Activité 06 - Fractales

Equipe Pédagogique LU1IN0*1

Consignes : Cette activité se compose d'une première partie guidée, suivie de suggestions. Il est conseillé de traiter en entier la partie guidée avant de choisir une ou plusieurs suggestions à explorer.

L'objectif de cette activité est la représentation graphique de fractales en utilisant des listes, des couples, et les primitives de dessin données dans l'exercice 1.9 du cahier d'exercices.
La plupart des fonctions de cette activité renvoient des `Image`. A ce titre, on ne pourra pas accompagner ces fonctions d'un jeu de test, et leur "test" devra se faire avec `show_image`, en comparant visuellement l'affichage au résultat espéré.

Fractales. Selon *Wikipedia*, une *fractale* est un objet mathématique qui présente une structure similaire à toutes les échelles. On consultera `https://fr.wikipedia.org/wiki/Fractale` pour plus de détails.

1 Partie guidée : Courbes

On définit deux types pour les points et les courbes :

```
Point = Tuple[float, float]
Courbe = List[Point]
```

Un *point* est donc représenté par son abscisse et son ordonnée. Une *courbe* est une liste de points, représentant la ligne brisée obtenue en reliant, dans l'ordre, les points de la liste.

Par exemple voici une courbe représentant un triangle rectangle :

```
O : Point = (0.0, 0.0)
tri1 : Courbe = [O, (0.0,0.3), (0.4,0.0), O]
```

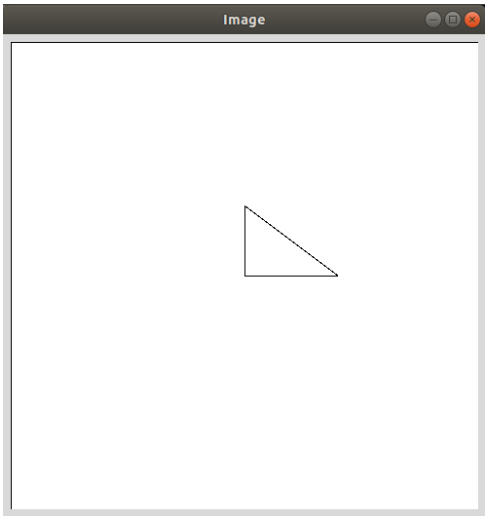
Question 1. Ecrire une fonction `longueur`, qui prend en entrée une courbe et calcule sa longueur.

```
assert abs((longueur_courbe(tri1) - 1.2)) < 10 ** -12
```

Question 2. Ecrire une fonction `segment`, qui prend en entrée deux points, et renvoie l'image correspondant au segment reliant ces deux points.

Question 3. Ecrire une fonction `image_courbe`, qui prend en entrée une courbe et renvoie l'image correspondant à cette courbe (c'est-à-dire, l'image obtenue en reliant les points de la courbe, dans l'ordre, par des segments)

Par exemple, `show_image(image_courbe(tri1))` devrait afficher :



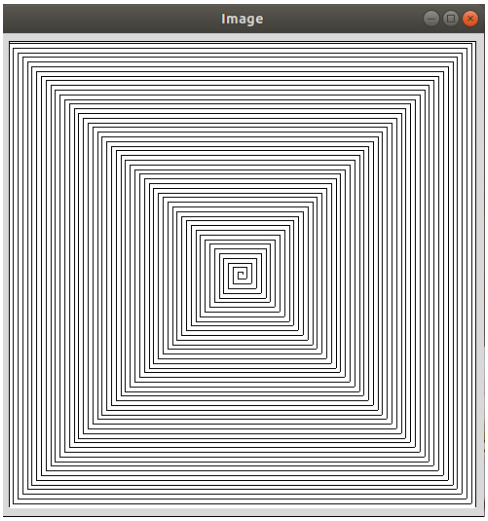
Question 4. Comme premier exemple de courbe générée par un programme, on se propose de tracer une spirale. Pour se faciliter la tache, on commence par écrire une fonction `deplace` qui prend un point du plan `p`, une direction (une chaîne de caractères `"H"`, `"B"`, `"G"`, `"D"` - pour Haut, Bas, Gauche, Droite) et une distance `di` et qui renvoie le point obtenu en déplacant le point `p` dans la direction `d` de la distance `di`.

```
assert deplace(0, "G", 1) == (-1.0, 0.0)
assert deplace(0, "H", 0.5) == (0.0, 0.5)
```

