Vers les L-systèmes

Le but de ce projet est tout d'abord de manipuler les classes et les objets en Python. Vous allez devoir définir des classes et les utiliser de façon à écrire un code le plus lisible et évolutif possible.
Le fichier de base sera le suivant :

```
import tkinter as tk

WIDTH = 600
HEIGHT = 400

window = tk.Tk()
window.title("Vers les L-systèmes - Partie ?")

canvas = tk.Canvas(window, width = WIDTH, height = HEIGHT, bg = 'ivory')
canvas.pack(padx = 10, pady = 10)

### Début de la partie à modifier ###

### Fin de la partie à modifier ###

window.mainloop()
```

Ouvrir le programme complet dans Spyder en cliquant sur le logo.

Partie 1 – De la géométrie avec `tkinter`

La module `tkinter` dispose d'un widget canevas, il s'agit d'une surface permettant, en autres, de dessiner des formes géométriques (rectangles, disques, du texte, ...).

- Les lignes :

```
canvas = tk.Canvas(window, width = WIDTH, height = HEIGHT, bg = 'ivory')
canvas.pack(padx = 10, pady = 10)
```

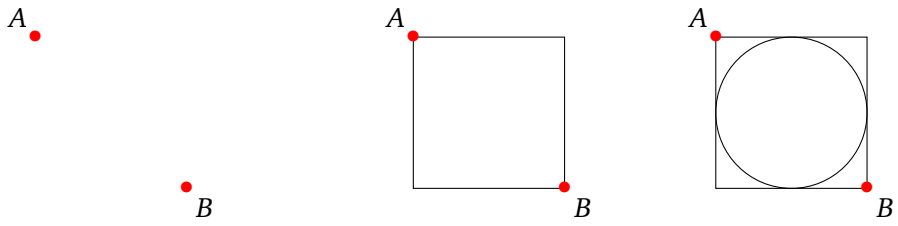
  permettent de créer un widget canevas dont les dimensions sont les valeurs des variables globales `WIDTH` et `HEIGHT`.
  Ce widget sera intégré dans la fenêtre principale `root` avec la méthode `pack`.

- Un segment est généré par la méthode `create_line` du `canvas` : l'instruction

```
A = (50, 50)
B = (150, 150)
canvas.create_line(A, B, width=3, fill="blue")
```

  permet de créer le segment coloré en bleu et d'épaisseur 3 pixels entre les points $A(50\,;\,50)$ et $B(150\,;\,150)$.

- Contrairement à ce que l'on pourrait s'attendre, un cercle n'est pas construit en donnant son centre et son rayon. En fait, le cercle est produit comme si on suivait les trois étapes ci-dessous :



On donne à `tkinter` deux points A et B qui sont deux sommets opposés du carré ayant leurs côtés parallèles aux axes et circonscrit au cercle que l'on veut dessiner.
Pour créer le cercle de centre $O\,(100\ ;\ 100)$ et de rayon 25, il faudra exécuter les instructions :

```
r = 25
A = (100-r, 100-r)
B = (100+r, 100+r)
canvas.create_oval(A, B, fill='red', outline='red')
```

ou encore

```
r = 25
canvas.create_oval(100-r, 100-r, 100+r, 100+r, fill='red', outline='red')
```

Pour représenter un **point**, il faudra choisir, par exemple, un rayon égale à 5 pixels.

Toutes les informations : https://docs.python.org/fr/3/library/tkinter.html

Exercice
Avec les instructions précédentes, éventuellement en écrivant des fonctions, réaliser dans une fenêtre 400x400, le dessin ci-dessous.
Sauvegarder votre fichier avec le nom `NOM_partie1.py`

