ALGO
RECURSIVITE

Durée : 4h

- 1 - Fractales

Objectif : réaliser une la fractale « Flocon de Koch »

Pour l'affichage, récupérez le code ci-dessous dans les ressources.

```
import sys, pygame
pygame.init()
w,h = 1024, 768
screen = pygame.display.set_mode((w,h))
screen.fill((0,0,0))



play = True
clock = pygame.time.Clock()

while play:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            play = False
        if event.type == pygame.KEYUP:
            print(event.key, event.unicode, event.scancode)
            if event.key == pygame.K_ESCAPE:
                play = False

    clock.tick(60)
    pygame.display.flip()
```

Initialisation (ne pas toucher)

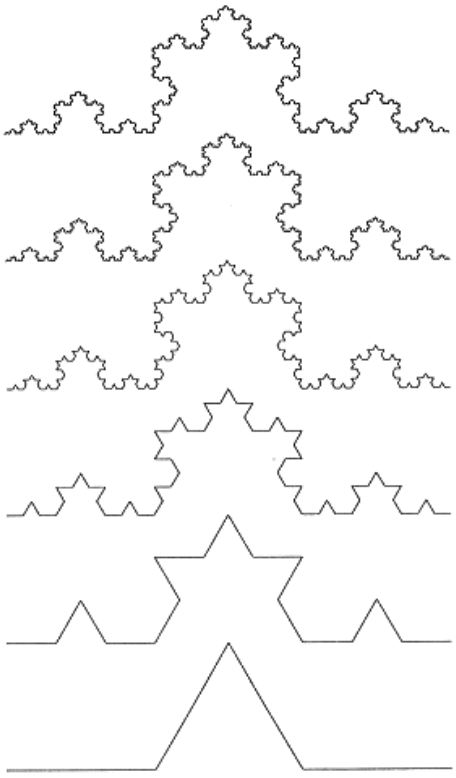
Les dessin se fait ici

Boucle qui conserve l'affichage et attends que l'utlisateur appuie sur une touche (ne pas toucher)

Exemple de tracé d'une ligne.
Paramètres : écan, couleur, point 1, point 2

```
pygame.draw.aaline(screen, (255,100,100), (42,10), (73,80))
```

D'autre fonctions ici si besoin :
https://www.pygame.org/docs/



- 2 - Défis sans boucles

Interdiction formelle d'utiliser des boucles, quelles qu'elles soient ! La récursivité

à réaliser :

- recIntDesc : Une fonction qui affiche les N premiers entiers (sens décroissant)
- recIntAsc : Une fonction qui affiche les N premiers entiers (sens croissant)
- recSum : Une fonction qui calcule la somme des N premiers entier
- recConcat : Une fonction qui prends en paramètre un tableau de chaînes de caractère et retourne une chaîne concaténée
- recPos : Une fonction qui prends un tableau et une valeur en paramètres et qui dit si la valeur est contenue dans le tableau

Astuce : parfois, un paramètre supplémentaire peut être utile, voir obligatoire !

- 3 - Les arbres

Définissez une classe Tree permettant de représenter un arbre n-aire avec les méthodes utiles :

- value, subTrees : Accesseurs
- __str__ : Chaîne pour affichage
- count : nombre de nœuds
- depth : profondeur
- list : parcours (à gauche) avec application de fonction lambda

- 4 - Pipotron

Réutilisez les arbres définis ci-dessus pour réaliser un pipotron = générateur automatique de phrases

https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9rateur_automatique_de_phrases

Définissez un arbre qui représente une structure de texte. On appelle ça une grammaire formelle (Exemple : https://fr.wikipedia.org/wiki/Forme_de_Backus-Naur).

Dans notre cas, un texte peut être constitué de phrases, elles même composées de sujets, verbes et compléments.

Une fois votre arbre réalisé, parcourez le avec la méthode « list » (parcours à gauche) avec une fonction anonyme qui tire aléatoirement les sujets, verbes, compléments, articles, adverbes, …