Exercices de Programmation & Algorithmique 1

Série 3 – Les fractales avec UTurtle
(6 octobre 2022)

Département d'Informatique – Faculté des Sciences – UMONS

Pré-requis : Comprendre la notion de récursivité; connaître les commandes de base de uturtle (cours jusqu'au **Chapitre 5** et Tutoriel 1. Utilisation du module uturtle).

Objectifs : Découvrir et comprendre la notion de fractales; savoir écrire des fonctions récursives; savoir utiliser les fonctions d'un module externe.

---

1 Le contrat

1.1 Introduction

Une fractale est un objet (ou une valeur) qui, lors de son affichage, présente des similarités quelque soit l'échelle (le "zoom") à laquelle cet objet est affiché. L'objet ne doit pas exactement présenter les mêmes structures quelque soit l'échelle considérée, mais doit présenter un ensemble de similarités structurelles. Un exemple de fractale simple est repris sur la figure 1.

FIGURE 1 – Étapes de construction du triangle de Sierpinski.

1.2 La courbe de Koch

La figure 2 introduit une nouvelle fractale appelée courbe de Koch.

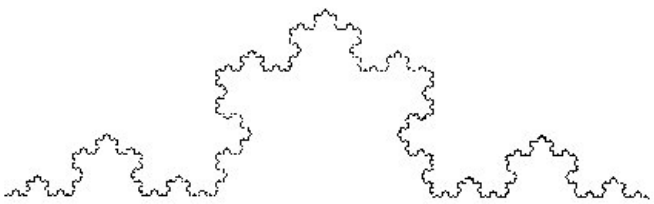

FIGURE 2 – Un exemple de courbe de Koch.

Pour dessiner une courbe de Koch d'une certaine longueur x, il faut tout d'abord dessiner une courbe de Koch du tiers de cette longueur, puis se tourner vers la gauche de 60 degrés, dessiner à nouveau une courbe de Koch du tiers de la longueur x, se tourner vers la droite de 120 degrés, à nouveau dessiner une courbe de Koch de longueur $\frac{x}{3}$, se tourner vers la gauche de 60 degrés et enfin dessiner une dernière fois une courbe de Koch de longueur $\frac{x}{3}$. Si, lors du dessin de la courbe de Koch, la longueur x est inférieure à un seuil donné, alors au lieu d'appliquer cette procédure, il faut juste dessiner une ligne droite de longueur x.

1 Vous trouverez sur moodle un fichier `recursif.py`. Ce fichier contient différentes fonctions qui doivent calculer la somme de tous les nombres entiers de 0 à n. Une seule de ces fonctions est correcte. Il vous est demandé de retrouver la fonction qui est correcte et, pour chacune des autres fonctions, d'indiquer quelle est l'erreur et pourquoi cela ne fonctionne pas. Vous pouvez vous aider du site `pythontutor.com/visualize.html` pour analyser ce qu'il se passe.

2 La fonction `triangle(n, size)` ci-dessous dessine une fractale. Cette fonction prend en entrée une tortue `t`, un naturel `n` et une taille `size`. **Sur papier**, appliquez cette fonction pour $n = 0$ et $n = 1$. Vous pouvez ensuite vérifier votre résultat à l'aide de l'ordinateur.

```
def triangle(t, n, size):
    if n == 0:
        moveForward(t, size)
        turnLeft(t, 120)
        moveForward(t, size)
        turnLeft(t, 120)
        moveForward(t, size)
        turnLeft(t, 120)
    else:
        triangle(t, n-1, size/2)
        moveForward(t, size/2)
        triangle(t, n-1, size/2)
        moveBackward(t, size/2)
        turnLeft(t, 60)
        moveForward(t, size/2)
        turnRight(t, 60)
        triangle(t, n-1, size/2)
        turnLeft(t, 60)
        moveBackward(t, size/2)
        turnRight(t, 60)
```

3 Écrivez une fonction `koch(t, x, seuil)` prenant en paramètre une variable `t` de type `Turtle`, une variable `x` représentant la longueur de la courbe de Koch à dessiner et une variable `seuil` indiquant à la fonction le seuil au delà duquel la courbe ne doit pas être dessinée.

4 Écrivez une fonction `flocon`, prenant les mêmes paramètres, qui dessine quelque chose ressemblant à un flocon de neige en utilisant la courbe de Koch.

5 Écrivez une fonction `carre` prenant les mêmes paramètres. Cette fonction dessine une variante de la courbe de Koch dans laquelle la figure de base est un carré plutôt qu'un triangle. Le résultat attendu est illustré à la figure 3.

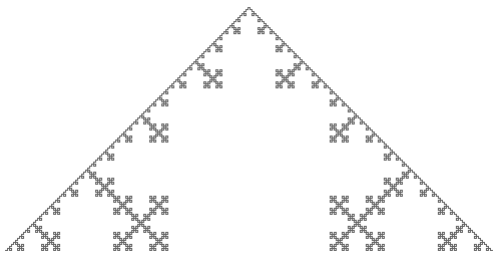

FIGURE 3 – Un exemple de courbe de Koch carrée.