Question 5. En déduire une fonction `spirale` qui prend en entrée un point d'origine `ori`, un décalage `dec` et un nombre d'étapes `n` et qui renvoie l'image de la spirale construite ainsi :

- on part du point `ori`
- à chaque étape, on se déplace dans une nouvelle direction, en commençant par Droite, puis Haut, puis Gauche, puis Bas,
- à l'étape n, le déplacement se fait d'une longueur n.`dec`

Ainsi, `show_image(image_courbe(spirale(0, 0.01, 400)))` devrait afficher :



2 Suggestion : Flocon de Koch

Le flocon de Koch est une figure obtenue itérativement ainsi :

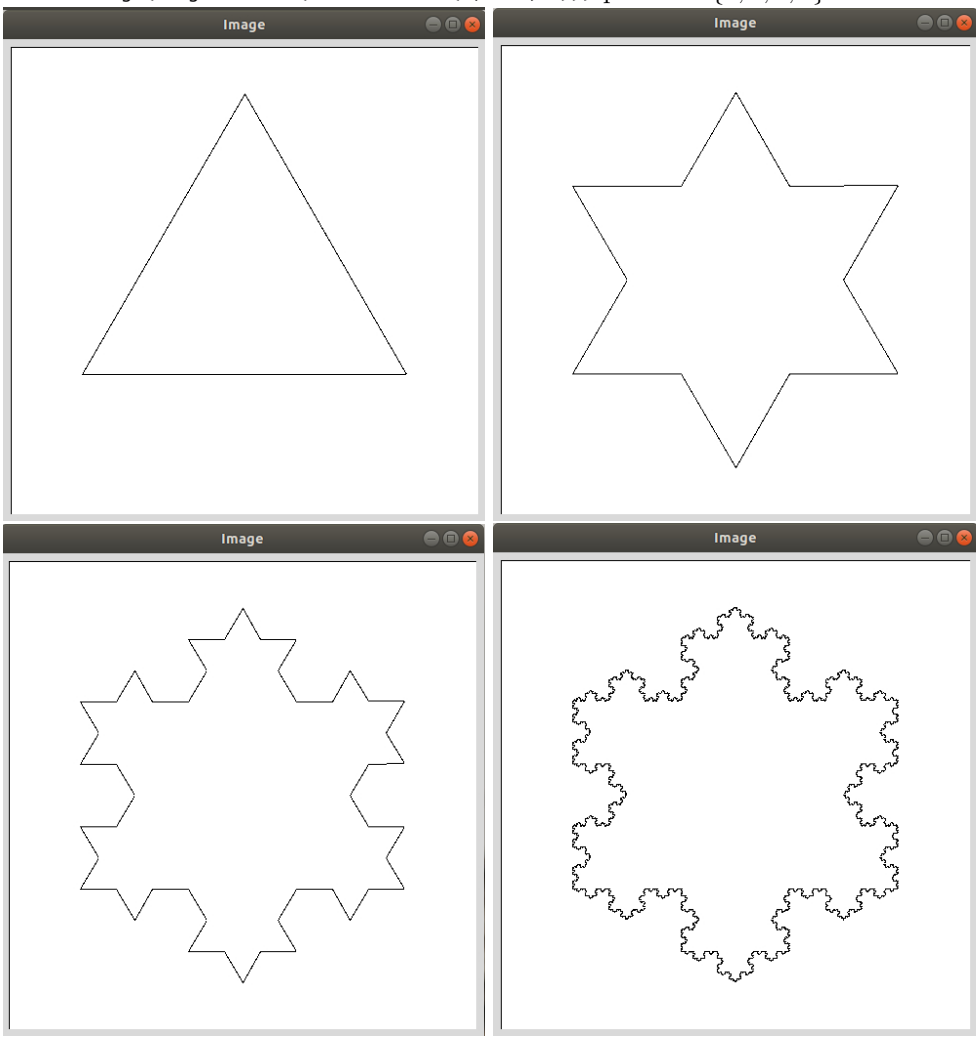
- On commence par dessiner un triangle (étape 0).

