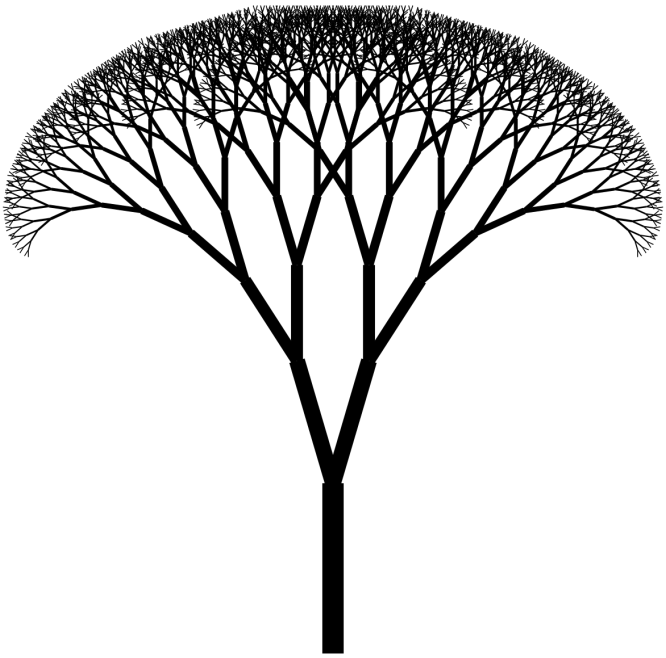
TP : Fractales et Récursivité

□ Objectifs

- Utiliser la récursivité pour générer des fractales

□ Pré-requis

- Fonctions récursives
- Module `turtle`



Définition. En Mathématiques, une **fractale** est un objet géométrique défini par un ensemble de propriétés précises, dont celle d'être **autosimilaire**, c'est-à-dire que le tout est semblable à l'une de ses parties.

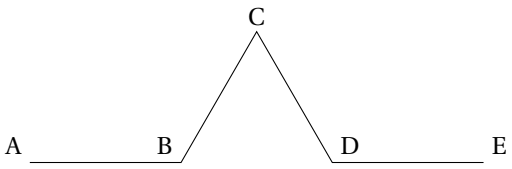
Quelques commandes de base du module turtle

- `forward(x)` : avance la tortue de `x` pixels
- `backward(x)` : recule la tortue de `x` pixels
- `right(x)` : fait pivoter la tortue sur la droite d'un angle `x`
- `left(x)` : fait pivoter la tortue sur la gauche d'un angle `x`
- `up()` : arrête l'écriture
- `down()` : démarre l'écriture
- `goto(x,y)` : déplace la tortue vers le point $(x; y)$
- `speed(x)` : change la vitesse de la tortue (`speed(0)` pour le maximum)
- `done()` : à placer en fin de programme pour laisser la fenêtre graphique ouverte

1 La courbe de Von Koch

La **courbe de Von Koch** est un bon exemple de fractale pouvant être représentée de façon récursive.
Partant d'un segment [AE] de longueur arbitraire, on sépare ce dernier en 4 parties de mêmes longueurs (avec AB = BD) :

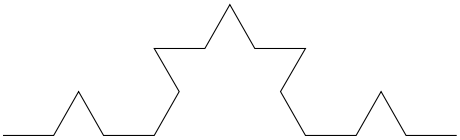
Étape 0



Étape 1

Question 01 En utilisant le module `turtle`, écrire une fonction `koch_1(L)` représentant l'étape 1 de la courbe de Koch, `L` étant la longueur initiale AE (en pixels). On utilisera uniquement les fonctions `forward`, `left` et `right`.

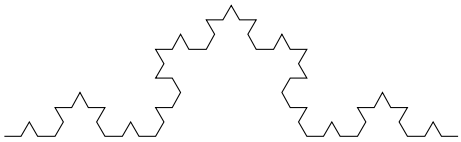
À l'étape 2, on reproduit l'étape 1 sur chacun des segments [AB], [BC], [CD] et [DE].



