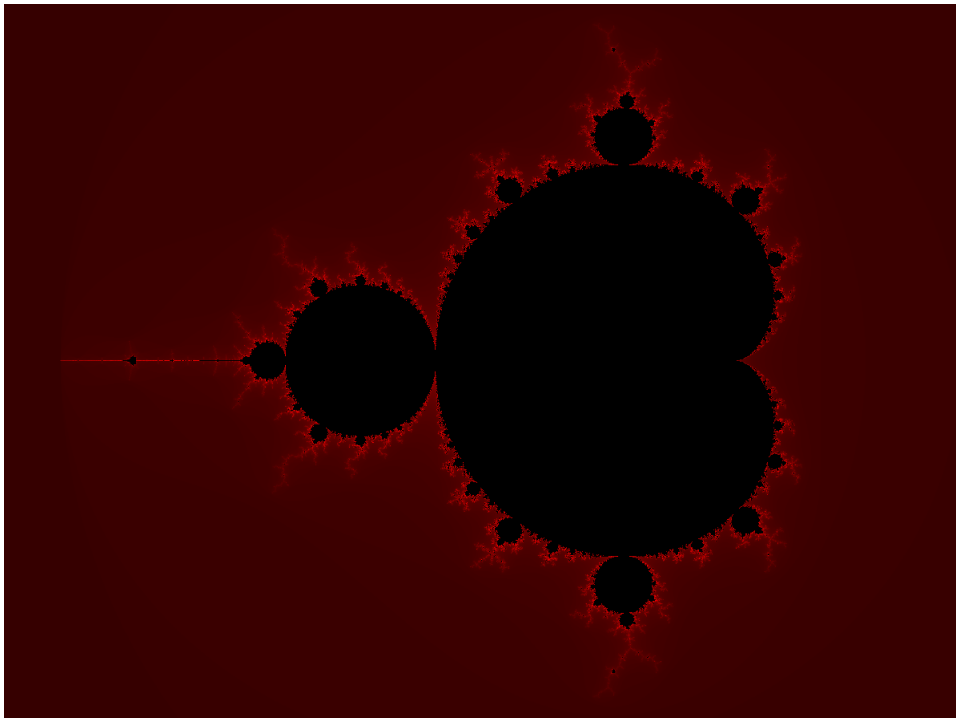
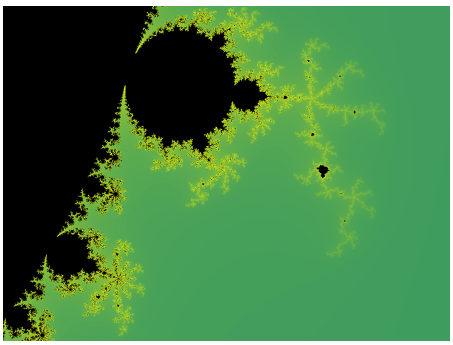
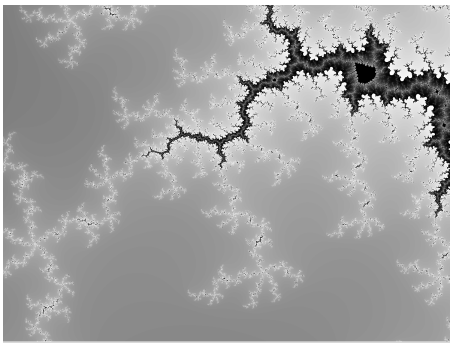
Ensemble de Mandelbrot

*Tu vas découvrir un univers encore plus passionnant qu'*Harry Potter *: l'ensemble de Mandelbrot. C'est une fractale, c'est-à-dire que lorsque l'on zoome sur certaines parties de l'ensemble, on retrouve une image similaire à celle de départ. On découvrira aussi les ensembles de Julia.*



Voici un zoom, puis un zoom du zoom ! Les points noirs forment l'ensemble de Mandelbrot.