- A l'étape $n+1$, chaque segment s de la figure à l'étape n est remplacé par 4 segments:
  - le premier tiers de s
  - les deux autres côtés du triangle équilatéral dont un des côtés est le deuxième tiers s, tourné la pointe vers l'extérieur,
  - le dernier tiers de s

On pourra consulter l'article *Wikipedia* afférent : `https://fr.wikipedia.org/wiki/Flocon_de_Koch`

Ecrire une fonction `courbe_flocon` qui prend un point `p`, un rayon `r` et un nombre d'étapes `n` et qui renvoie l'image correspondant au flocon de Koch à l'étape `n` dont le triangle à l'étape 0 est de centre `p` et de "rayon" `r`.

Par exemple `show_image(image_circuit(courbe_flocon(0, 0.8, i)))` pour `i` $\in \{0, 1, 2, 6\}$ devrait afficher :



3 Suggestion : Courbe du Dragon

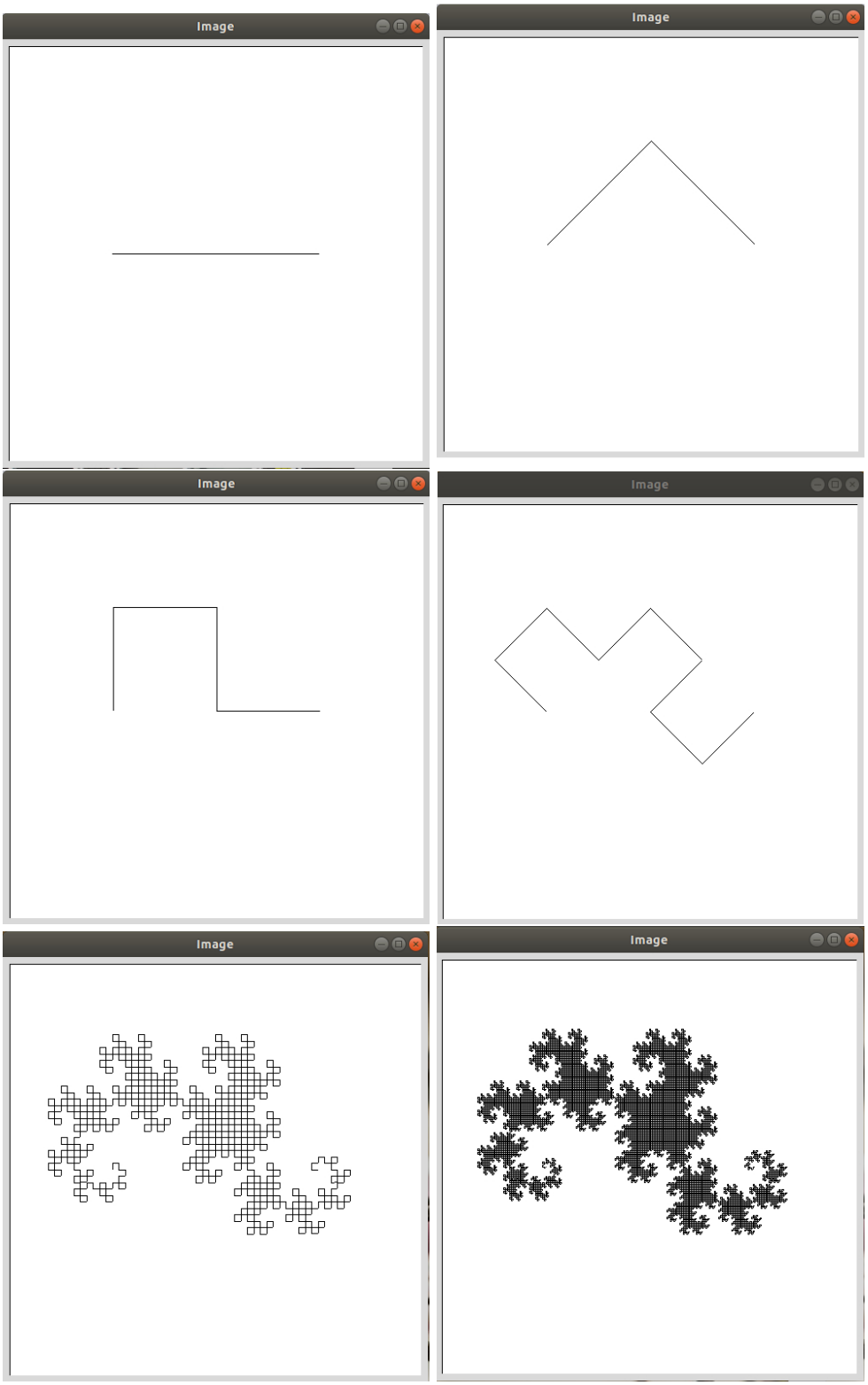
La courbe du Dragon est une figure obtenue itérativement ainsi :