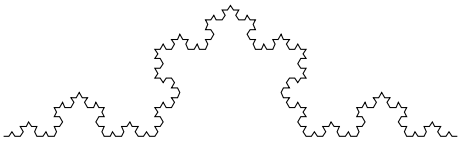
Étape 2

Question 02 Écrire une fonction `koch_2(L)` représentant l'étape 2 de la courbe de Koch, `L` étant la longueur du segment à l'étape 0. On utilisera la fonction `koch_1` définie précédemment ainsi que les fonctions `left` et `right`.

Le processus est itéré à chaque étape : on applique l'étape 1 sur chaque segment obtenu après l'étape 2, et on obtient l'étape 3, puis l'étape 4...



Étape 3



Étape 4

Question 03 Écrire la fonction `koch_3(L)` en utilisant la fonction `koch_2`.

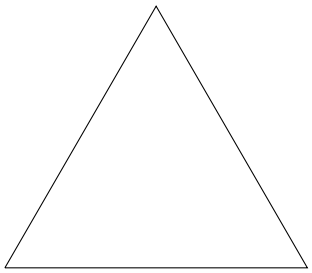
Question 04 Écrire une fonction **récursive** `koch(n, L)` représentant l'étape `n` de la courbe de Von Koch, pour une longueur initiale de `L`. Comme toute fonction récursive, `koch` sera sous la forme :

```
1  def koch(n, L):
2      if cas_de_base:
3          instruction(s) de base
4      else:
5          appel(s) récursif(s)
```

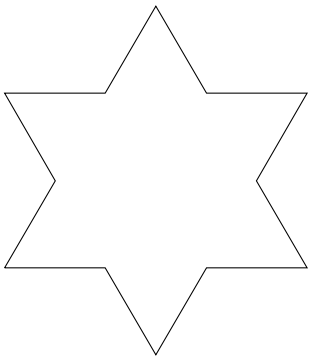
Question 05 *(Maths)* Pour $L = 1$, quelle est la longueur de la courbe de Von Koch à l'étape n ?

2 Le flocon de Von Koch

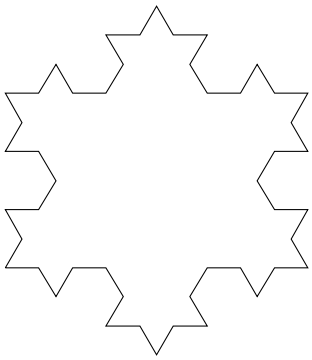
Le flocon de Von Koch est en lien étroit avec la courbe du même nom. Plutôt que de partir d'un segment, on part maintenant d'un triangle équilatéral, et on applique l'étape 1 précédente à chaque segment :



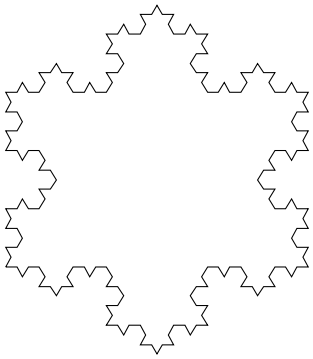
Étape 0



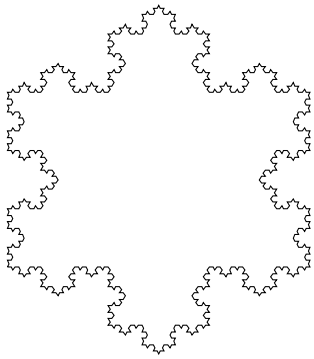
Étape 1



Étape 2



Étape 3



Étape 4

Question 06 En utilisant la fonction `koch` définie précédemment, écrire une fonction `flocon(n, L)` représentant l'étape `n` du flocon de Von Koch, partant d'un triangle équilatéral de longueur `L`.

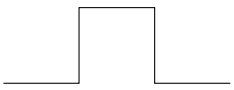
Question 07 *(Maths)* Calculer le périmètre et l'aire du flocon de Von Koch à l'étape n.
Comment varient ces deux grandeurs lorsque n devient grand ?

☞ *Aire d'un triangle équilatéral de côté c :* $\frac{\sqrt{3}}{2}c^2$

Variantes

L'étape 1 est la plus importante : c'est elle qui définit la forme de la fractale.

