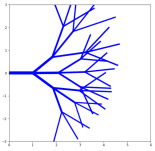
ENSAE TD noté, rattrapage, 3 mars 2021

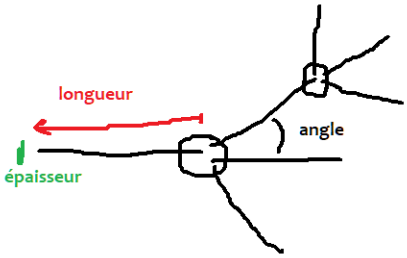
Le programme devra être envoyé par mail au chargé de TD et au professeur. Toutes les questions valent deux points excepté les questions 5 et 8 qui en valent trois.

1

On veut dessiner un arbre avec matplotlib de façon algorithmique comme celui-ci :



Pour se faire, on s'inspire des fractales. On part d'un segment de longueur L, d'épaisseur E. A une extremité, on trace trois autres segments, un peu plus petit, un peu moins épais, avec trois angles différents comme ceci.



Pour obtenir la première image, il suffit de répéter plusieurs fois l'opération. Pour la suite, on définit un segment (ou une branche) par un dictionnaire :

```
seg = {'x1': 0, 'y1': 0,  # première extrémité
       'x2': 1.0, 'y2': 0.0,  # seconde extrémité
       'E': 0.001,  # épaisseur
       'angle': 0  # angle, c'est l'angle entre le segment et l'axe des abscisses.
}
```

1) Le code suivant permet de tracer plusieurs segments avec matplotlib. Ecrire quelques lignes qui utilisent cette fonction pour tracer deux segments choisis comme vous voulez.

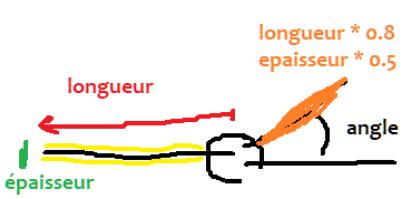
```
import matplotlib.pyplot as plt


def draw(segments, L=1):
    fig, ax = plt.subplots(1, 1, figsize=(8, 8))
    maxxy = 0
    for seg in segments:
        ax.plot([seg['x1'], seg['x2']], [seg['y1'], seg['y2']], 'b-', lw=seg['E'] * 10000)
    ax.set_xlim([0, 2 * L])
    ax.set_ylim([-L, L])
    return ax

draw([segment()]);
```

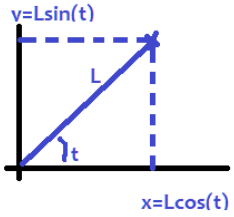
Cette fonction vous permet de vérifier visuellement que votre code est correct.

2) Ecrire une fonction qui calcule la distance d'un segment défini par un dictionnaire.

```
def distance(seg):
    return ...
```

3) Ecrire une fonction qui prend le segment noir et jaune comme argument et construire le segment orange dans la figure qui suit.






```
def nouveau_segment(seg, delta_angle):
    return ...
```

4) On crée maintenant une fonction qui appelle la fonction précédente et crée trois segments.

```
def nouveau_segment_3(seg, delta_angle):
    return [ ... ]
```

5) Il faut maintenant répéter le même processus sur plusieurs niveaux. La fonction retourne une liste de segments. A chaque itération, la fonction appelle la fonction appelle la fonction précédente sur chaque segment d'un niveau pour créer tous les segments du niveau suivant (3 points).

```
def arbre(seg, profondeur=3, delta_angle=math.pi/6):
    return [ ... ]
```

6) On veut paramétrer la décroissance de l'arbre. Reprendre les trois fonctions `nouveau_segment`, `nouveau_segment_3`, `arbre`. Créer une nouvelle version de chacune qui prend aussi comme paramètre : `facteur_L`, `facteur_E`. Au niveau `n`, la longueur est `L`, au niveau `n+1`, la longueur est `L * facteur_L`. On procède de même pour l'épaisseur.

7) Combien l'arbre contient-il de segments en fonction du nombre de niveaux ?

8) Les arbres dessinés ainsi sont tous les mêmes. On souhaite ajouter un peu d'aléatoire. Suggérer et implémenter une solution (3 points).