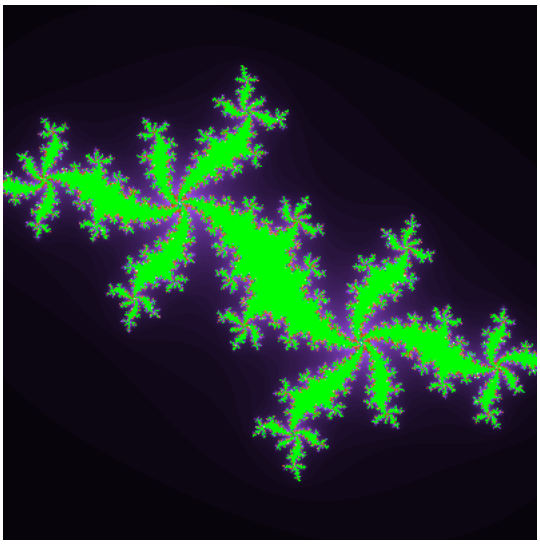
Programmer des fractales avec Python (2/2)

1 Ensemble de Julia

1.1 Définition



Ensemble de Julia pour $c = (0,5\,;-0,6)$

Un ensemble de Julia J_c est une fractale qui est paramétrée par un couple de réels $c = (a\,;b)$. Pour déterminer si un point $M\left(x\,;y\right)$ du plan, on définit une suite de couples de nombres $\left(x_n;y_n\right)$ par $\left(x_0;y_0\right) = \left(x\,;y\right)$ et par la relation de récurrence :

$$\begin{cases} x_{n+1} = x_n^2 - y_n^2 + a \\ y_{n+1} = 2x_n \times y_n + b \end{cases} \tag{1}$$

Par exemple si $\left(x_0;y_0\right) = (3\,;4)$ et si $(a\,;b) = (2\,;1)$ alors :

- $x_1 = 3^2 - 4^2 + 2 = -5$ et $y_1 = 2x_0 \times y_0 + 1 = 2 \times 3 \times 2 + 1 = 13$
- $x_2 = (-5)^2 - 13^2 + 2 = -142$ et $y_1 = 2x_1 \times y_1 + 1 = 2 \times (-5) \times 13 + 1 = -129$

Le point $M\left(x\,;y\right)$ n'appartient pas à l'ensemble de Julia J_c si le nombre $x_n^2 + y_n^2$ (le carré de la distance du point de coordonnées $\left(x_n;y_n\right)$ à l'origine) tend vers l'infini lorsque n tend vers l'infini. Une propriété permet d'affirmer que si $x_n^2 + y_n^2$ dépasse 2^2 à partir d'un certain n alors la suite $x_n^2 + y_n^2$ diverge vers l'infini [1] et le point initial $M\left(x\,;y\right)$ n'appartient pas à J_c.
Pour en savoir plus sur les ensembles de Julia on pourra consulter :

- la page web `https://fr.wikipedia.org/wiki/Ensemble_de_Julia`
- la rubrique ensemble de Julia du didacticiel du logiciel Xaos

En général, on définit un ensemble de Julia J_c à l'aide des **nombres complexes**. Si on a un point du plan $M\left(x\,;y\right)$ on lui fait correspondre le nombre complexe $m = x + \mathrm{i}y$ où i est un nombre imaginaire tel que $\mathrm{i}^2 = -1$. Le nombre complexe m est l'affixe du point M, x est sa **partie réelle** et y sa **partie imaginaire**. Par exemple $A\,(3\,;4)$ a pour affixe $a = 3 + \mathrm{i}4$ de partie réelle 3 et de partie imaginaire 4. Les règles de calculs avec les nombres complexes sont les mêmes que celles avec les nombres réels en rajoutant la règle $\mathrm{i}^2 = -1$.
Avec les nombres complexes, l'ensemble de Julia de paramètre $(a\,;b)$ avec $c = a + \mathrm{i}b$ se détermine en tout point $M\left(x\,;y\right)$ d'affixe $m = x + \mathrm{i}y$ par l'étude de la suite de Julia définie par :

$$\begin{cases} z_0 = x + \mathrm{i}y \\ z_{n+1} = z_n^2 + c \end{cases} \text{ où } z_n = x_n + \mathrm{i}y_n \tag{2}$$

1. Si $a^2 + b^2 \leqslant 4$

L'ensemble de Julia est l'ensemble des points $M(x;y)$ tels que $x_n^2+y_n^2$ ne tend pas vers l'infini lorsque n tend vers l'infini.
Pour tracer un ensemble de Julia J_c avec l'ordinateur, on va dans l'ordre :

1. Choisir une zone du plan ($x_{min}=-1.25$; $x_{max}=1.25$; $y_{min}=-1.25$; $y_{max}=1.25$ en général) où il se passe des choses intéressantes ...

