CHAPITRE 5 : EXERCICES RECURSIVITE

Exercice 1 : récursivité sur une chaîne de caractères

Soit une chaîne de caractères, écrire un algorithme récursif permettant de déterminer sa longueur.

Exercice 2 : passer de l'itératif au récursif

Rendre récursive la fonction somme suivante :

```
def somme(L):
    s = 0 :
    for val in L :
        s = s + val
    return s
```

Exercice 3 : générer récursivement une liste

Ecrire une fonction récursive d'argument n qui génère la liste : $[n, n-1, n-2, \ldots, 1, 0]$

Exercice 4 : PGCD

Le plus grand commun diviseur de deux nombres peut être déterminé à l'aide de l'algorithme d'Euclide.

Celui-ci affirme que le pgcd de a et b, b étant un entier non nul, est le même que le pgcd de b et c, avec c le reste de la division euclidienne de a par b. Le pgcd de a et de 0 est a.

a. Ecrire une fonction récursive calculant le pgcd de a et b.

b. En utilisant la fonction précédente, écrire une fonction qui prend en entrée deux nombres entiers naturels non nuls a et b représentant une fraction $\frac{a}{b}$ et qui renvoie un couple d'entiers c et d tels que $\frac{c}{d}$ est l'écriture sous forme irréductible de $\frac{a}{b}$.

Exercice 5 : palindromes

Un mot est un palindrome si on peut le lire dans les deux sens de gauche à droite et de droite à gauche. Exemple KAYAK est un palindrome. Ecrire une fonction récursive permettant de vérifier si un mot est palindrome.

12 Découvrir la récursivité croisée

→ FICHE 5

On considère les fonctions suivantes :

```
def u(n):
    if n == 0:
        return True
    return v(n - 1)

def v(n):
    if n == 0:
        return False
    return u(n - 1)
```

Que calcule `u(n)` ? Le démontrer en utilisant un raisonnement par récurrence.

Soit un tableau X de N entiers, écrire une fonction récursive simple permettant de déterminer le maximum du tableau

```
def maximum(T):
    if len(T) == 1:
        return T[0]

    # principe de la recherche dichotomique
    m = len(T)//2
    max1 = maximum(T[:m])
    max2 = maximum(T[m:])
    if max1 > max2:
        return max1
    return max2
```

Exercice 9.3

Décomposer un problème en sous-problèmes
Concevoir des solutions algorithmiques

La méthode du paysan russe est un très vieil algorithme de multiplication de deux nombres entiers déjà décrit, sous une forme légèrement différente, sur un papyrus égyptien rédigé autour de 1650 avant J.-C. Il s'agissait de la principale méthode de calcul en Europe avant l'introduction des chiffres arabes.

Les premiers ordinateurs l'ont utilisé avant que la multiplication ne soit directement intégrée dans le processeur sous forme de circuit électronique. Sous une forme moderne, il peut être décrit ainsi :

```
Fonction Multiplication(x, y) :
    p = 0
    TANT QUE x>0 :
        Si x est impair :
            p = p + y
        x = x //2
        y = y + y
    Retourner p
```

1. Appliquer cette fonction pour effectuer la multiplication de 105 par 253. Détailler les étapes en remplissant le tableau suivant :

x	y	p
105	253	...
...	...	...

2. On admet que cet algorithme repose sur les relations suivantes :

$$x*y = \begin{cases} 0 & si\ x = 0 \\ (x\ //\ 2)*(y+y) & si\ x\ est\ pair \\ (x\ //\ 2)*(y+y)+y & si\ x\ est\ impair \end{cases}$$

Proposer alors la fonction équivalente selon la méthode récursive.

Exercice 9.4

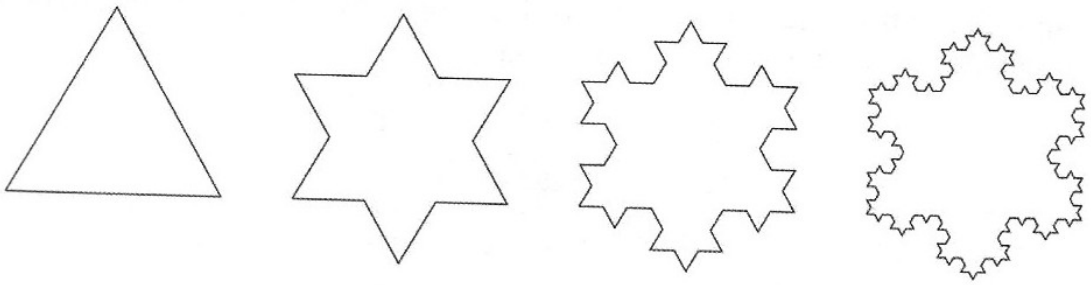
Mobiliser les concepts
Concevoir des solutions algorithmiques

L'objectif de cet exercice est de réaliser la fractale de Von Koch à l'aide du module Python *turtle*. **Il faudra donc lire la documentation attentivement.**

Une fractale est une sorte de courbe mathématique un peu complexe extrêmement riche en détail, et qui possède une propriété intéressante visuellement : lorsque l'on regarde des détails de petite taille, on retrouve des formes correspondant aux détails de plus grande taille (auto-similarité). Cela nous rappelle étrangement la récursivité ! La première courbe à tracer a été imaginée en 1904 par le mathématicien suédois Niels Fabian Helge von Koch.

Le principe est simple : on divise un segment initial en trois morceaux, et on construit un triangle équilatéral sans base au-dessus du morceau central. On réitère le processus n fois, n est appelé l'ordre. Dans la figure suivante on voit les ordres 0, 1, 2 et 3 de cette fractale.

Et si l'on trace trois fois cette figure, on obtient successivement un triangle, une étoile, puis un flocon de plus en plus complexe :



1. Proposer une fonction récursive Python permettant de dessiner la fractale de Von Koch en lui donnant comme paramètres l'ordre et la longueur du segment initial.
2. Proposer une fonction permettant de faire le flocon complet à partir de la fonction réalisée dans la question précédente.