1.3 L'arbre

Utilisez la tortue du module uturtle pour dessiner un arbre. Un tel arbre est composé d'un tronc se séparant en un certain nombre de branches. Chacune de ces branches, à son tour, se divise en ce même

nombre de branches et ainsi de suite jusqu'à obtenir un certain nombre de branches imbriquées. Un tel arbre est présenté sur la figure 4. Sur cet arbre, les branches sont à chaque fois divisées en 2 jusqu'à atteindre 5 branches imbriquées. Remarquez également que la taille des branches et l'angle entre elles diminue à chaque niveau de profondeur.

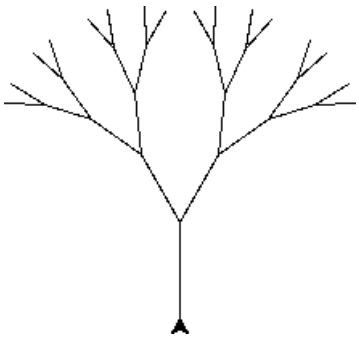

FIGURE 4 – Arbre dessiné avec la tortue.

6 Écrivez une fonction `arbre(t, x, a, n)` prenant en paramètre une variable `t` de type `Turtle`, une variable `x` représentant la longueur du tronc, une variable `a` représentant l'angle de branchement, et une variable `n` représentant le nombre souhaité de branches imbriquées (la "profondeur" de l'arbre).

2 Exercices complémentaires

Écrivez **récursivement** les fonctions suivantes :

★☆☆ 7 Une fonction `puissance(x, n)` qui calcule et retourne la n-ième ($n \in \mathbb{N}$) puissance de `x` sans utiliser l'opérateur `**`.

★☆☆ 8 Une fonction `contient(n, d)` qui retourne vrai si l'entier positif représenté par `n` contient le digit ($\in \{0, \ldots, 9\}$) représenté par `d` (astuce : utilisez la division et le modulo).

★★☆ 9 Une fonction `pgcd(x, y)` qui calcule le plus grand commun diviseur de `x` et `y`, deux entiers positifs. Pour cela, utilisez la méthode d'Euclide. Cette méthode se base sur l'observation suivante : si r est le reste de la division de x par y, alors le pgcd de x, y est égal au pgcd de y, r. Considérez comme cas de base `pgcd(x, 0) = x`.

★☆☆ 10 Une fonction `est_multiple(n, d)` qui retourne vrai si l'entier positif `n` est un multiple de l'entier positif `d`. Vous ne pouvez pas utiliser `/` ni `%`.

★☆☆ 11 Une fonction `ackermann(m, n)` qui implémente la fonction de Ackermann $A(m, n)$ définie comme suit :

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1, A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0. \end{cases}$$

Vérifiez que `ackermann(3, 6)` donne 509. Que se passe-t-il pour de plus grandes valeurs ?

★★☆ 12 Une fonction `triangle_pascal(i, j)` qui, étant donnés deux entiers positifs `i` et `j`, retourne la valeur située à la i-ème ligne et la j-ème colonne du triangle de Pascal. Pour rappel, le triangle de Pascal est construit comme suit :

— La valeur en $(0, 0)$ vaut 1
— Pour toute case du triangle, la valeur de cette case est la somme de la valeur située au-dessus et de celle située au-dessus à gauche.
— Pour toutes les autres "cases", considérez une valeur nulle.

★★★ 13 Toute expression mathématique "simple" peut être représentée à l'aide d'un triplet `(a, op, b)` où `a` et `b` sont les opérandes de l'opérateur représenté par `op`. Par exemple, $14 + 2.5$ peut s'écrire `(14, '+', 2.5)`. Il est possible de combiner ces expressions simples afin d'obtenir des expressions plus complexes.
Par exemple, $(18 * 3) + (14/2) - 1$ peut s'écrire `(((18, '*', 3), '+', (14, '/', 2)), '-', 1)` .

Écrivez une fonction récursive `evaluer(exp)` qui, étant donnée une expression arithmétique telle que présentée ci-dessus, retourne le résultat de l'évaluation de cette expression. Veuillez noter que :

— seuls les opérateurs $+, -, *, /$ sont considérés et représentés à l'aide des chaînes de caractères `'+'`, `'-'`, `'*'`, `'/'`.
— on souhaite que l'opérateur `'/'` corresponde à la division "réelle" (par opposition à la division entière).
— Etant donnée une variable `x` en Python, les tests suivants retournent vrai si `x` est un tuple :
  — `isinstance(x, tuple)`
  — `type(x) == type((1, 2))`

Exam 14 (20 novembre 2009) Problème 1 : Affichage récursif.

Exam 15 (18 janvier 2011) Problème 4 : Approximation d'une racine d'une fonction.

Exam 16 (26 octobre 2011) Problème 3 : Intersection de fonctions.

Exam 17 (13 juin 2012) Problème 2 : Aire d'un polygone.