ISM MPSI/PCSI, Informatique
Année 2023/2024

TP Informatique 25

⚠ On commencera par importer le module `matplotlib` :

```
import matplotlib.pyplot as plt
```

Exercice 1

Écrire une fonction récursive `arbre(n)`, qui prend en argument un entier n, et qui renvoie une liste de listes représentant un arbre. La k-ième liste doit contenir toutes les branches générées lors de la k-ième récursion.

- Le cas de base correspond à la première branche, et relie les points d'affixes `0` et `1j`.
- Pour la propagation, étant donnée une branche d'extrémités a et b, on construit trois nouvelles extrémités c, d, e avec les formules suivantes :

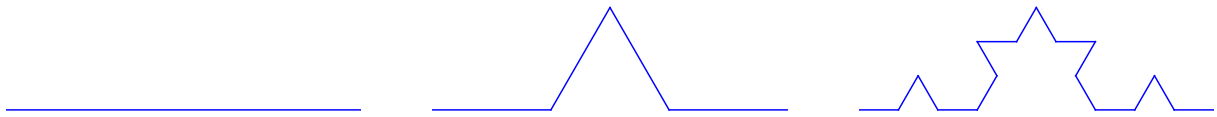
$$c = b + \frac{2}{3}(b-a)e^{\frac{i\pi}{6}}, \quad d = b + \frac{2}{3}(b-a)e^{-\frac{i\pi}{8}}, \quad e = b + \frac{1}{2}(b-a)e^{-\frac{i\pi}{5}}.$$

Tracer les arbres obtenus au bout de 4 et 7 itérations.
Pour une variable `z` de type `complex`, `z.real` et `z.imag` donnent les parties réelles et imaginaires.

Exercice 2

On veut ici tracer la courbe fractale appelée *flocon de Koch* et définie de la façon suivante :

- au départ, la courbe est un segment ;
- à chaque étape, on découpe chaque segment en 3, et on ajoute un triangle équilatéral sur le segment du milieu.



Si l'on note (x_1, y_1) et (x_2, y_2) les coordonnées des deux extrémités d'un segment, on admet que les points ajoutés dans une étape du processus ont pour coordonnées :

$$(x_3, y_3) = \left(\frac{2x_1 + x_2}{3}, \frac{2y_1 + y_2}{3}\right), \quad (x_4, y_4) = \left(\frac{x_1 + x_2}{2} + \frac{y_1 - y_2}{2\sqrt{3}}, \frac{y_1 + y_2}{2} + \frac{x_2 - x_1}{2\sqrt{3}}\right),$$

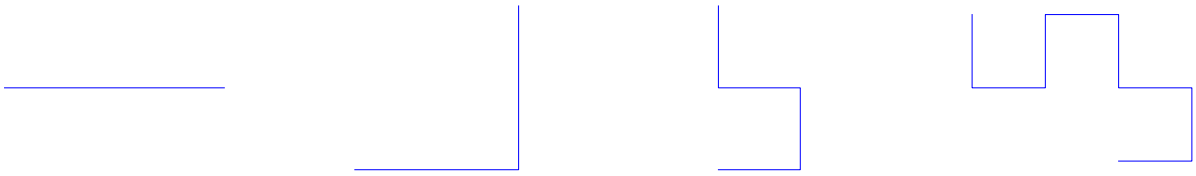
$$\text{et } (x_5, y_5) = \left(\frac{x_1 + 2x_2}{3}, \frac{y_1 + 2y_2}{3}\right).$$

1. Écrire une fonction `ajout_triangle(Lx,Ly)` qui prend en argument une liste contenant les abscisses `Lx=[x1,x2]` et une liste contenant les ordonnées `Ly=[y1,y2]` de deux points, et qui renvoie deux listes `[x1,x3,x4,x5,x2]` et `[y1,y3,y4,y5,y2]` contenant les abscisses et les ordonnées des 5 points obtenus après ajout du triangle.
2. Écrire une fonction récursive `flocon(n,Lx,Ly)` qui prend en argument un entier `n` et deux listes contenant les abscisses et les ordonnées de points, et qui renvoie les listes des abscisses et ordonnées obtenues au bout de n itérations du processus.
3. Tracer la figure obtenue au bout de $2, 3, 4, 5$ étapes.
   On peut montrer que la fractale obtenue au bout d'une infinité d'étapes en partant d'un triangle équilatéral a une aire finie, mais un périmètre infini !

Exercice 3

On s'intéresse à la *courbe du dragon* définie de la façon suivante :

- au départ, la courbe est le segment reliant le point $(0, 0)$ au point $(1, 0)$;
- à chaque étape, on reprend la courbe précédente, et on lui ajoute la rotation de cette même courbe d'angle $-\frac{\pi}{2}$ et de centre le dernier point obtenu.



1. Écrire une fonction récursive `dragon(n)` qui prend en argument un entier n, et qui renvoie la liste des points obtenus après n itérations.
   Le cas de base $n = 0$ correspond à renvoyer la liste des deux points de départ : `[[0,0],[1,0]]`.
   Dans le cas $n \geqslant 1$, on appelle la fonction au rang $n - 1$, et on ajoute à la liste obtenue les images de ces points par la rotation.
   On pourra utiliser le fait que l'image du point de coordonnées (x, y) par la rotation de centre (x_c, y_c) et d'angle $-\frac{\pi}{2}$ a pour coordonnées $(x_c + y - y_c, y_c + x_c - x)$.
   On devra obtenir :

```
>>> dragon(1)
[[0, 0], [1, 0], [1, 1]]

>>> dragon(2)
[[0, 0], [1, 0], [1, 1], [0, 1], [0, 2]]
```

2. Tracer la courbe du dragon obtenue au bout de 5, 10 et 13 itérations.