L. DIDIER & R. CABANE

Le flocon de Von Koch, courbe fractale

Dans le programme (spécialité Première)

Contenus Exemples de modes de génération d'une suite : explicite $u_n = f(n)$, par une relation de récurrence $u_{n+1} = f(u_n)$, par un algorithme, par des motifs géométriques. Suites géométriques : exemples, définition, calcul du terme général.	Sur des exemples, introduction intuitive de la notion de limite, finie ou infinie, d'une suite.

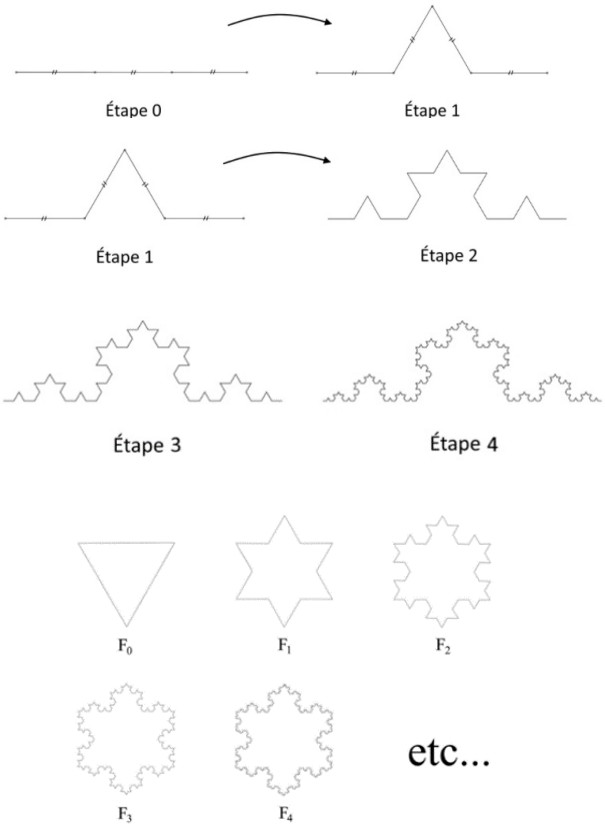
Situation déclenchante

Le flocon de Koch, imaginé en 1904 par le mathématicien suédois Helge Von Koch, est l'une des premières courbes fractales à avoir été décrites. Les fractales permettent notamment de modéliser le développement de certaines bactéries.

Dans cet exercice, nous allons montrer une propriété assez étonnante que possède le flocon de Koch.

La transformation consiste à découper le segment initial en trois segments de même longueur, puis à construire un triangle équilatéral ayant pour base le segment du milieu, et enfin à retirer ce segment servant de base.

On applique désormais ces transformations en prenant pour figure initiale un triangle équilatéral dont les côtés sont de longueur 1, et on applique la transformation de façon à ce que l'aire de la figure obtenue soit supérieure à celle de la figure précédente. Soit n un entier naturel, on note F_0 le triangle équilatéral initial et F_n la figure obtenue à l'étape n.



Buts à atteindre

1. Écrire une fonction Python qui prend comme paramètre un entier naturel n et qui renvoie le nombre de côtés de la figure F_n ainsi que la longueur de ses côtés.
2. Écrire une fonction Python qui prend comme paramètre un entier naturel n et qui renvoie le périmètre de la figure F_n ainsi que l'aire de cette figure. Quelle conjecture peut-on faire sur le périmètre et l'aire de cette figure pour de grandes valeurs de n ?
3. **TI-83** Écrire un script Python à l'aide du module Turtle qui permet de tracer la figure F_n.

L. DIDIER & R. CABANE

Fiche méthode

Proposition de résolution

Pour atteindre l'objectif 1 :

Une fonction `figure` qui prend comme argument un entier naturel n. Cette fonction renvoie dans une liste le nombre de cotés ainsi que la longueur d'un côté à l'étape n. Ce sont en fait des suites géométriques.

```
ÉDITEUR : FLOCON
LIGNE DU SCRIPT 0010
from math import *

def figure(n):
  cote=3
  longueur=1
  for i in range(n):
    cote=cote*4
    longueur=longueur/3
  return [cote,longueur]
```

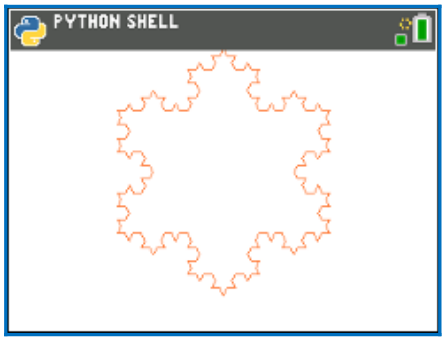
Pour atteindre l'objectif 2 :

Une fonction `floc` qui prend comme argument un entier naturel n et qui renvoie dans une liste le périmètre du flocon à l'étape n ainsi que l'aire de la figure à l'étape n.

