TP 11 – Dessins récursifs

1 Entrée en matière

Exercice 1 —

1. Écrire une fonction **récursive** `triangle_bas(n)` permettant d'afficher un triangle pointe en bas dont la première ligne ligne contient n étoiles.
   (on pourra utiliser la fonction `print('*'*5)` pour répéter plusieurs fois (ici 5 fois) le même caractère.

2. Écrire une fonction **récursive** `triangle_haut(n)` permettant d'afficher un triangle pointe en haut dont la dernière ligne ligne contient n étoiles.

Exemple :
```
****
***
**
*
```
Exemple :
```
*
**
***
****
```

2 Dessins de fractales

2.1 Le module `Turtle`

Pour des dessins plus avancés nous utilisons le module turtle que nous importons avec le nom `tl` avec la commande

```
import turtle as tl
```

Le module turtle permettant de manipuler une tortue, tout comme la célèbre tortue du langage Logo inventée à la fin des années 1960.
Les différentes fonctions disponibles dans ce module sont :

forward(x) : pour avancer de x pixels

backward(x) : pour reculer de x pixels

left(x) : pour tourner sur la gauche de x degrés

right(x) : pour tourner sur la droite de x degrés

goto(a,b) : pour aller en (a,b) (coordonnées en pixels)

penup() : pour lever le crayon

pendown() : pour baisser le crayon

position() : donne le couple coordonnées où se trouve le crayon/tortue

home() : pour remettre la tortue en (0,0)

reset() : pour effacer et repartir en (0,0)

color(x) : pour prendre la couleur x (qui est une chaîne, par exemple 'red')

2.2 Flocon de Von Koch

La courbe de Von Koch K_1 est obtenue à partir d'un segment de droite, en modifiant récursivement chaque segment de droite de la façon suivante :

- On divise le segment de droite en trois segments de longueurs égales.
- On construit un triangle équilatéral ayant pour base le segment médian de la première étape.
- On supprime le segment de droite qui était la base du triangle de la deuxième étape.

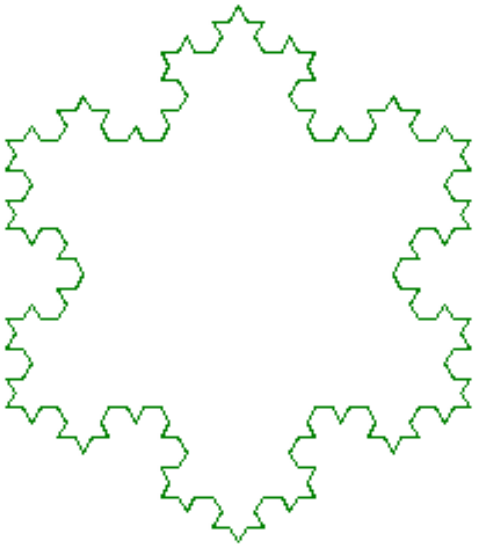
Au bout de ces trois étapes, l'objet résultant a une forme similaire à une section transversale d'un chapeau de sorcière.

On peut alors répéter les trois opérations précédentes pour chacun des 4 segments de la figure K_1. Ce faisant, on obtient la courbe de vonKoch dans le cas $n = 2$.

On recommence avec chacun des segments de K_2 pour obtenir la courbe K_3.

La courbe de Koch est la limite des courbes obtenues, lorsqu'on répète indéfiniment les étapes mentionnées ci-avant.

Le flocon de Von Koch est obtenu suivant le même procédé que précédemment, mais en partant d'un triangle équilatéral au lieu d'un segment. De façon plus simple, il est obtenu en remplaçant chacun des segments d'un triangle équilatéral par une courbes de Von Koch.



1. Écrire une fonction **récursive** `courbeVonKoch(n,cote)` qui construit la courbe K_n en partant d'un segment de longueur `cote`
2. à la fin de votre de dessin, ou est la tortue, et quelle est son orientation ?
3. Écrire une fonction `flocon (n,cote)` qui construit le flocon de Von Koch en utilisant trois courbes K_n.

2.3 Triangles de Sierpinski

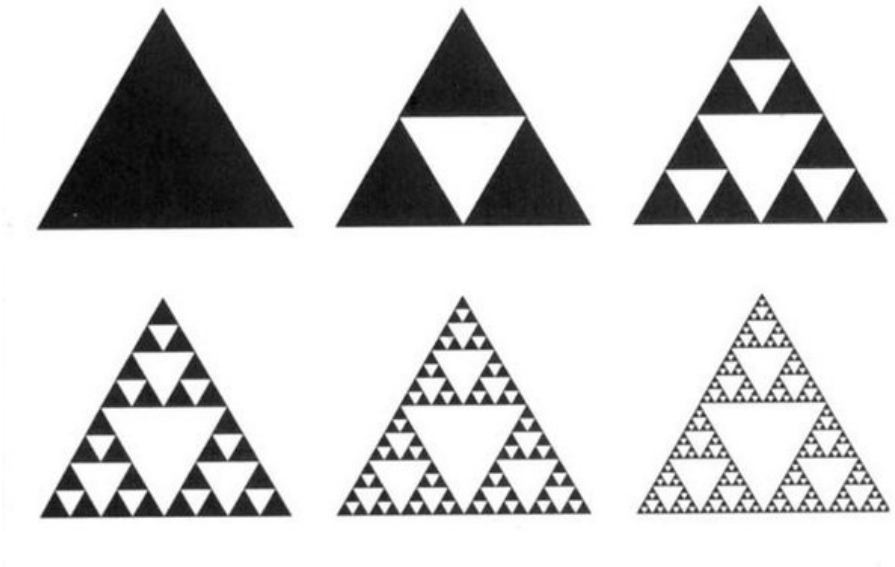
Pour tracer un triangle équilatéral plein, on peut utiliser le script suivant

```
def Triangle(cote):
    tl.begin_fill()
    for i in range(3):
        tl.forward(cote)
        tl.left(120)
    tl.end_fill()
    return
```

Premières questions :

1. Entrez ce code en Python, et testez le.
2. Que fait ce code ?
3. Ou se trouve la tortue une fois la fonction terminée ?

à partir de maintenant on va utiliser la fonction triangle pour définir une fractale appelée "Triangle de Sierpinski" :



4. **Facultatif :** tracer les 3 premiers triangles de Sierpinski
5. Écrire Une fonction **récursive** `Sierpinski(cote, n)` qui trace le n-ième triangle de Sierpinski (ou la variable `cote` prend la valeur du coté du triangle).

$\sim$