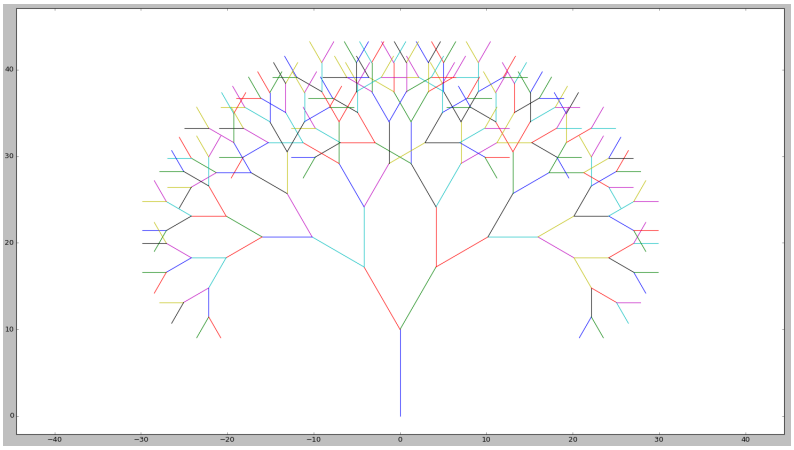
TP : fractales corrigé
Informatique pour tous

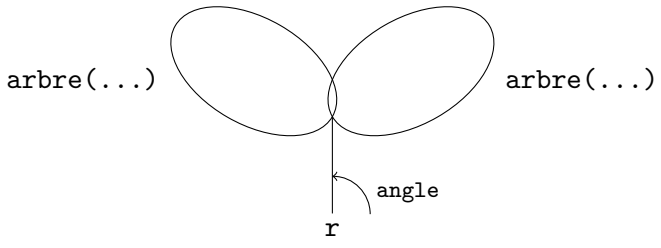
Dans tout le TP, un point sera représenté par un couple (x, y) (x : abscisse, y : ordonnée).

I Arbre récursif

On veut dessiner un arbre récursif, constitué d'un tronc d'où part deux arbres récursifs plus petits :



1. Écrire `import matplotlib.pyplot as plt` et `import numpy as np`.
2. On rappelle que, si `X` et `Y` sont deux listes (des abscisses et des ordonnées, respectivement), `plt.plot(X, Y)` relie le point (`X[0]`, `Y[0]`) à (`X[1]`, `Y[1]`), lui-même relié à (`X[2]`, `Y[2]`)...
   Écrire une fonction `segment` ayant deux points (sous forme de couples) en arguments et dessinant le segment défini par ces deux points. On utilisera seulement `plt.plot` et pas `plt.show` pour l'instant.
   ▶
   ```
   def segment(p1, p2):
       plt.plot([p1[0], p2[0]], [p1[1], p2[1]])
   ```
3. Écrire une fonction `rot` telle que `rot(c, d, angle)` renvoie le point situé à distance `d` du point `c` et faisant un angle `angle` dans le sens trigonométrique. On pourra utiliser `np.cos` et `np.sin`
   ▶
   ```
   def rot(c, d, angle):
       return (c[0] + d * np.cos(angle), c[1] + l * np.sin(angle))
   ```
4. Écrire une fonction récursive `arbre` telle que `arbre(r, d, angle, n)` dessine un arbre dont le tronc commence au point `r`, est de longueur `d` et avec un angle `angle` dans le sens trigonométrique. Cette fonction devra afficher le tronc puis s'appeller récursivement pour afficher deux sous-arbres plus petits, par exemple dont les longueurs sont divisées par `1.2` et les angles augmentés ou diminués de `np.pi/6`. `n` est le nombre d'appels récursifs : on le diminuera à chaque appel récursif et on s'arrêtera lorsque il vaut 0 (c'est le cas de base).
   Tester et afficher avec `plt.show()`.

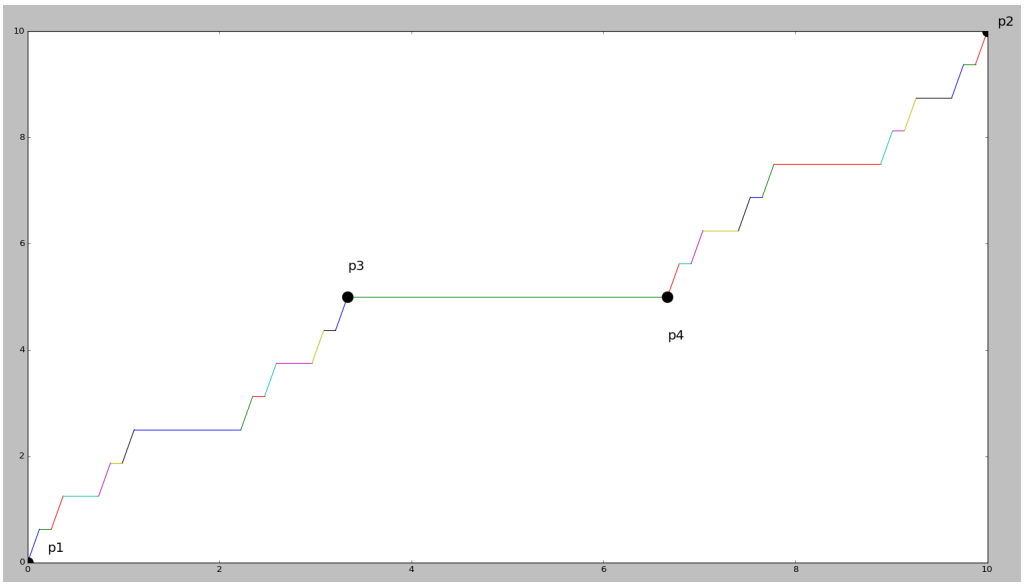


▶ la fonction ne renvoie pas de valeur (en fait, elle renvoie par défaut `None`, c'est à dire rien).

```
def arbre(r, l, angle, n):
    if n != 0:
        next = rot(r, l, angle)
        segment(r, next)
        arbre(next, l/1.2, angle - np.pi/6, n-1)
        arbre(next, l/1.2, angle + np.pi/6, n-1)
```

II Escalier de Cantor

L'escalier de Cantor (ou du diable) est un exemple de fonction continue, dérivable et de dérivée nulle presque partout (en particulier, sur un ensemble dense) et pourtant non constante.



Écrire une fonction escalier telle que `escalier(p1, p2, n)` dessine un escalier de Cantor du point `p1` vers le point `p2`, avec `n` appels récursifs.
Pour cela :

1. Si `n == 0`, on dessine le segment de `p1` à `p2`.
2. Sinon, on dessine un escalier de Cantor de `p1` à `p3`, un segment de `p3` à `p4`, et un escalier de Cantor de `p4` à `p2`. L'ordonnée de `p3` est le milieu des ordonnées de p1 et p2. L'abscisse de `p3` est à 1/3 entre `p1` et `p2`.

▶
```
def escalier(p1, p2, n):
    if n == 0:
        segment(p1, p2)
    else:
        p3 = (2*p1[0] + p2[0])/3, (p2[1] + p1[1])/2
        p4 = (p1[0] + 2*p2[0])/3, (p2[1] + p1[1])/2
        escalier(p1, p3, n-1)
        segment(p3, p4)
        escalier(p4, p2, n-1)
```