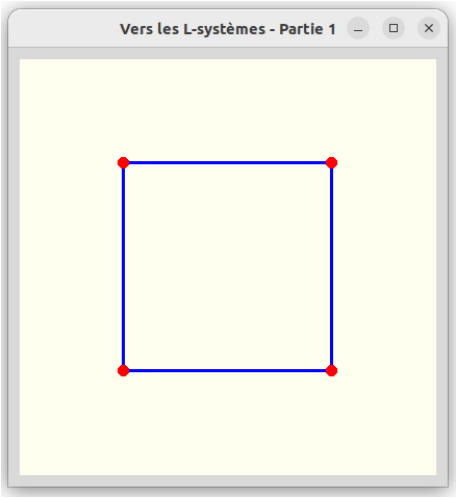


FIGURE 1 – Figure à réaliser

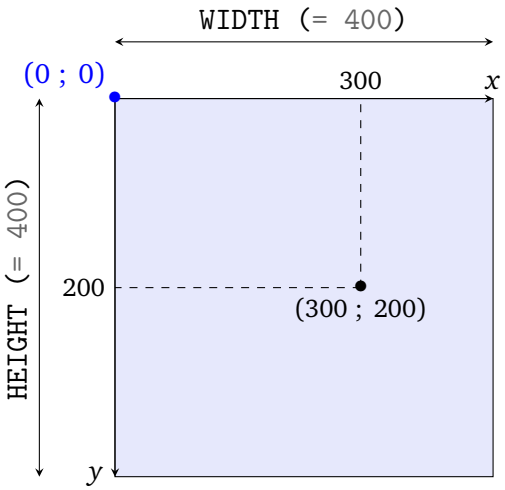


FIGURE 2 – Orientation des axes dans `tkinter`

Comme le montre la figure ci-dessus, dans une fenêtre `tkinter`, l'origine du repère se situe en haut à gauche, l'axe des abscisses est horizontal et orienté vers la droite et l'axe des ordonnées est vertical orienté vers le bas.

Partie 2 – La classe `Point`

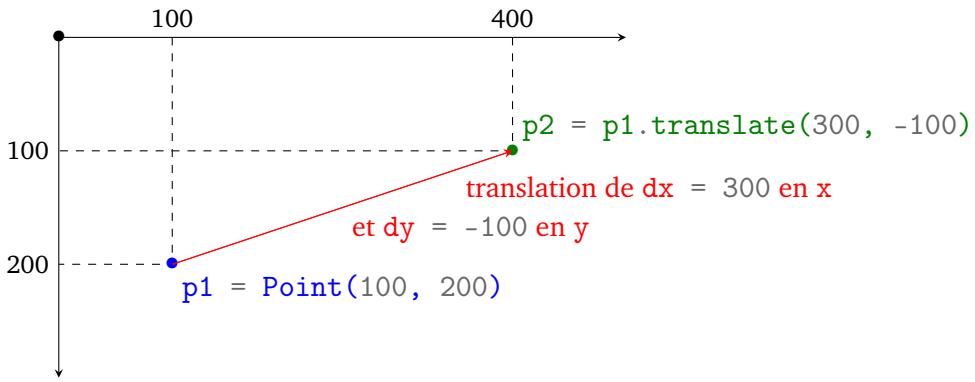
1. Écrire une classe `Point` possédant :
   - ▷ un constructeur avec deux attributs `x` et `y` de type `int` ;
   - ▷ une méthode `draw(surface, color, rayon)` qui dessine un point aux coordonnées `(x ; y)` de couleur `color` dans la `surface` ; le paramètre `rayon` (par défaut initialisé à 5) permet de dessiner un disque plus ou moins gros.
   - ▷ une méthode `drawLine(point, surface, color, width)` qui dessine une ligne de couleur `color` entre le point `self` et le point de coordonnées `(point.x, point.y)` dans la `surface` ; le paramètre `width` (par défaut initialisé à 3) permet gérer l'épaisseur du trait.

   Le code suivant doit avoir le même comportement que celui de la **partie 1** :

```
### Début de la partie à modifier ###

p1 = point.Point(100, 100)
p2 = point.Point(300, 100)
p3 = point.Point(300, 300)
p4 = point.Point(100, 300)

p1.drawLine(canvas, p2, "blue")
p2.drawLine(canvas, p3, "blue")
p3.drawLine(canvas, p4, "blue")
p4.drawLine(canvas, p1, "blue")

p1.draw(canvas, 'red')
p2.draw(canvas, 'red')
p3.draw(canvas, 'red')
p4.draw(canvas, 'red')

### Fin de la partie à modifier ###
```

Ouvrir le programme complet dans Spyder en cliquant sur le logo.