- On commence par dessiner un segment (étape 0).
- A l'étape $n+1$, chaque segment s de la figure à l'étape n est remplacé par deux segments correspondants aux deux petits côtés d'un triangle isocèle rectangle d'hypothénuse s, dont la pointe est tournée, alternativement, vers la gauche puis vers la droite (en commençant par la gauche pour le premier segment de la figure à l'étape n).

On pourra consulter l'article *Wikipedia* afférent : `https://fr.wikipedia.org/wiki/Courbe_du_dragon`

Ecrire une fonction `courbe_dragon` qui prend en entrée deux points correspondant aux extrémités du segment à l'étape 0 et un nombre d'étapes et qui dessine la courbe à l'étape n.

Par exemple `show_image(image_courbe(courbe_dragon((−0.5,0), (0.5,0), i)))` pour `i` $\in \{0, 1, 2, 3, 10, 14\}$ devrait afficher :



4 Suggestion : Triangle de Sierpinski

Le triangle de Sierpenski est une figure obtenue itérativement ainsi :

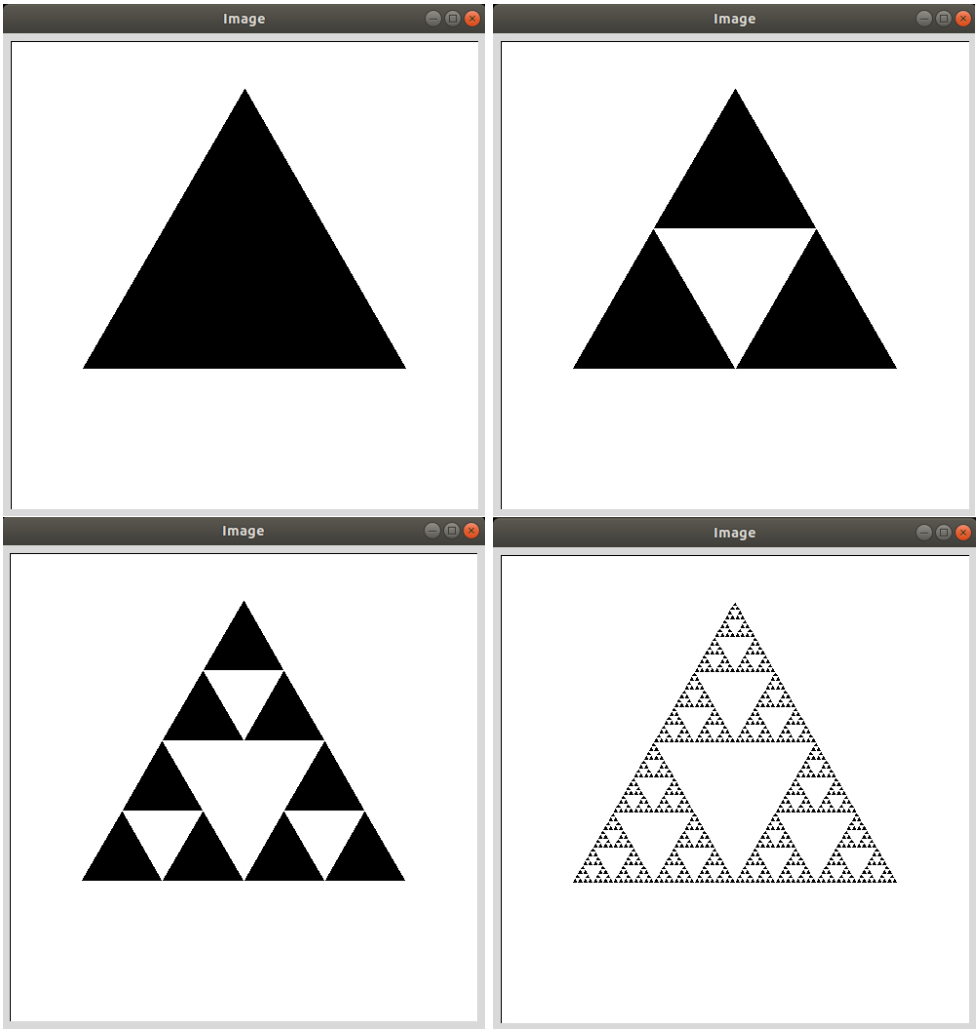
- On commence par dessiner un triangle plein (étape 0).
- A l'étape $n+1$, chaque triangle plein t de la figure à l'étape n est remplacé par 3 triangles pleins, obtenu en évidant à l'intérieur de t, le triangle ayant pour sommets les milieux des côtés de t.

On pourra consulter l'article *Wikipedia* afférent :

`https://fr.wikipedia.org/wiki/Triangle_de_Sierpi%C5%84ski`

Ecrire une fonction `triangle_sierpinski` qui prend en entrée un point `p`, une distance `d` et une étape `n`, et qui dessine le triangle de Sierpinski à l'étape n, en partant à l'étape 0 d'un triangle équilatéral pointe en haut centré en `p` et de rayon `d`.

Par exemple, si on dispose d'une fonction `triangle_remplis` qui dessine les triangles d'une liste de courbes triangles, `show_image(triangles_remplis(triangle_sierpinski(0, 0.8, i)))` pour `i` $\in \{0, 1, 2, 6\}$ devrait afficher :



5 Suggestion : Ensemble de Mandelbrot

L'*ensemble de Mandelbrot* est constitué des points (x, y) du plan tels que la suite complexe définie par

$$\begin{cases} z_0 & = & 0 \\ z_{n+1} & = & z_n^2 + (x + \mathbf{i}y) \end{cases}$$

converge.