   Choisir une fenêtre graphique de l'écran comportant un nombre fini de pixels (par exemple 400 pixels de largeur fois 400 pixels de hauteur). L'image affichée est constituée d'un tableau de pixels caractérisé par ses dimensions (largeur x hauteur), chaque pixel apparaissant d'une certaine couleur. De plus on doit associer chaque pixel à un point de la fenêtre du plan (soit $400\times400=1\,600$ par exemple). Un pixel étant repéré par un couple `(ligne;colonne)` on devra donc définir une fonction qui à un couple `(ligne;colonne)` associe le couple de coordonnées $(x;y)$ du point du plan représenté par le pixel.

2. Parcourir avec deux boucles **for** imbriquées (parcours des colonnes puis des lignes de haut en bas et de gauche à droite en général),les pixels de notre fenêtre graphique et calculer pour chacun, les coordonnées $(x;y)$ du point du plan associé.

3. Calculer dans une boucle `while`, les termes de la suite $(x_n;y_n)$. Pour ne pas rentrer dans une boucle infinie, il nous faudra un compteur d'itérations et un teste d'arrêt : la boucle tournera tant que le test d'appartenance à l'ensemble de Julia J_c est réalisé ($x_n^2+y_n^2<=2^2$) et que le compteur d'itérations sera inférieur à un nombre d'itérations maximum.

4. Choisir une couleur pour le pixel `(ligne;colonne)` selon la valeur du compteur d'itérations en sortie de la boucle **while**.

   - Si le nombre d'itérations est supérieur au seuil maximal et si le test d'appartenance à J_c du dernier terme de la suite est vérifié, on considère que le pixel appartient à J_c, on le colorie alors en noir.
   - Sinon le pixel n'appartient pas à J_c n'a pas été vérifié et on colorie le pixel d'une couleur qui dépend du nombre d'itérations effectuées.

1.2 Programme de tracé en Python

Nous allons écrire un programme de tracé d'ensemble de Julia avec Python (version 2).
Ouvrir l'environnement de développement pour Python 2 et créer un nouveau fichier `julia.py` dans l'éditeur de texte.
Nous allons dérouler les étapes du programme dans l'ordre décrit ci-dessus.

1. On commence par le code suivant :

```
1  #pour que la division en python2 se comporte comme en python3
2  from __future__ import division
3  from PIL import Image
4
5  a = 0.39
6  b = 0.6
7  taille = 400
8  xmin = -1.25
9  xmax = 1.25
10 ymin = -1.25
11 ymax = 1.25
12 iterationmax = 200
13 im = Image.new('RGB',(taille,taille),(255,255,255))
```

```
14 pixels = im.load()
```

- En ligne 2 on paramètre la fonctionnement de la division et en ligne 3 on importe le module Image de la bibliothèque PIL de qui contient des fonction spécifiques de traitement d'image.
- En lignes 5 et 6 on initialise les variables a et b du paramètre $c = (a\,;\,b)$ de l'ensemble de Julia.
- En ligne 7 on initialise la taille de la fenêtre graphique, valeur commune de sa largeur et de sa hauteur.
- Dans les lignes 8 à 11 on initialise les valeurs délimitant la zone du plan et en ligne 12 on définit le nombre maximal d'itérations
- Ensuite, on utilise les fonctions spécifiques de la bibliothèque PIL, pour créer en ligne 13 une image vide de dimensions 400x400 dont tous les pixels sont en blanc, puis pour charger en ligne 14 le tableau de pixels de cette image dans une variable `pixels`.

  On utilise ici le codage (Rouge,Vert,Bleu) des couleurs. Chaque couleur est codée par trois entiers compris entre 0 et 255.

  Le rouge est codé (255,0,0), le vert (0,255,0), le bleu (0,0,255), le blanc (255,255,255) et le noir (0,0,0).

  Pour en savoir plus on pourra consulter la page web : `http://fr.wikipedia.org/wiki/Rouge_vert_bleu`

2. Suite du code (attention à l'indentation) :

```
15 for line in range(taille):
16     for col in range(taille):
17         i = 1
18         x = xmin+col*(xmax-xmin)/taille
19         y = ymax-line*(ymax-ymin)/taille
20         while i<=iterationmax and (x**2+y**2)<=4:
21             stock = x
22             x = x**2-y**2+a
23             y = 2*stock*y+b
24             i += 1
25         if i>iterationmax and (x**2+y**2)<=4:
26             pixels[col,line] = (0,0,0)
27         else:
28             pixels[col,line] = ((4*i)%256,2*i,(6*i)%256)
```

a. En lignes 15 et 16 on amorce nos deux boucles **for** imbriquées pour parcourir les lignes et les colonnes du tableau de pixels de notre image. Les lignes 17 à 28 seront exécutées pour chaque pixel auquel on accède par `pixels[colonne,ligne]`.

b. En ligne 17 on initialise notre compteur d'itération `i` à 1.

c. En ligne 18 et 19 on associe au pixel `(ligne,colonne)` le point du plan $(x\,;\,y)$ dont on souhaite déterminer s'il appartient à l'ensemble de Julia J_c.

