La suite de Prouhet-Thue-Morse

Définition :

On considère la suite (u_n) définie sur $\mathbb{N}$ par son premier terme $u_0 = 0$ ainsi que par les relations $u_{2n} = u_n$ et $u_{2n+1} = 1 - u_n$ pour tout entier naturel *n*.

Exemples :

Calculer u_1, u_2, u_3, u_4, u_5.

$u_1 = 1$

$u_2 = 1$

$u_3 = 0$

$u_4 = 1$

$u_5 = 0$

On ne peut pas trouver de formule explicite pour u_n.

La suite de Prouhet-Thue-Morse a des applications dans de nombreux domaines tels que la théorie des nombres, l'informatique théorique, le football...

Programme Python :

On est obligé d'utiliser des listes.

Pour *n* pair, on a $u_n = u_{\frac{n}{2}}$.

Pour *n* impair, on a $u_n = 1 - u_{\frac{n-1}{2}}$.

```
def terme_indice(n):
    L=[0]
    for i in range(1,n+1):
        if i%2==0:
            L.append(L[i//2])
        else:
            L.append(1-L[(i-1)//2])
    return L[n]
```

`i//2` renvoie le quotient entier de `i` par 2, ce qui est nécessaire pour travailler avec les numéros des éléments de la liste `L` (les numéros des éléments correspondent aux indices des termes de la suite).

On peut par exemple chercher u_{2020}.

On trouve 1.

Lien avec l'écriture en base deux (admis sans démonstration) :

- Si *n* compte un nombre pair de 1 dans son écriture en base deux, u_n est nul.
- Si *n* compte un nombre impair de 1 dans son écriture en base deux, u_n vaut 1.

Un ***L-system*** ou un **système de Lindenmayer** est un système de grammaire formelle, inventé en 1968 par le biologiste hongrois Aristid Lindenmayer (1925 – 1989).

Un *L-system* modélise le processus de développement et de prolifération de plantes ou de bactéries.

Voir article Wikipedia

L'idée est d'associer une instruction de la tortue Python à chaque valeur de u_n (terme d'indice *n* de la suite définie au début). On décide que 0 signifie « avancer d'un certain nombre d'unités » et l « tourner à gauche de 60° (sans avancer) ».

```
from turtle import *
import turtle as tl

n = int(input("rang de la suite"))
L=[0]

for i in range(1,n+1):
    if (i%2==0):
       L.append(L[i//2])
    else:
       L.append(1-L[(i-1)//2])

tl.penup()
tl.setposition(-100,-100)
tl.pendown()
tl.speed(0)

for k in L:
    if L[k]==0:
       tl.backward(20)
    else:
      tl.left(60)
tl.mainloop()
```

1°) Réaliser à la main la figure obtenue pour la valeur $n = 4$ saisie en entrée.

2°) Réaliser le programme Python correspondant puis l'exécuter pour plusieurs valeurs de *n*. Utiliser trinket.io.

3°) Exécuter le programme pour de très grandes valeurs de *n* en réduisant le déplacement de la tortue pour voir la figure. Il faut réduire le déplacement de la tortue pour voir la figure en entier.

Lorsque *n* est très grand (de l'ordre de 10000), la tortue décrit un ensemble fractal bien connu appelé flocon de Von Koch (« courbe » de Koch ou « flocon » de Koch).
Se documenter à son sujet.

Somme des entiers naturels signés

Pour tout entier naturel *n*, on pose $S_n = (-1)^{u_0} \times 0 + (-1)^{u_1} \times 1 + ... + (-1)^{u_n} \times n$. On peut écrire $S_n = \sum_{k=0}^{k=n} (-1)^{u_k} \times k$.

Autrement dit, on additionne les entiers naturels en leur affectant un signe selon les valeurs des termes de la suite (u_n).

1°) Effectuer la somme « à la main » pour les entiers signés de 0 à 4.

2°) Programmer cette somme et déterminer les valeurs de *n* (jusqu'à 100) pour lesquelles la somme vaut 0.