Exercice 1 : **Figures alphanumériques**

Le principe des deux programmes ci-dessous est le suivant :
« afficher une étoile de moins à la ligne suivante, jusqu'à n'avoir qu'une étoile »

Comparer et justifier l'affichage de chacun des programmes.

```
def star1(n):
    if n > 0:
        print(n * '*')
        star1(n-1)

star1(5)
```

```
def star2(n):
    if n > 0:
        star2(n-1)
        print(n * '*')

star2(5)
```

Exercice 2 : **Mutation sur place**

Copier-coller le programme ci-dessous :

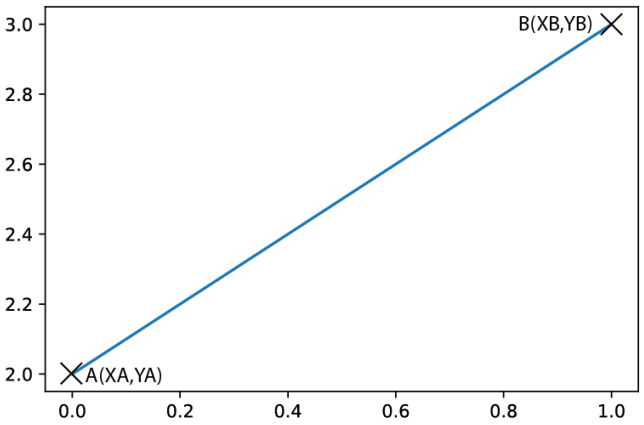
```
def transfo(B):
    B.append(1)

def recurs():
    if len(A)<100:
        transfo(A)
        recurs()
A=[]
recurs()
```

Q 1: Ce programme utilise-t-il une fonction récursive ? Si oui, quel est son cas trivial ?
Q 2: Ce programme génère-t-il des effets de bord ?
Q 3: Justifier le résultat constaté du programme.

Exercice 3 : **Flocon de Koch**

On rappelle la méthode pour tracer un segment avec matplotlib.

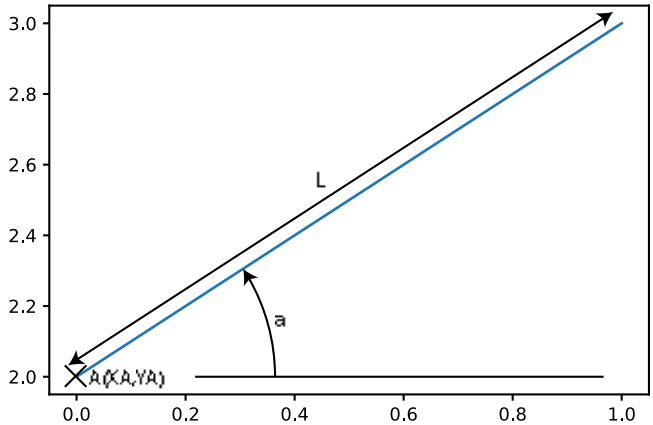
```
from matplotlib import pyplot as plt
plt.plot([XA,XB],[YA,YB])
```



Partie 1 : Notations à utiliser

Dans cet exercice, un segment orienté sera défini par 4 données :

- XA et YA : les coordonnées de son origine
- L : sa longueur
- a : l'angle orienté formé avec l'horizontale



Q 1: écrire une fonction qui trace un segment à partir des quatre arguments définis ci-dessus.

Partie 2 : Tracé du flocon de Koch

Q 2: Ecrire un programme qui trace les 3 côtés d'un triangle équilatéral. Leur longueur sera 100. Le programme utilisera trois fois la fonction définie **Erreur ! Source du renvoi introuvable.**

Le flocon de Koch est une image fractale. Partant d'une figure initiale très simple (un triangle équilatéral), les figures suivantes se construisent en modifiant chaque segment selon le principe suivant :

devient

Le segment de longueur *L* est donc remplacé par 4 nouveaux segments juxtaposés de même longueur $\frac{L}{3}$.

Ainsi, en partant d'un triangle équilatéral, et en appliquant la modification à chaque segment, on obtient :

devient

(n = 0) (n = 1)

Q 3: Décrire les opérations nécessaires à la transformation d'un segment quelconque. Puis écrire une fonction qui à partir des 4 arguments XA, YA, L et a, trace les 4 segments résultants de la transformation.

Q 4: Adapter la fonction précédente en fonction récursive :

1) Partie récursive : la fonction s'appelle 4 fois, sur chacun des quatre segments obtenus par le découpage.
2) Partie triviale : si la longueur du segment est inférieure à 0,1, la fonction trace le segment.

Q 5: Adapter le programme pour qu'il s'arrête à une profondeur d'appel récursif de 5. Mettre en place un compteur pour connaitre combien de fois la fonction a été appelée.

Exercice 4 : **Fusion de listes triées.**

Supposons que deux listes triées par ordre croissant L1 et L2 sont en mémoire.
On cherche à obtenir 1 liste contenant les éléments de L1 et de L2, elle-même triée par ordre croissant.

Exemple :
Les deux listes en donnée :

```
L1=[2,5,9]
L2=[3,5,19,24,56]
```

Deviennent

```
L=[2,3,5,5,9,19,24,56]
```

- Cette opération complexe est décomposée en opérations simples dont le principe est le suivant :
  - On compare les premiers éléments de chaque liste entre eux ;
  - Le plus petit est sorti de sa liste ;
  - Il est mis au bout de la nouvelle liste ;
  - On recommence autant que nécessaire.

Q 1: Exprimer un critère de fin pour cet algorithme.

Q 2: Proposer deux versions de programme réalisant ce traitement :

- l'un basé sur une récurrence ;
- l'autre basé sur une récursivité.