Cette activité est proposée en deux versions : une avec les nombres complexes (pour ceux qui connaissent) et une version avec les nombres réels.

Cours 1 (L'ensemble de Mandelbrot (avec les nombres complexes))**.**

Voici la définition de l'ensemble de Mandelbrot pour ceux d'entre vous qui connaissent les nombres complexes.

On fixe un nombre complexe c. On définit une suite de nombres complexes par récurrence :

$$z_0 = 0 \qquad \text{et pour } n \geqslant 0 \quad z_{n+1} = z_n^2 + c.$$

Si $|z_n|$ ne tend pas vers l'infini (lorsque $n \to +\infty$) alors c est par définition un point de l'ensemble de Mandelbrot $\mathcal{M}$.
Autrement dit, l'ensemble de Mandelbrot $\mathcal{M}$ est formé de toutes les valeurs $c \in \mathbb{C}$ telles que la suite récurrente (z_n) (qui dépend de c) reste bornée.

Activité 1 (Mandelbrot (version complexe))**.**

Objectifs : préparer le calcul de l'ensemble de Mandelbrot, en utilisant directement les nombres complexes.

1. Programme une fonction (toute simple) `f(z,c)` qui pour z et c, des nombres complexes donnés, renvoie $z^2 + c$.
2. Programme une fonction `iterer(c)` qui pour un nombre complexe c donné, renvoie le nombre d'itérations qu'il a fallu pour que la suite (z_n) s'échappe « à l'infini » ; si au bout d'un certain nombre d'itérations la suite ne s'échappe pas, la fonction renvoie 0.

   On peut prouver que (z_n) s'échappe vers l'infini dès que l'on trouve $i \geqslant 1$ tel que $|z_i| > 2$. On stoppe alors les calculs et on renvoie l'indice i. Si cela n'arrive pas au bout des 100 premiers termes par exemple, on considère que la suite ne s'échappera jamais et on arrête les calculs (on renvoie 0), et c est un point de l'ensemble de Mandelbrot.

   Voici les détails afin de programmer cette fonction.

   Algorithme.
   - — Entrée : un nombre complexe c.
     — Sortie : le premier indice $i \geqslant 1$ avec $|z_i| > 2$, ou 0 si cela n'arrive pas pour les MaxIter premiers termes.
   - Définir la constante MaxIter, par exemple MaxIter $= 100$.
   - Poser $z = 0$ (correspondant à z_0).
   - Poser $i = 1$.
   - Tant que $|z| \leqslant 2$ et $i <$ MaxIter :
     — Faire $z \leftarrow f(z, c)$.
     — Faire $i \leftarrow i + 1$.
   - Une fois la boucle terminée, si $i =$ MaxIter alors renvoyer 0, sinon renvoyer i.

Tu peux passer directement à l'activité 3 pour afficher l'ensemble de Mandelbrot.

Cours 2 (L'ensemble de Mandelbrot (avec les nombres réels))**.**

Voici la définition de l'ensemble de Mandelbrot pour ceux qui ne connaissent pas encore les nombres complexes.

On fixe un point du plan (a, b). On définit deux suites réelles par récurrence :

$$x_0 = 0 \qquad \text{et} \qquad y_0 = 0$$

et pour $n \geqslant 0$:

$$x_{n+1} = x_n^2 - y_n^2 + a \qquad \text{et} \qquad y_{n+1} = 2x_n y_n + b.$$

Si la suite des points (x_n, y_n) ne tend pas vers l'infini (lorsque $n \to +\infty$) alors (a, b) est par définition un point de l'ensemble de Mandelbrot $\mathcal{M}$.
Autrement dit, l'ensemble de Mandelbrot $\mathcal{M}$ est formé de tous les points $(a, b) \in \mathbb{R}^2$ tels que la suite des points (x_n, y_n) (qui dépend de (a, b)) reste bornée.