Question 08 Que deviennent la courbe et le flocon de Von Koch si l'étape 1 est définie comme ci-dessous ?



Question 09 Définissez vous-même la transformation de l'étape 1 et admirez vos résultats...

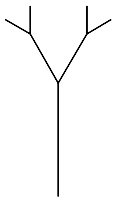
3 Arbre fractal

Un arbre fractal est un arbre dont chaque branche est encore un arbre, mais de taille réduite.

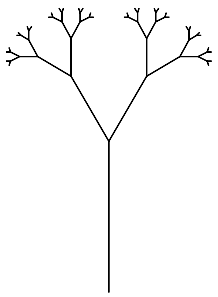
Question 10 Écrire une fonction `arbre_1(L)` dessinant une branche de longueur `L`, et **ramenant la tortue au point de départ**.
On utilisera uniquement les fonctions `forward` et `backward`.

Question 11 Écrire une fonction `arbre_2(L)` dessinant un arbre possédant un tronc de longueur `L` et 2 branches de longueur `L/2` et faisant un angle de 60°.
On utilisera les fonction `arbre_1`, `forward`, `backward`, `left` et `right`.
On dira que cet arbre est de **profondeur 2**.

Question 12 Écrire une fonction `arbre_3(L)` dessinant un arbre possédant un tronc de longueur `L` et 2 branches possédant chacune 2 branches.
On utilisera les fonction `arbre_2`, `forward`, `backward`, `left` et `right`.
On dira que cet arbre est de **profondeur 3**.



Question 13 Écrire une fonction **récursive** `arbre(n, L)` dessinant un arbre fractal possédant un tronc de longueur `L` et de profondeur `n`, chaque branche mesurant la moitié de la longueur de la branche de laquelle elle est issue, l'angle entre deux branches étant de 60°.

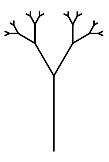


Arbre de profondeur 6

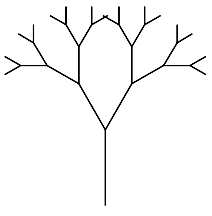
Notre arbre fait de la peine ! Le problème vient du **rapport de réduction** égal à 0,5 ici (chaque branche fait la moitié de la précédente...). On peut modifier ce rapport afin d'obtenir un arbre moins « fatigué ».

Question 14 Modifier la fonction `arbre` précédente, en rajoutant un paramètre `R` égal au rapport de réduction utilisé.

`arbre(5,100,0.3)`

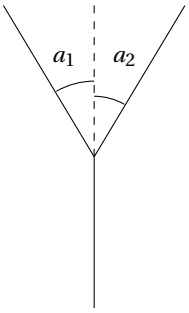


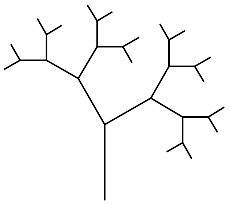
`arbre(5,100,0.5)`



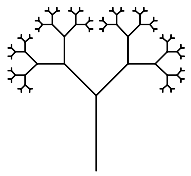
`arbre(5,100,0.7)`

Question 15 Modifier encore la fonction `arbre` en ajoutant deux paramètres `a_1` et `a_2`, les angles faits par les branches avec la branche dont elles sont issues.



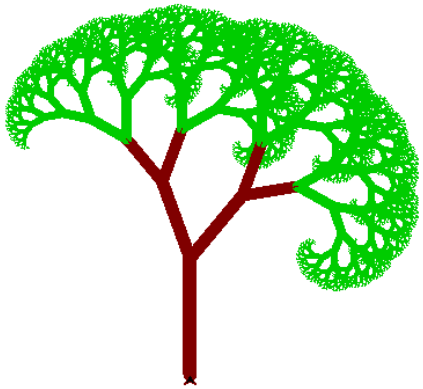


`arbre(5,100,0.7,30,60)`



`arbre(7,100,0.6,45,45)`

Question 16 Modifier la taille et la couleur de la tortue au cours des itérations afin d'obtenir un joli résultat :



4 Des fractales, encore et encore...

Besoin d'idées ? Regardez par là :

`https://mathcurve.com/fractals/fractals.shtml`