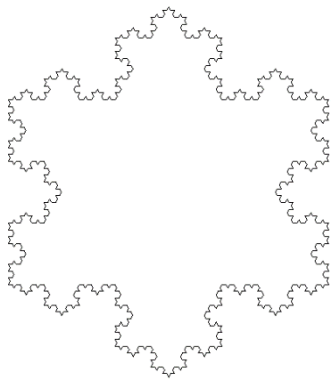
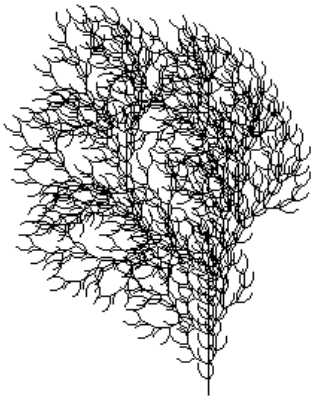
T NSI

Dessiner des fractales et simuler la croissance de végétaux avec Turtle





A. Initiation au module turtle

Le module **turtle** (tortue, en français) de Python permet de réaliser des dessins à l'aide de la programmation. L'utilisateur doit déplacer un objet appelé "tortue" sur l'écran à l'aide de commandes simples (avance, recule, tourne à droite, etc.). A chaque déplacement, la tortue laisse un trait derrière elle et ainsi on peut créer des dessins. La fenêtre graphique est munie d'un repère orthonormé, d'origine le centre de la fenêtre. C'est le point de départ de la tortue. Au départ, la tortue est orientée vers la droite.
►Pour charger le **module turtle**, taper au début du programme : **from turtle import ***
►**Terminer** le programme principal par **mainloop().**

On dispose d'une liste partielles d'instructions données ci-dessous :

forward(x) ou fd(x) : permet de déplacer la tortue de x pixels en avant
backward(x) ou bk(x) : permet de déplacer la tortue de x pixels en arrière
right(x) ou rt(x) : permet de tourner la tortue vers le la droite de x degrés
left(x) ou lt(x) : permet de tourner la tortue vers le la gauche de x degrés
penup() ou pu() ou up() : permet de lever le crayon (le déplacement de la tortue ne laissera plus de trace)
pendown() ou down() ou pd() : permet de baisser le crayon
goto(x,y) : déplace la tortue au point de coordonnées (x,y)

1. Un premier carré

Dans l'éditeur de programmes, taper le programme suivant et le lancer :

```
fd(100)
lt(90)
fd(100)
lt(90)
fd(100)
lt(90)
fd(100)
lt(90)
mainloop()  #évite de faire planter le programme
```

fd(100) signifie forward(100) = avance de 100
lt(90) signifie left(90) = tourne à gauche de 90°

On voit que l'on a **répété 4 fois** la même série d'instructions fd(100) ; lt(90) ;
On peut éviter cette répétition en utilisant une **boucle for**.

2. Avec une boucle for

On remplace la série d'instructions précédentes par :

```
for i in range(4) :
        fd(100)
        lt(90)
mainloop()
```

3. Avec une fonction

On va créer maintenant une fonction carre(*a*) qui va tracer un carré de côté *a* :
Remplacer la série d'instructions précédentes par :

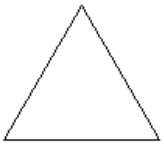
```
def carre(a):
    for i in range(4):
        fd(a)
        lt(90)

carre(150)   # on donne l'ordre de tracer un carré de côté 150
mainloop()
```

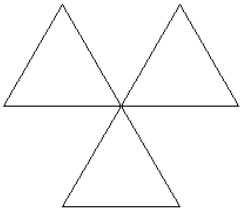
4. Exercices

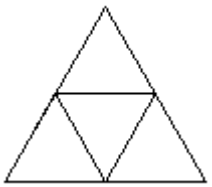
1) Tracer un carré et ses diagonales (conseil pour les diagonales : utiliser goto).
La tortue doit revenir à sa position initiale, avec son angle initial.

2) Créer une fonction **triangle(*a*)** qui trace un triangle équilatéral de côté *a*.

