le premier numéro en partant du haut est à une distance de 19 pixels du bord haut de l'écran (soit $y = 19$).




```
from kandinsky import*
from random import*
black = color(0,0,0)
white = color(255,255,255)

p1=[[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0],[0,0,1,0,1,0
    ,0,0,0,0,1,0,0,1,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,1,0],[0,0,1,1,0,1,1,1,1
    ,1,0,0,0,0,0,0],[0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[0,0,1,1,0,1,1,1,1,1,0,0
    ,0,0,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,1,0],[0,0,1,0,1,0,0,0,0,0,1,0,0,1,0
    ,0],[0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]
p2=[[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0],[0,0,0
    ,0,0,0,0,0,0,0,0,0,1,0,0,0],[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0],[0,0,1,1,1,0
    ,0,0,0,0,1,0,0,0,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0],[1,0,1,1,0,1,1,1,1
    ,1,0,0,0,0,0,0],[0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[1,0,1,1,0,1,1,1,1,1,0,0
    ,0,0,0,0],[0,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0],[0,0,1,1,1,0,0,0,0,0,1,0,0,0,0
    ,0],[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],[0
    ,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]
p3=[[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,1
    ,0,0,0,0,0,0,0,0,0,0,0,1,0],[0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0,1,0
    ,0,0,0,0,0,1,1,0,0,0],[0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0],[0,0,1,1,0,1,1,1,1
    ,1,0,0,0,0,0,0],[0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[0,0,1,1,0,1,1,1,1,1,0,0
    ,0,0,0,0],[0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0],[0,0,0,0,1,0,0,0,0,0,0,1,1,0,0
    ,0],[0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0],[0
    ,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]
p4=[[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0
    ,0,0,0,0,1,1,0,0,0,0,0,0,0],[0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0],[0,0,0,1,1,0
    ,0,0,0,0,0,0,0,0,0,0],[0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0],[1,1,0,1,1,1,1,1,1
```

```
    ,1,1,1,1,1,1,0],[1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0],[1,1,0,1,1,1,1,1,1,1,1,1
    ,1,1,1,0],[0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0],[0,0,0,1,1,0,0,0,0,0,0,0,0,0,0
    ,0],[0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0],[0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0]]

nage=[p1,p2,p3,p4]

def draw_swimmer(x,y,pos):
  for i in range (16):
    for j in range (16):
      if pos[j][i]==1:
        set_pixel(x+i,y+j,black)
      else:
        set_pixel(x+i,y+j,white)

def draw_pool():
  for i in range (6):
    draw_string(str(i+1),0,19+35*i)
    for j in range (320):
      set_pixel(j,35*i+10,black)

def swim():
  draw_pool()
  pos=[300,300,300,300,300,300]
  rang=0
  for j in range (6):
    draw_swimmer(300,j*35+20,p1)
  while max(pos)>10:
    i=randint(0,5)
    if pos[i]>10:
      draw_swimmer(pos[i]-1,i*35+20,nage[pos[i]%4])
      pos[i]=pos[i]-1
      if pos[i]==10:
      rang=rang+1
      draw_string(str(rang),300,19+35*i)
```