Activité 2 (Mandelbrot (version réelle))**.**

Objectifs : préparer le calcul de l'ensemble de Mandelbrot, en utilisant uniquement des nombres réels (les nombres complexes restent cachés).

1. Programme une fonction `f(x,y,a,b)` qui pour x, y, a, b, des nombres réels, calcule
$$x' = x^2 - y^2 + a \qquad \text{et} \qquad y' = 2xy + b$$
et renvoie les deux nombres réels x' et y'.
2. Programme une fonction `iterer(a,b)` qui pour un couple de réel (a, b) donné, renvoie le nombre d'itérations qu'il a fallu pour que la suite (x_n, y_n) s'échappe « à l'infini » ; si au bout d'un certain nombre d'itérations la suite ne s'échappe pas, la fonction renvoie 0.
On peut prouver que (x_n, y_n) s'échappe vers l'infini dès que l'on trouve $i \geqslant 1$ tel que $x_i^2 + y_i^2 > 4$. On stoppe alors les calculs et on renvoie l'indice i. Si cela n'arrive pas au bout des 100 premiers termes par exemple, on considère que la suite ne s'échappera jamais et on arrête les calculs (on renvoie 0), et (a, b) est un point de l'ensemble de Mandelbrot.
Voici les détails afin de programmer cette fonction.

Algorithme.
- — Entrée : un couple de nombres réels (a, b).
  — Sortie : le premier indice $i \geqslant 1$ avec $x_i^2 + y_i^2 > 4$, ou 0 si cela n'arrive pas pour les MaxIter premiers termes.
- Définir la constante MaxIter, par exemple MaxIter $= 100$.
- Poser $x = 0$ et $y = 0$ (correspondant à $(x_0, y_0) = (0, 0)$).
- Poser $i = 1$.
- Tant que $x^2 + y^2 \leqslant 4$ et $i <$ MaxIter :
  — Faire $x, y \leftarrow f(x, y, a, b)$.
  — Faire $i \leftarrow i + 1$.
- Une fois la boucle terminée, si $i =$ MaxIter alors renvoyer 0, sinon renvoyer i.

Dans les activités suivantes tu vas afficher l'ensemble de Mandelbrot.

Activité 3 (Préparer l'affichage de l'ensemble de Mandelbrot)**.**

Objectifs : se préparer à afficher l'ensemble de Mandelbrot. Le tout en couleur !

On utilise le module `tkinter` pour afficher la fractale.
Le code principal sera le suivant (la fonction `mandelbrot()` sera définie dans l'activité suivante) :

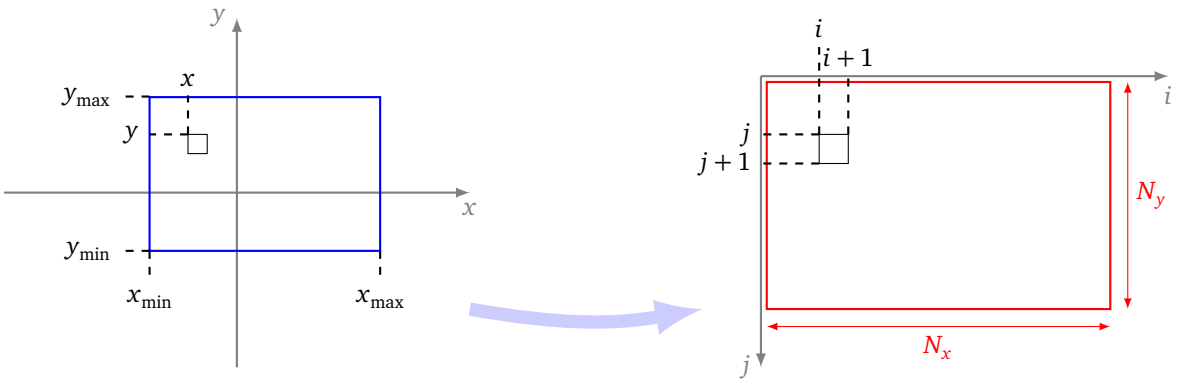
```
from tkinter import *
root = Tk()
canvas = Canvas(root, width=Nx, height=Ny, background="white")
canvas.pack(side=LEFT, padx=5, pady=5)

mandelbrot()

root.mainloop()
```