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # L'exécution de FLOCON
>>> from FLOCON import *
>>> figure(4)
[768, 0.01234567901234568]
>>> flocon(4)
[9.481481481481481, 0.6826830343
504216]
```

TI-83 Pour atteindre l'objectif 3 :

- Une fonction `courbeVonKoch` qui permet de définir la transformation de Von Koch.
- Une fonction `trianglekoch` qui permet de définir la figure initiale sur laquelle on applique la transformation.
- Une fonction `trace` qui permet de régler les paramètres d'affichage et qui permet de tracer la figure souhaitée.



Étapes de résolution

Pour atteindre l'objectif 1 :

On initialise le nombre de côtés à 3 et la longueur du côté à 1. Lors de la transformation de Von Koch, à chaque étape, le nombre de côtés est multiplié par 4 et la longueur de chacun est divisée par 3 (voir le code ci-contre en haut).

```
ÉDITEUR : FLOCON
LIGNE DU SCRIPT 0018
def floc(n):
  rac=sqrt(3)/4
  aire=rac # aire 1er triangle
  f=fig(0) # [nbre côtés,long.]
  for i in range(n):
    g=fig(i+1)
    aire=aire+f[0]*rac*g[1]**2
    f=g
  peri=f[0]*f[1]
  return [peri,aire]
```

Pour atteindre l'objectif 2 :

On rappelle que l'aire d'un triangle équilatéral de côté b est égale à $\frac{\sqrt{3}}{4}b^2$.

Pour calculer l'aire de la figure F_{n+1}, il faut additionner l'aire de la figure F_n avec l'aire des triangles équilatéraux de côtés de longueur L_{n+1} (où L_n désigne la longueur d'un côté de la figure F_n, valeur calculée grâce à l'instruction `figure(i+1)[1]`). Le nombre de ces triangles est calculé par `figure(i+1)[0]` (rappel : dans une liste `L=[3,8]`, le premier élément est `L[0]`, valant 3 ici).

TI-83 Pour atteindre l'objectif 3 :

Pour effectuer le tracé il faut importer la bibliothèque `Turtle`[1].

La fonction `courbeVonKoch` permet de définir la transformation de Von Koch à l'aide d'un appel récursif (c'est-à-dire, quand la fonction s'appelle elle-même, ici quatre fois).

```
ÉDITEUR : TRACEFLO
LIGNE DU SCRIPT 0003
from ce_turtl import *
from math import *

def courbeVonKoch(n,cote):
  if n == 0 :
    turtle.forward(cote)
  else :
    courbeVonKoch(n-1, cote/3)
    turtle.left(60)
    courbeVonKoch(n-1, cote/3)
    turtle.right(120)
    courbeVonKoch(n-1, cote/3)
    turtle.left(60)
    courbeVonKoch(n-1, cote/3)
```

La fonction `trianglekoch` permet de définir la figure initiale sur laquelle on applique la transformation. Ici on applique la transformation sur chaque côté du triangle équilatéral montré ci-contre.

La fonction `trace` permet de régler les paramètres d'affichage et permet de tracer la figure souhaitée.

L'utilisation du module `turtle` est détaillée dans l'appendice 2.

