Dessiner avec un ordre

La fonction (procedure) **`dessine`** permet de donner une *séquence* d'ordres à la tortue sous forme d'une chaîne de caractères.

Paramètres:

- chaine de caractères appelée **`figure`**
- longueur ℓ
- angle α exprimé exprimé en degrés

La tortue interprètera chacun des caractères de la chaîne comme une commande à effectuer :

caractère	ordre
F	avance d'une longueur l
+	tourne vers la gauche de l'angle défini α
-	tourne vers la droite de l'angle défini α

```
def dessine(figure,l,alpha):
    for i in figure:
        if i == 'F':
            t.forward(l)
        if i == '+':
            t.left(alpha)
        elif i == '-':
            t.right(alpha)
```

Par exemple, l'appel de la fonction avec les paramètres `'++F-F--F-F'` ,100 , 45 permettrait de tracer une « maison » dont les segments font 100 pixels:

```
dessine('++F-F--F-F' ,100 , 45)
```

QUESTIONS:

1. Quelle instruction permet de tracer:
   A. un carré
   B. un triangle
   C. un pentagone

2. Ecrire une procédure **`cinq_carres()`** qui dessine cinq carrés emboîtés ayant comme sommet commun la position initiale de la tortue. Cette procédure devra utiliser la fonction **`dessine`**.
3. Que dessine l'instruction: **`dessine('F+F-F-F+F',20,90)`**

Dériver avec un ordre

Dériver un ordre consiste à remplacer chacun des caractères 'F' d'un ordre donné par une autre chaîne de caractères.
Par exemple, la dérivation de **`'F+F'`** par **`'F-F'`** produit **`'F-F+F-F'`**.

Vous allez écrire une fonction **`derive`** qui accepte deux chaînes en paramètres, F1 et F2, et renvoie la dérivation de la première par la seconde.

Cette fonction retourne une chaine appelée **`figure`** qui est construite de la manière suivante:

- **`figure`** vaut '' au debut (chaine vide)
- une boucle bornée parcourt la première chaine de caractères F1: `` **`for c in F1`**:` ``
- Si le caractère **`c`** est la lettre **`'F'`**, alors ajouter la chaine F2 à **`figure`**
- sinon ajouter le caractère **`c`**

QUESTIONS:

1. Ecrire le script en python de la fonction **`derive.`**

2. Quel est le résultat de l'appel de la fonction avec les arguments suivants:

```
derive('+FF','+FF')
```

Dérivées successives

Il est possible de dériver plusieurs fois de suite un ordre par une autre chaîne de caractères. On obtient alors la dérivation -ième de l'ordre par la chaîne.

Par exemple les dérivations successives de `'F'` par `'+FF'` produisent :

```
>>> derive_nieme('F', '+FF', 1)
'+FF'
>>> derive_nieme('F', '+FF', 2)
'++FF+FF'
>>> derive_nieme('F', '+FF', 3)
'+++FF+FF++FF+FF'
```

QUESTIONS:

Proposez une fonction **`derive_nieme`** qui accepte deux chaînes **`F1`** et **`F2`**, et un entier **`n`** en paramètres, et renvoie la dérivation -ième de la première chaîne par la seconde.
Quel est le resultat de l'instruction: **`derive_nieme('F','F+F-F-F+F',2)`**

Fractales

Pour dessiner les figures suivantes, On utilisera la fonction **`dessine.`** **`Parametrer`** la vitesse avec l'instruction `speed(0)` du module `turtle` pour accélerer le déplacement de la tortue.

Courbe quadratique de Koch

La figure suivante est obtenue avec la fonction **`koch`** et les valeurs suivantes pour les paramètres:

- l'entier **`n`** vaut 4
- la chaine de caractères F1 vaut 'F'
- la chaine de caractères F vaut 'F+F-F-F+F'
- la longueur l vaut 4
- l'angle: 90°

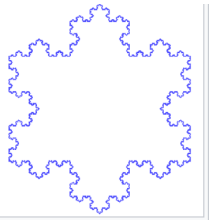
```
def koch(F0,F1,l,angle,N):
    for n in range(1,N+1):
        fig = derive_nieme(F0,F1,n)
        dessine(fig,l,angle)
```

QUESTIONS:

1. Ecrire l'instruction qui dessine la figure quadratique de koch. Utiliser la fonction **`koch`**.
2. Dessiner la figure à l'aide de votre calculatrice/tablette.

Flocons de Koch

La figure suivante est obtenue en dessinant à partir de la chaine issue de la dérivée 4e de de 'F--F--F par 'F+F—F+F'. Les segments font 4 pixels. Et l'angle fait 60°.



QUESTIONS:

1. Ecrire l'instruction qui dessine la figure du flocon de koch. Utiliser la fonction **`dessine`**, ainsi que celle **`derive_nieme:`**
   `dessine(derive_nieme(..),..,..)`
2. Dessiner la figure à l'aide de votre calculatrice/tablette.