1. Programme une fonction `afficher_pixel(i,j,couleur)` qui affiche un pixel coloré en position (i, j) d'une fenêtre graphique (qui sera définie après).
   Tu peux par exemple tracer un segment à l'aide de `create_line()` de (i, j) à $(i+1, j)$.
2. Programme une fonction `choix_couleur(i)` qui renvoie une couleur en fonction du nombre i (qui correspondra au nombre d'itérations).

```
    if i == 0:
        R,V,B = 0,0,0
    else:
        R,V,B = 50 + 2*i,0,0
    couleur = '#%02x%02x%02x' % (R, V, B)
```

3. Définis des variables `xmin`, `xmax`, `ymin`, `ymax` qui correspondent à la zone de l'ensemble de Mandelbrot à visualiser. Par exemple pour avoir tout l'ensemble de Mandelbrot on fait :

```
xmin, xmax = -2.2, 1
ymin, ymax = -1.2, 1.2
```



Plan des coordonnées (x, y) **Écran des coordonnées** (i, j)

Le plan de coordonnées (x, y) est représenté sur la figure de gauche ; le grand rectangle correspond à la zone à afficher ; le petit carré est la zone qui sera représentée pas un pixel. Sur la figure de droite c'est l'écran ; les coordonnées sont données par des coordonnées entières (i, j) ; l'écran est composé de pixels (un seul est dessiné), chacun de largeur 1.

4. Définis deux variables `Nx` et `Ny` qui correspondent à la taille de la fenêtre graphique à afficher. Par exemple pour une fenêtre de largeur `Nx`= 400 pixels et de hauteur correspondant à la zone à visualiser, faire :

```
Nx = 400
Ny = round( (ymin-ymax)/(xmin-xmax) * Nx )
```

Activité 4 (Tracer l'ensemble de Mandelbrot)**.**

Objectifs : dessiner l'ensemble du Mandelbrot ou un zoom.

- Tu as une zone rectangulaire définie par `xmin`, `xmax`, `ymin`, `ymax` qui correspond à la zone de l'ensemble de Mandelbrot à visualiser.
- Cette zone s'affichera dans une fenêtre de largeur `Nx` pixels et de hauteur `Ny` pixels.
- Définis d'abord deux variables `pasx`, `pasy` qui correspondent à la taille couverte par un pixel selon les formules :

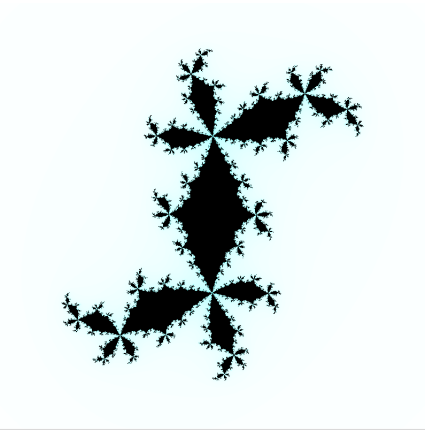
```
pasx = (xmax-xmin)/Nx        pasy = (ymax-ymin)/Ny
```

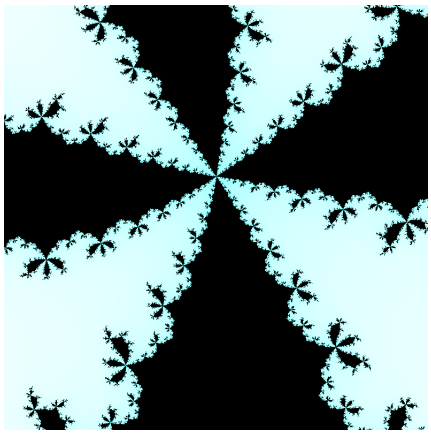
Programme une fonction `mandelbrot()` qui trace la zone désirée de l'ensemble de Mandelbrot selon les instructions suivantes :