```
ÉDITEUR : TRACEFLO
LIGNE DU SCRIPT 0027
def trianglekoch(n,cote):
  for i in range(3) :
    courbeVonKoch(  n, cote  )
    turtle.right(120)
```

Les instructions utilisées ici :

```
ÉDITEUR : TRACEFLO
LIGNE DU SCRIPT 0032
def trace(n):
  turtle.clear()
  turtle.penup()
  turtle.home()
  turtle.goto(-80,60)
  turtle.pendown()
  turtle.pensize(0)
  turtle.color(255,128,64)
  turtle.speed(1)
  trianglekoch(n,160)
  turtle.show()
```

`clear` : efface l'écran
`penup` : lève le stylet
`home` : tortue au centre, tournée vers la droite
`goto` : déplace la tortue vers ...
`pendown` : baisse le stylet
`pensize` : taille du stylet (petite)
`color` : couleur du tracé (Rouge – Vert – Bleu)
`speed` : vitesse du tracé (rapide)

`show` : termine le tracé, se met en attente

1 Le module complémentaire `ce_turtl` ou `Turtle` doit être auparavant téléchargé puis installé (voir l'appendice 2).

Pour aller plus loin

Approfondissements possibles

- Effectuer divers tracés en changeant couleurs, tailles, etc.
- Quand on teste `floc(n)` pour des valeurs croissantes de n, le périmètre de la figure F_n augmente et semble tendre vers l'infini avec n : pourquoi ?

```
PYTHON SHELL
>>> [floc(i) for i in range(16)]
[[3, 0.433], [4.0, 0.577], [5.3,
 0.642], [7.1, 0.67], [9.5, 0.68
3], [12.6, 0.6879999999999999],
[16.9, 0.6909999999999999], [22.
5, 0.692], [30.0, 0.692], [40.0,
 0.693], [53.3, 0.693], [71.0000
0000000001, 0.693], [94.70000000
000001, 0.693], [126.3, 0.693],
[168.4, 0.693], [224.5, 0.693]]
```

- L'aire de la figure F_n semble tendre vers une limite finie (0,692...) quand n tend vers l'infini : c'est le phénomène dit de la « longueur des côtes de la Bretagne » ! Mais pourquoi ?

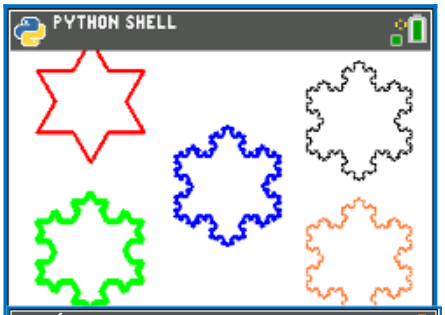
Voici comment le justifier mathématiquement :

On note a_n l'aire de la figure à l'étape n.

1) On pourra justifier, en argumentant, que pour tout entier naturel n, $a_{n+1}=a_n+\frac{\sqrt{3}}{12}\left(\frac{4}{9}\right)^n$.
2) En déduire que pour tout entier naturel n, $a_n=\frac{\sqrt{3}}{4}+\frac{3\sqrt{3}}{20}\left(1-\left(\frac{4}{9}\right)^n\right)$.
3) Vérifier votre conjecture.

La figure complète :

Le programme ayant produit la figure du haut de cette page est reproduit ci-contre. Sa longueur est surtout due aux commandes graphiques.




```
ÉDITEUR : SNOWFLK2
LIGNE DU SCRIPT 0011
from turtle import *
t=Turtle()

def side(n,l):
    if n==1:
        t.forward(l)
        return
    else:
        side(n-1,l)
        t.left(60)
        side(n-1,l)
        t.right(120)
        side(n-1,l)
        t.left(60)
        side(n-1,l)

colors=[(255,0,0),(0,255,0),(0,0
    ,255),(200,0,120),(180,180,
    0)]

def koch(n,l,a,b):
    for i in range(n-1):
        l=l/3
    t.penup()
    t.goto(a,b)
    t.pendown()
    t.pensize(0)
    cc=colors[n-1]
    t.pencolor(cc[0],cc[1],cc[2])
    for i in range(3):
        side(n,l)
        t.right(120)
    t.penup()

t.clear()
t.hideturtle()
t.speed(10)
koch(1,80,-140,90)
koch(2,80,-140,-25)
koch(3,80,-35,25)
koch(4,80,65,75)
koch(5,80,65,-30)
t.show()
```

Une sélection de couleurs pour les figures successives

penup : permet de ne pas tracer tous les déplacements qui seront effectués en dessous.

goto : envoie la souris à un point défini par coordonnées.

pendown : permet de tracer tous les déplacements qui seront effectués en dessous.

clear : pour effacer l'écran avant d'effectuer les tracés.