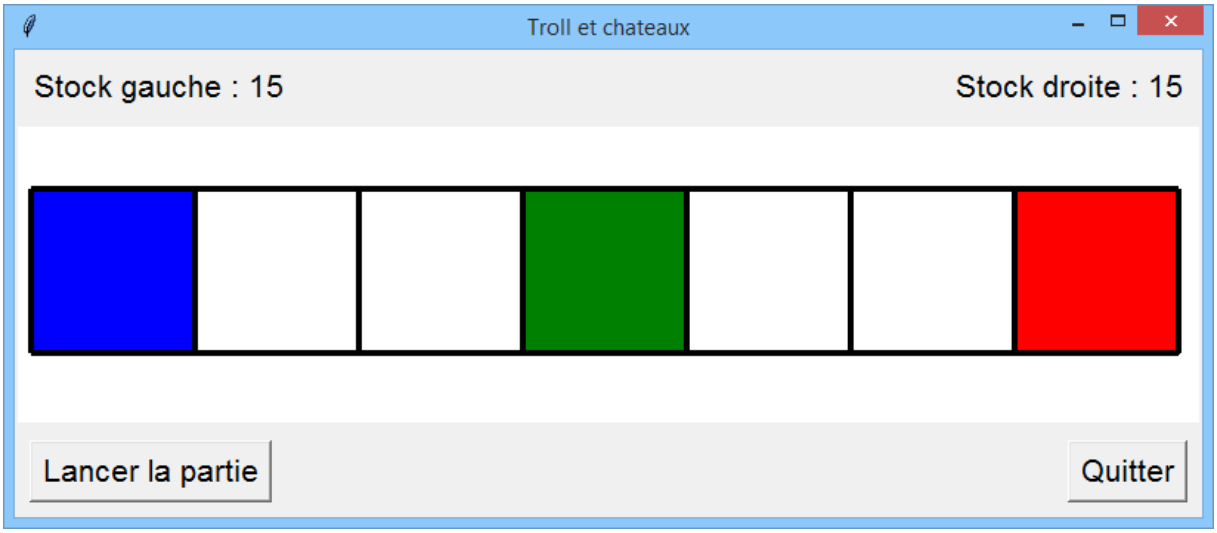
Premières interfaces graphiques

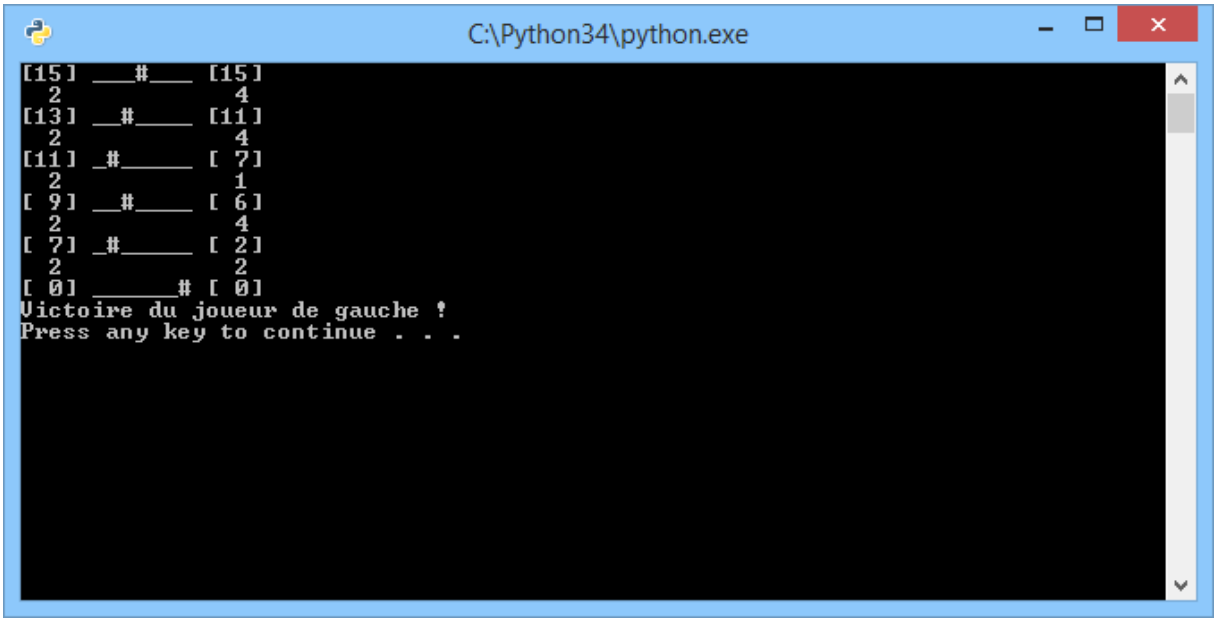
I. Le module `tkinter`

a) Choisir une case au hasard

L'objectif de ce TD est de vous faire réaliser vos premières interfaces graphiques en Python.