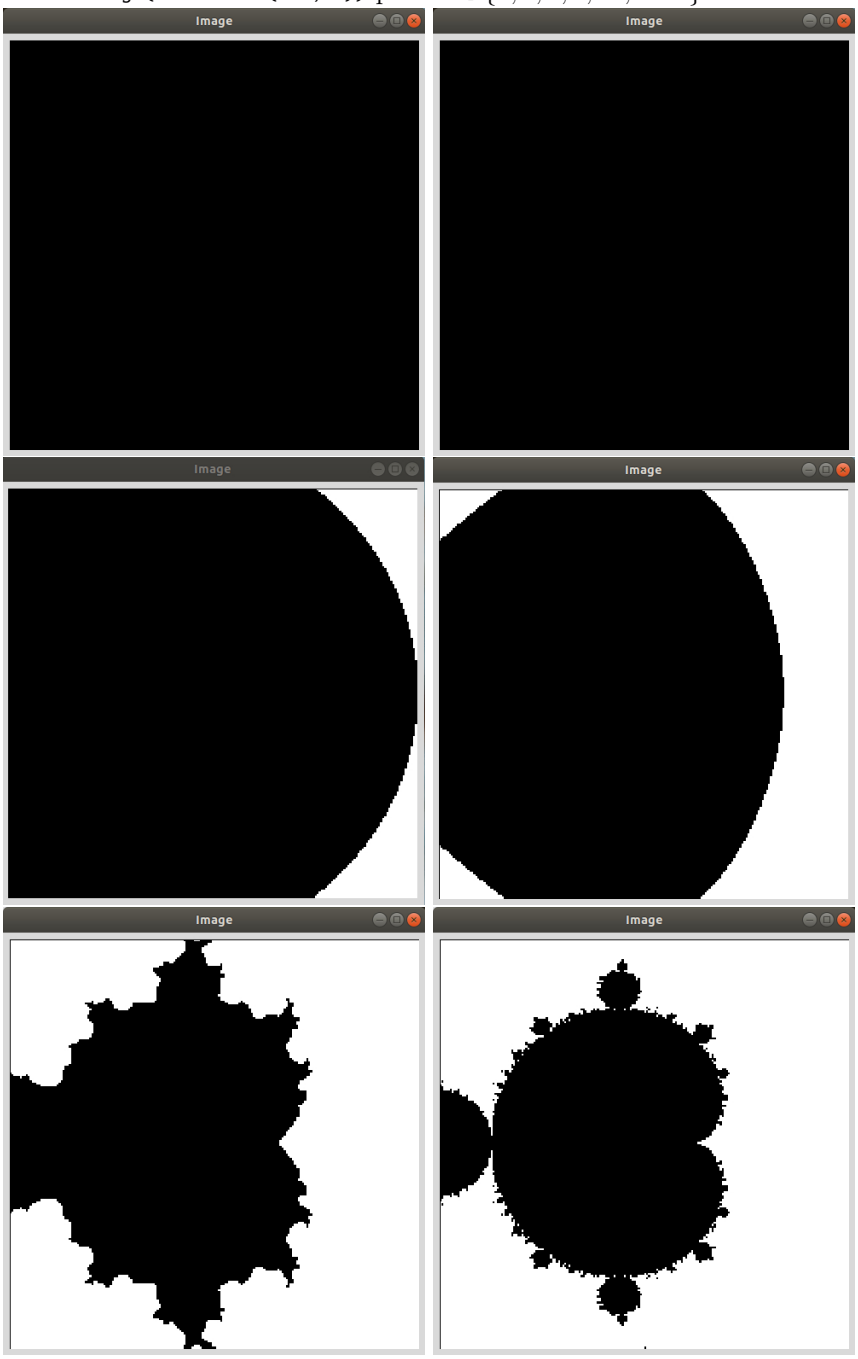
L'*approximation à l'étape n de l'ensemble de Mandelbrot* est constitué des points (x, y) du plan tels que les $n+1$ premiers termes de la suite complexe définie plus haut sont de module inférieur à 2.

On pourra consulter l'article *Wikipedia* afférent :

`https://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot`

Ecrire une fonction `mandelbrot` qui prend en entrée un entier `g` et un entier `n` et qui dessine l'approximation à l'étape n de l'ensemble de Mandelbrot pour g^2 points (x, y) répartis uniformément entre $[-1, -1]$ et $[1, 1]$, chacun représenté par un carré de côté $\frac{2}{g}$ dont le point en bas à gauche a pour coordonnées (x, y).

Par exemple `show_image(mandelbrot(400, 0))` pour `i` $\in \{0, 1, 2, 3, 10, 1000\}$ devrait afficher :



6 Suggestion : Autres fractales

On pourra représenter d'autres courbes générées automatiquement, notamment d'autre figures fractales, en cherchant des exemples sur le Web.