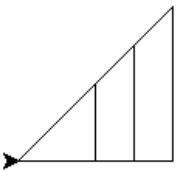


3) Créer des fonctions qui tracent les figures suivantes, en utilisant la fonction **triangle(*a*)** précédente et une **boucle for** :





4) a) Créer une fonction **trirect(*a*)** qui trace un triangle rectangle isocèle de côté *a*.
b) Utiliser cette fonction pour tracer la figure ci-dessous, formée de 3 triangles rectangles isocèles, dont les côtés sont agrandis à chaque fois de la même valeur :



B. Représentation d'un déplacement avec une chaîne de caractères

On suppose maintenant que la tortue se déplace en utilisant uniquement 3 instructions :

forward(longueur) # avance d'une certaine longueur à préciser.
right(60) # tourne à droite de 60°
left(60) # tourne à gauche de 60°

On décide de coder les déplacements de la tortue avec une chaîne de caractères, où chaque caractère représente l'un des déplacements précédents.
Ainsi, forward(longueur) est codé par le caractère "A", left(60) est codé par "+" et right(60) est codé par "-".

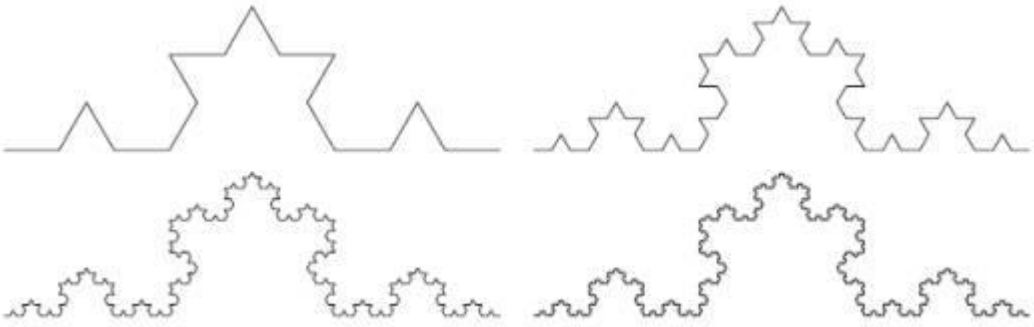
1) Exemple : tracer à main levée le trajet de la tortue représenté par la chaîne :
"A++A- -A"

2) Créer une fonction Python **trace(*chaine, longueur*)** qui a pour argument une chaîne de caractères (*chaine*) comme celle précédente (formée uniquement avec les caractères A, + et -) et un entier positif (longueur), égal à la longueur (en pixels) d'un déplacement de la tortue en ligne droite. Cette fonction doit faire tracer par la tortue le chemin codé par la chaîne de caractère *chaine*.

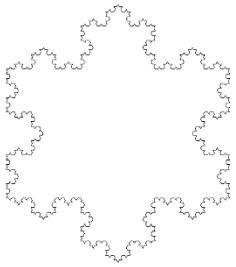
C. Tracé d'une courbe fractale : le Flocon de Koch

Le flocon de Koch est une figure dite "fractale", qui se construit en remplaçant un segment de longueur initiale *a* par 4 segments de longueur *a* / 3 de la façon suivante :

Le "triangle central" est équilatéral.
En réitérant le procédé plusieurs fois, on obtient les courbes successives suivantes (échelles de longueur non respectées)



En partant d'un triangle initial équilatéral, on obtient, après quelques itérations, la figure suivante, semblable à un flocon de neige :

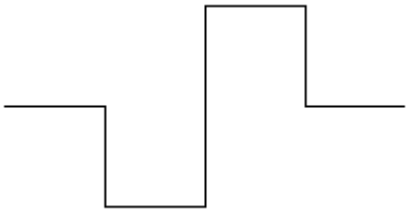


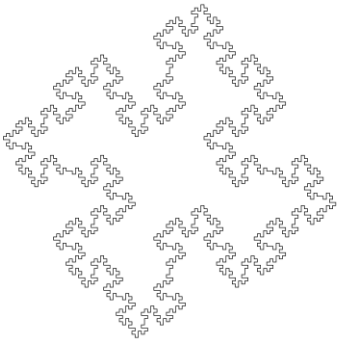
1) Donner la chaîne de caractères permettant de tracer le motif
(ne pas se préoccuper de la longueur effective d'un segment)