- Partir de `a = xmin` et `b = ymin`.
- Pour `i` variant de `0` à `Nx` :
  - Pour `j` variant de `0` à `Ny` :
    - ⋆ (Version complexe) Poser `c = a + b*1j` (c'est-à-dire $c = a + ib \in \mathbb{C}$) et calculer `vitesse = iterer(c)`.
    - ⋆ (Version réelle) Calculer `vitesse = iterer(a,b)`.
    - ⋆ Choisir une couleur correspondant à la vitesse d'échappement `couleur = choix_couleur(vitesse)`.
    - ⋆ Allumer le pixel correspondant par `afficher_pixel(i,j,couleur)`.
    - ⋆ Faire `b = b + pasy`.
  - Faire `b = ymin`.
  - Faire `a = a + pasx`.

Le jeu est maintenant double : trouver les plus jolies couleurs possibles (en modifiant la fonction `choix_couleur()`*) et surtout chercher des zooms avec les plus belles formes possibles.*

Les ensembles de Julia se construisent par une méthode proche de celle utilisée pour l'ensemble de Mandelbrot. Cependant les images obtenues sont très différentes et en plus il y a une infinité d'ensembles de Julia possibles.





Ci-dessus l'ensemble de Julia avec $c = 0.3 + 0.55i$ ($a = 0.3$, $b = 0.55$), appelé le « lapin de Douady », avec à droite un zoom. Ci-dessous l'ensemble de Julia pour $c = -1.31$ (à gauche) et $c = -0.101 + 0.956i$ (à droite).

Cours 3 (Les ensembles de Julia (avec les nombres complexes))**.**

Voici la définition des ensembles de Julia pour ceux d'entre vous qui connaissent les nombres complexes.

On fixe un nombre complexe c. On va définir l'ensemble de Julia $\mathcal{J}(c)$ associé à cette valeur. Pour chaque $z_0 \in \mathbb{C}$ on définit une suite de nombres complexes par récurrence pour $n \geqslant 0$:

$$z_{n+1} = z_n^2 + c$$

Si $|z_n|$ ne tend pas vers l'infini (lorsque $n \to +\infty$) alors z_0 est par définition un point de l'ensemble de Julia $\mathcal{J}(c)$. Cette fois l'ensemble de Julia $\mathcal{J}(c)$ est formé de toutes les valeurs $z_0 \in \mathbb{C}$ telles que la suite récurrente (z_n) (qui dépend de z_0) reste bornée.
La différence avec la définition de l'ensemble de Mandelbrot est qu'ici, le terme c est fixé pour chaque fractale, et c'est le terme initial z_0 que l'on fait varier.

Activité 5 (Julia (version complexe))**.**

Objectifs : dessiner des ensembles de Julia, en utilisant les nombres complexes.

Il s'agit d'adapter ton programme qui a servi à dessiner l'ensemble de Mandelbrot.

- Conserve ta fonction `f(z,c)` qui pour z et c, des nombres complexes donnés, renvoie $z^2 + c$.
- Adapte la fonction `iterer()` en une fonction `iterer(z0,c)` pour tenir compte du terme initial z_0.
- Pour un complexe c fixé, programme une fonction `julia(c)` qui affiche l'ensemble de Julia $\mathcal{J}(c)$. C'est presque comme pour l'ensemble de Mandelbrot, mais cette fois lorsque tu fais varier a, b c'est pour définir $z_0 = a + ib$; c lui reste fixé.

Cours 4 (Les ensembles de Julia (version réelle))**.**

Voici la définition des ensembles de Julia en utilisant seulement les nombres réels.