Comme nous avons pu le voir avec le projet "Trolls & châteaux", une interface graphique, même basique, permet d'obtenir un rendu plus agréable et plus interactif que la console :





En Python, il existe de nombreux outils pour réaliser de telles interfaces graphiques. Dans le cadre de ce cours, nous allons utiliser le module `tkinter`, qui a le gros avantage d'inclus par défaut dans les distributions de Python (il n'y a donc rien à installer de plus).

Pour utiliser ce module, il suffit d'ajouter `import tkinter` au début du fichier.

Remarque : La syntaxe n'est pas exactement la même entre Python 2.7 et Python 3.4. Par conséquent, si vous n'utilisez pas la bonne version de Python, vous risquez d'obtenir le message suivant : `ImportError: No module named tkinter`

Question 1. Créer un nouveau fichier `.py`, et importer le module `tkinter`.

b) Fenêtre principale et `canvas` :

Le module `tkinter` permet de créer des fenêtres graphiques assez variées, avec différents types de composants : textes, champs de saisie, boutons, etc. Pour ce TD, nous n'utiliserons qu'un seul composant : le `Canvas`, c'est-à-dire une zone dans laquelle il est possible de "dessiner".

Pour commencer, il faut créer l'objet qui va correspondre à la "fenêtre principale" : c'est dans cet objet que l'on ajoutera ensuite les différents composants de notre interface graphique. Pour créer cet objet, la syntaxe est la suivante :

```
fenetre = tkinter.Tk()
fenetre.mainloop()
```

Question 2. Utiliser le code ci-dessus pour créer la fenêtre principale.

Remarque : L'instruction `fenetre.mainloop()` a pour effet de donner la main à l'interface graphique : le programme ne s'arrêtera donc pas tant que la fenêtre principale n'aura pas été fermée.

La fenêtre principale est un peu vide : nous allons donc y ajouter un objet de type `Canvas`, qui nous permettra de "dessiner" des éléments à l'écran. Pour créer cet objet, notre code devient :

```
# On cree la fenetre principale
fenetre = tkinter.Tk()

# On cree un "canvas"
canvas = tkinter.Canvas(fenetre, width=800, height=400, bg = "white")
# On ajoute ce canvas dans la fenetre principale
canvas.pack()

# On donne la main a l'interface graphique
fenetre.mainloop()
```

Question 3. Utiliser le code ci-dessus pour afficher une fenêtre contenant un `Canvas`.

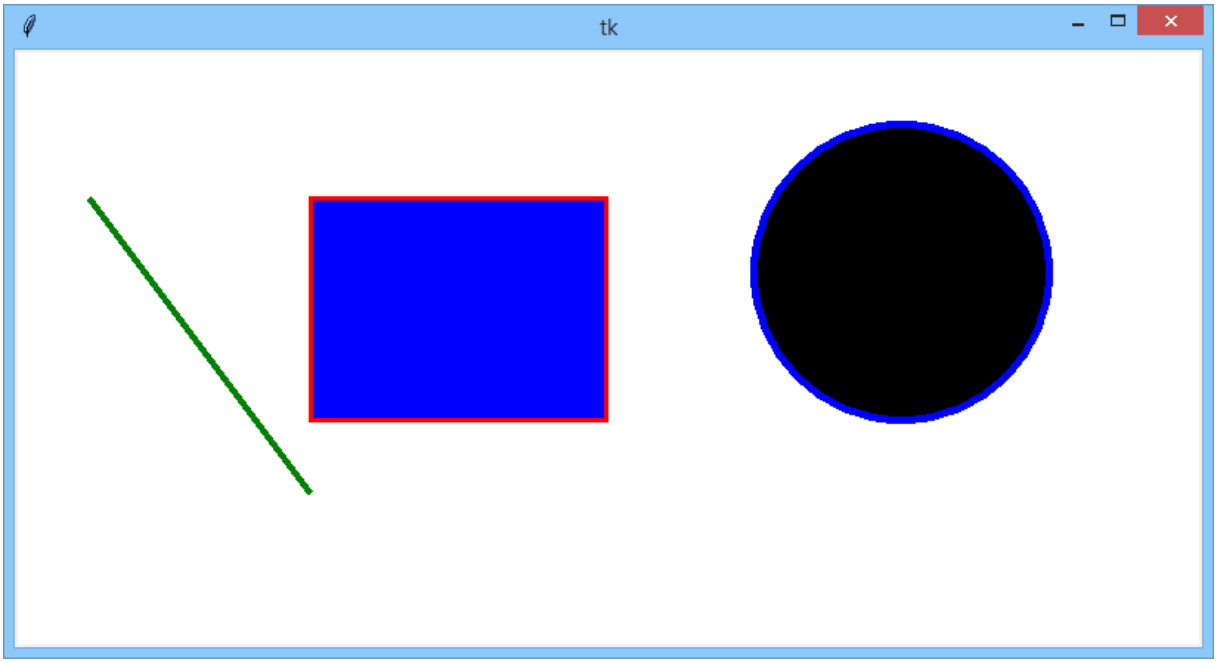
Remarque : Vous pouvez modifier les paramètres `width`, `height` et `bg` pour vérifier que vous avez bien compris leur signification.