2. Réécrire le code ci-dessus en utilisant des boucles pour éviter la répétition de code.
3. Ajouter la méthode `translate(dx, dy)` qui renvoie un nouveau point issu de la translation du point de `dx` sur l'axe `x` et de `dy` sur l'axe `y`.

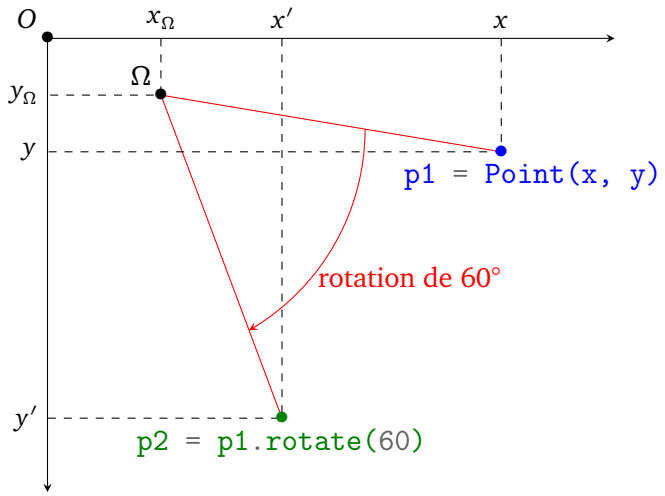


4. Le code suivant doit avoir le même comportement que celui de la **question 1** :

```
### Début de la partie à modifier ###

p1 = point.Point(100, 100)
p2 = p1.translate(200, 0)
p3 = p2.translate(0, 200)
p4 = p3.translate(-200, 0)

p1.drawLine(canvas, p2, "blue")
p2.drawLine(canvas, p3, "blue")
p3.drawLine(canvas, p4, "blue")
p4.drawLine(canvas, p1, "blue")

p1.draw(canvas, 'red')
p2.draw(canvas, 'red')
p3.draw(canvas, 'red')
p4.draw(canvas, 'red')

### Fin de la partie à modifier ###
```

Ouvrir le programme complet dans Spyder en cliquant sur le logo.

5. Réécrire le code ci-dessus en utilisant des boucles pour éviter la répétition de code.
6. Ajouter une méthode `rotate(angle)` qui crée un nouveau point en effectuant une rotation d'angle `angle` (exprimé en degrés) dans le sens des aiguilles d'une montre et de centre $\Omega(x_\Omega\ ;\ y_\Omega)$.



Pour effectuer une rotation, vous pouvez utiliser les formules suivantes où $\alpha = (\pi \times \text{angle})/180$.
Pour obtenir les coordonnées $(x'\ ;\ y')$ d'un point image du point de coordonnées $(x\ ;\ y)$ par la rotation de centre $\Omega(x_\Omega\ ;\ y_\Omega)$ et d'angle α, on utilise :

$$\begin{cases} x' &= (x-x_\Omega)\cos(\alpha)-(y-y_\Omega)\sin(\alpha)+x_\Omega \\ y' &= (x-x_\Omega)\sin(\alpha)+(y-y_\Omega)\cos(\alpha)+y_\Omega \end{cases}$$

Vous devez utiliser la bibliothèque `math` avec en particulier :

- ▷ `math.pi` pour valeur de π ;
- ▷ `math.cos(x)` : le cosinus de x (avec x exprimé en radians) ;
- ▷ `math.sin(x)` : le sinus de x (avec x exprimé en radians).

7. En utilisant la classe `Point`, programmer le dessin de la roue suivante.
   La fenêtre est de taille `400x400` ; remarquer que les disques représentant les points n'ont pas tous le même rayon, les traits n'ont pas tous la même épaisseur.
   Sauvegarder votre fichier avec le nom : `NOM_partie2_question7.py`