Les pixels sont repérés à partir d'un pixel origine $(0\,;0)$ situé dans le coin supérieur gauche de l'image, l'axe des abscisses étant orienté vers la droite et l'axe des ordonnées vers le bas.

On en déduit donc la correspondance suivante entre pixels et points :

Pixel	(0 ;0)	(taille ;0)	(0 ;taille)	(taille ;taille)
Point du plan	$(x_{min}\,;y_{max})$	$(x_{max}\,;y_{max})$	$(x_{min}\,;y_{min})$	$(x_{min}\,;y_{min})$

Sachant que l'abscisse x et l'ordonnée y du point sont respectivement des fonctions affines de `colonne` et `ligne`, expliquer les formules des lignes 18 et 19.

d. En ligne 20 on amorce la boucle `while` qui calcule les termes successifs de la suite de Julia définie par $(x_0; y_0) = (x; y)$ et $x_{n+1} = x_n^2 - y_n^2 + a$ et $y_{n+1} = 2x_n \times y_n + b$.

- Dans quels cas sort-on de la boucle ?
- Expliquer le code des lignes 21 à 24. A quoi sert la variable `stock` définie en ligne 21 ?

e. En lignes 25 et 26, si le nombre d'itérations est supérieur au seuil maximum et si le test d'appartenance à J_c du dernier terme de la suite est vérifié, le pixel est dans J_c et on le colorie en noir.
Comment modifier la ligne 26 pour colorier le pixel en vert ? en rouge ? en jaune ?

f. Si le test de la ligne 25 n'est pas vérifié,le pixel n'est pas dans J_c et on paramètre ses composantes (Rouge,Vert,Bleu) en fonction de la valeur `i` du compteur d'itérations (L'opérateur modulo % retourne le reste dans la division euclidienne).

g. Dernière partie du programme (indentation nulle) :

```
29 im.save('julia.png')
30 im.show()
```

On sauvegarde l'image en ligne 29 puis on l'affiche à l'écran en ligne 30.

h. Tester ce programme pour représenter plusieurs ensembles de Julia selon les valeurs du couple de paramètres $(a; b)$:
$(a; b) = (-0.5; 0.6)$, $(a; b) = (0.285; 0.013)$, $(a; b) = (0; -0.8)$

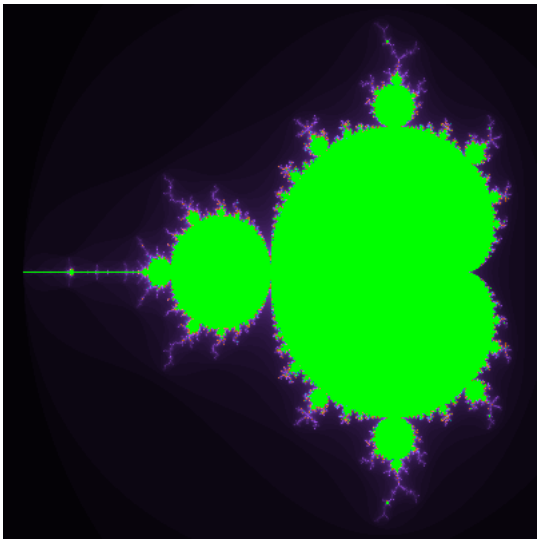
2 Ensemble de Mandelbrot

2.1 Définition

Un point de coordonnées $c = (a; b)$ du plan appartient à l'ensemble de Mandelbrot s'il appartient à l'ensemble de Julia J_c c'est-à-dire si la suite définie par $(x_0; y_0) = (a; b)$ et $x_{n+1} = x_n^2 - y_n^2 + a$ et $y_{n+1} = 2x_n \times y_n + b$ est telle que $x_n^2 + y_n^2$ ne tend pas vers l'infini lorsque n tend vers l'infini.
On peut explorer la fractale de Mandelbrot avec le logiciel Xaos.
Pour des compléments on pourra consulter :

- la page web `http://images.math.cnrs.fr/L-ensemble-de-Mandelbrot.html`
- la page web `http://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot`
- la rubrique ensemble de Mandelbrot du logiciel Xaos

L'ensemble de Mandelbrot est contenu dans la zone du plan délimitée par :
$x_{min} = -2$; $x_{max} = 0.5$; $y_{min} = -1.2$; $y_{max} = 1.2$



Ensemble de Mandelbrot

2.2 Programmation

En modifiant le programme `julia.py`, écrire un programme Python `mandelbrot.py` qui trace l'ensemble de Mandelbrot.