On fixe un couple de réels (a, b). On va définir l'ensemble de Julia $\mathcal{J}(a, b)$ associé à ce couple. Pour chaque couple $(x_0, y_0) \in \mathbb{R}^2$ on définit une suite de nombres réels par récurrence pour $n \geqslant 0$:

$$x_{n+1} = x_n^2 - y_n^2 + a \qquad \text{et} \qquad y_{n+1} = 2x_n y_n + b$$

Si la suite des points (x_n, y_n) ne tend pas vers l'infini (lorsque $n \to +\infty$) alors (x_0, y_0) est par définition un point de l'ensemble de Julia $\mathcal{J}(a, b)$. Cette fois l'ensemble de Julia $\mathcal{J}(a, b)$ est formé de tous les couples $(x_0, y_0) \in \mathbb{R}^2$ tels que la suite (x_n, y_n) (qui dépend de (x_0, y_0)) reste bornée.
La différence avec la définition de l'ensemble de Mandelbrot est qu'ici, le couple (a, b) est fixé pour chaque fractale, et c'est le couple initial (x_0, y_0) que l'on fait varier.

Activité 6 (Julia (version réelle))**.**

Objectifs : dessiner des ensembles de Julia, en utilisant les nombres réels.

Il s'agit d'adapter ton programme qui a servi à dessiner l'ensemble de Mandelbrot.

- Conserve ta fonction `f(x,y,a,b)` qui pour x, y et a, b, nombres réels donnés, renvoie les deux réels $x' = x^2 - y^2 + a$ et $y' = 2xy + b$.
- Adapte ta fonction `iterer()` en une fonction `iterer(x0,y0,a,b)` pour tenir compte des termes initiaux x_0 et y_0.
- Pour un couple (a, b) fixé, programme une fonction `julia(a,b)` qui affiche l'ensemble de Julia $\mathcal{J}(a, b)$. C'est presque comme pour l'ensemble de Mandelbrot, mais cette fois tu fais varier x_0 et y_0 à chaque pixel de l'écran ; a et b eux restent fixés.

Cours 5 (Pour quelques secondes de moins...)**.**

Les calculs pour tracer l'ensemble de Mandelbrot (et ceux de Julia) sont très longs. Il n'y a pas de solutions immédiates avec `Python` pour aller plus vite. (Les scripts `Python` sont interprétés et sont plus longs à exécuter qu'un programme compilé.)

Voici quelques petites astuces pour aller plus vite.

- Pour calculer x^2 il peut être plus rapide de calculer `x*x` que `x**2`.
- Il est plus rapide de vérifier $x^2 + y^2 > 4$ que $\sqrt{x^2 + y^2} > 2$.
- Enfin il est important de ne pas recalculer plusieurs fois la même expression. Par exemple à chaque itération, on a besoin de calculer

  $$x' = x^2 - y^2 + a \qquad \text{et} \qquad y' = 2xy + b$$

  et de vérifier si

  $$x^2 + y^2 > 4.$$

  Donc on calcule deux fois x^2 et deux fois y^2. Pour éviter cela on peut donc commencer par calculer $x_2 = x^2$, $y_2 = y^2$, puis faire les calculs :

  $$x' = x_2 - y_2 + a, \qquad y' = 2xy + b \quad \text{et} \quad x_2 + y_2 > 4.$$

  On est donc passé du calcul de 5 multiplications (4 carrés et le produit xy) à 3 multiplications (2 carrés et le produit xy).
- Enfin, on peut éviter de calculer des produits de deux nombres différents mais uniquement des carrés (en théorie c'est un tout petit peu plus rapide). Il suffit de remarquer que

  $$2xy = (x + y)^2 - x^2 - y^2$$

  (il n'y a qu'un nouveau carré à calculer car on a déjà calculé x^2 et y^2). Au total on a donc seulement 3 carrés à calculer à chaque itération.