II. Premiers tests d'affichage

a) Les méthodes de la classe **Canvas** :

La classe `Canvas` met à votre disposition plusieurs méthodes qui permettent de "dessiner" à l'écran. L'exemple ci-dessous utilise trois d'entre elles :

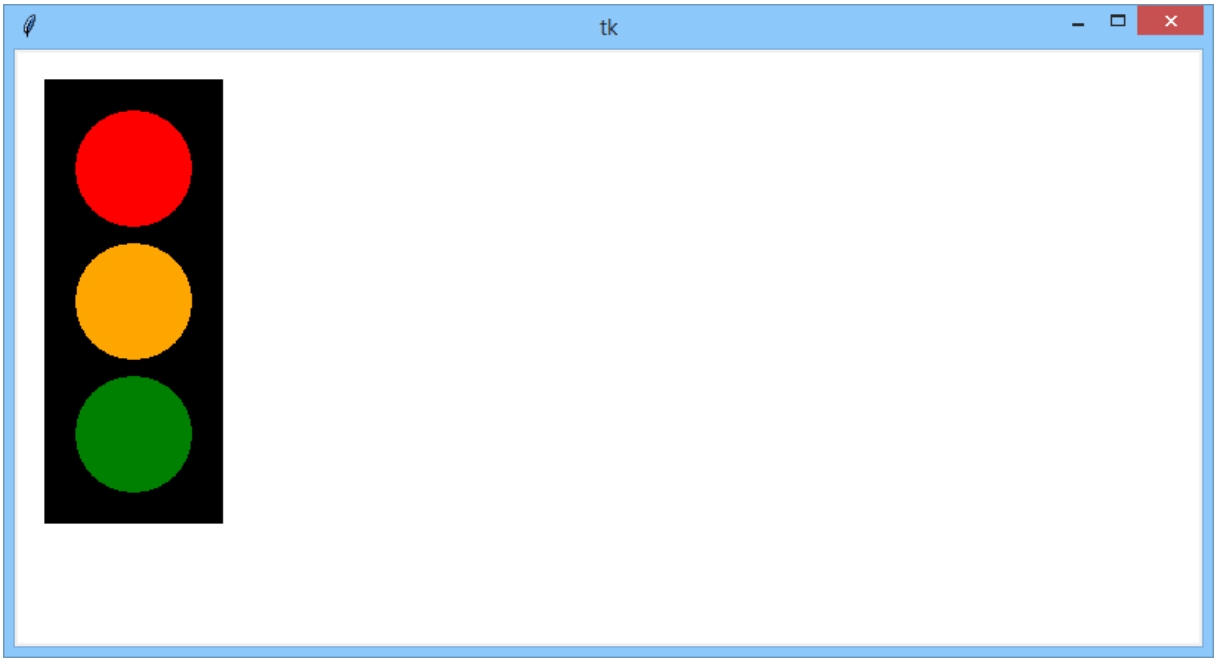
```
import tkinter

fenetre = tkinter.Tk() # On cree la fenetre principale

canvas = tkinter.Canvas(fenetre, width=800, height=400, bg =
"white")
canvas.pack()

# Tracer une ligne du point (50, 100) au point (200, 300), de
couleur verte et d'epaisseur 4
canvas.create_line(50, 100, 200, 300, fill="green", width=4)

# Tracer un rectangle compris entre les points (200, 100) et (400,
250), avec un fond bleu et une bordure rouge d'epaisseur 3
canvas.create_rectangle(200, 100, 400, 250, fill="blue",
outline="red", width=3)

# Tracer une ellipse comprise entre les points (500, 50) et (700,
250), avec un fond noir et une bordure bleue d'epaisseur 5
canvas.create_oval(500, 50, 700, 250, fill="black", outline="blue",
width="5")

fenetre.mainloop()
```