2) Créer une fonction Python **deplacement(s, motif, n)** qui retourne une chaîne de caractères (chaine), fabriquée de la façon suivante : à partir de la chaîne **s** de départ comportant les caractères A, + ou -, on remplace tous les A par la chaîne **motif**, ceci n fois successivement.
Par exemple, avec s = "A+A" et motif = "A-A" :
deplacement("A+A", "A-A", 1) doit retourner "A-A+A-A"
deplacement("A+A", "A-A", 2) doit retourner "A-A-A-A+A-A-A-A"

3) Tracer le flocon de Koch, en utilisant successivement les fonctions **deplacement** et **trace**.
Pour s, on prendra une chaine représentant un triangle équilatéral, et pour motif, on prendra la chaine de la question 1. Faire le tracé avec n = 3 ou 4.

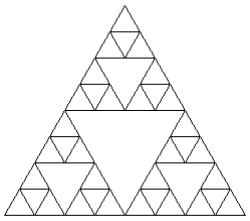
4) En partant d'un carré, et avec le motif suivant, on obtient la courbe fractale dite "courbe quadratique de Koch de type 2"





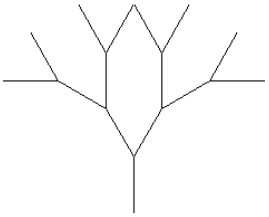
D. Tracé d'une courbe fractale : le triangle de Sierpinsky

On a tracé ici le triangle de Sierpinsky pour n = 3.
La chaine s de départ représente un triangle équilatéral.
Trouver le motif par lequel on doit remplacer chaque caractère A de s pour obtenir le triangle de Sierpinsky.



E. Simulation de la croissance de végétaux : dessiner des arbres

1) Dessiner un arbre "simple".



On commcera par orienter la tortue vers le haut, en faisant left(90).
Le principe est le même que pour les courbes fractales précédentes, mais il s'agit de remplacer "A" par le motif "A[+A][-A]".
Pour la figure présentée, l'angle de rotation est de 30°.
Le premier A trace le tronc.
[+A] trace la branche de gauche et [-A] celle de droite.

►Lorsqu'on rencontre un crochet ouvrant "[", il faut mémoriser dans une **pile** la position de la tortue (=ses coordonnées (x,y)) ainsi que son angle.
L'angle de la tortue est donné par **heading()** et le couple de coordonnées (x,y) est donné par **position()**. On stocke donc ces deux données dans une liste, que l'on **empile** dans une pile.

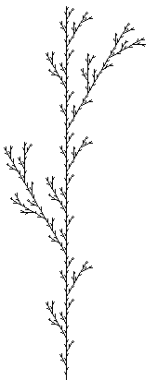
►Lorsqu'on rencontre un crochet fermant "]", on **dépile** la liste contenant l'angle et le couple de coordonnées de la tortue, mémorisés précédemment. On "téléporte" alors la tortue à cette position (en levant le crayon pour ne pas tracer de trait), en lui donnant l'angle mémorisé avec la fonction setheading(angle), et on baisse le crayon.

a) Créer une liste vide représentant une **pile**.
Créer deux fonctions **empiler()** et **dépiler()**.
empiler() doit empiler dans la pile la liste [angle, coordonnées] de la tortue.
dépiler() doit :
-lever le crayon
-dépiler la liste [angle, coordonnées] de la pile
-positionner la tortue aux coordonnées ainsi récupérées
-donner l'angle récupéré à la tortue
-baisser le crayon

b) Modifier la fonction **trace(chaine, longueur)** pour qu'elle lance empiler() à chaque fois que le caractère "[" est rencontré dans la chaine, et dépiler() lorsque la caractère "]" est rencontré.

c) Lancer la fonction **deplacement** avec pour chaine de départ "A" et pour motif "A[+A][-A]", puis la fonction **trace** appliquée à la chaine récupérée, avec n = 3, pour obtenir l'arbre.

2) Tracé d'une plante longiligne



Partir de la chaine de départ "A" et prendre pour motif "A[+A]A[-A]A".
Prendre n = 4 et longueur = 5.

3) Tracé d'un buisson de type "aneth"

Partir de la chaine de départ "A" et prendre pour motif :
"AA+[+A-A-A]-[-A+A+A]".
Prendre n = 4 et longueur = 5.