Question 4. Modifier les différents arguments de l'exemple précédent pour vous approprier ces fonctions.

b) Premiers dessins

Question 5. Reproduire le feu tricolore ci-dessous

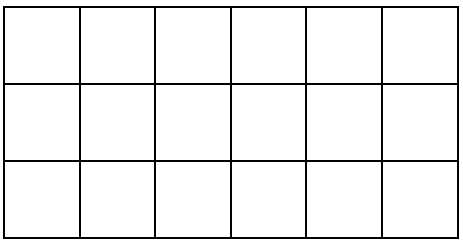


Question 6. Dessiner les disques suivants :

- un disque noir de rayon 100 et de centre (200, 250)
- un disque noir de rayon 60 et de centre (120, 150)
- un disque noir de rayon 60 et de centre (280, 150)

Indice : Vous pouvez commencer par créer une fonction qui prend en argument un objet `c` de type `Canvas`, des coordonnées `(x,y)` et un rayon `r`, et qui trace un disque noir du centre `(x,y)` et de rayon `r` sur le `Canvas c`.

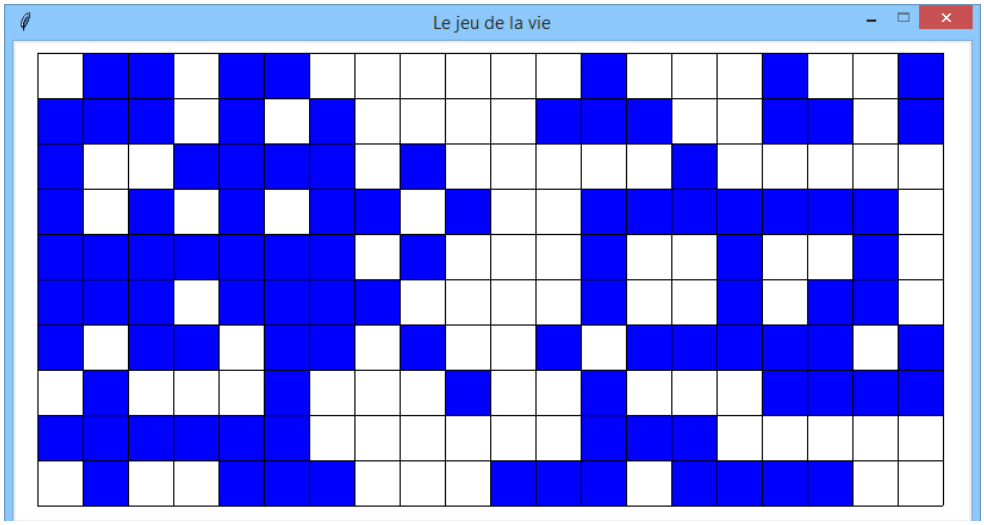
Question 7. Dessiner la grille ci-dessous :



III. Le jeu de la vie

a) Première grille

Question 8. Ecrire une fonction qui prend en argument une grille du jeu de la vie, et qui affiche cette grille dans la fenêtre graphique



Remarque : Pour cette question, vous pouvez fixer arbitrairement la taille des cases, ou passer cette taille en argument de votre fonction.

★ **Question 9.** Améliorer la fonction précédente pour calculer automatiquement la taille des cases pour occuper un maximum de place dans la fenêtre graphique.

b) Le jeu de la vie, version graphique

A ce stade, il ne reste plus qu'à mettre à jour la grille et rafraichir l'affichage à chaque fois. Pour cela, on peut utiliser la syntaxe suivante juste avant l'appel à l'instruction `fenetre.mainloop()`

```
def boucleAffichage():
   fonctionAffichage() # Votre fonction d'affichage
   fenetre.after(1000, boucleAffichage) # 1000 ms entre chaque appel

boucleAffichage()

fenetre.mainloop()
```

Ce code consiste à appeler la fonction `boucleAffichage` juste avant de donner la main à l'interface graphique. Cette fonction appelle la fonction d'affichage, puis se rappelle (grâce à la méthode `after`) au bout d'un certain temps (ici 1000 millisecondes).

★ **Question 10.** Utiliser cette syntaxe pour afficher les différentes étapes du jeu de la vie