Fractale Tapis de Sierpinski

import turtle

turtle.tracer(0,0) # accélération du tracé

turtle.screensize(2000,2000) # taille fenêtre graphique

turtle.pu()

turtle.goto(-500,0)

turtle.pd()

def dessiner(courbe, longueur, angle):

""" réalise une représentation graphique d'une courbe donnée par des chaines de caractères """

for caractere in courbe:

if caractere == '+': turtle.left(angle)

elif caractere == '-': turtle.right(angle)

elif caractere in ['F', 'G']: turtle.forward(longueur)

#dessiner('F', 50, 120)

def regleKoch(chaine):

nouvelleChaine = '' # on crée une nouvelle chaine de caractères VIDE

for lettre in chaine: # on épelle la chaine de caractères donnée en paramètres

if lettre == 'F': # si dans l'ancienne chaine, il y a un 'F'

nouvelleChaine = nouvelleChaine + 'F-G+F+G-F' # alors, on écrit F-G+F+G-F dans la nouvelle chaine

elif lettre == 'G': # si dans l'ancienne chaine, il y a un 'F'

nouvelleChaine = nouvelleChaine + 'GG' # alors, on écrit GG dans la nouvelle chaine

else :

nouvelleChaine = nouvelleChaine + lettre # sinon, on reporte la lettre telle quelle

return nouvelleChaine

def courbeKoch(motifInitial, niter):

"""

appelle niter fois regleKoch pour créer la courbe de Koch

"""

courbe = motifInitial # on part du motif initial donné par l'utilisateur en paramètres

for i in range(niter):

nouveauMotif = regleKoch(courbe) # on trouve le nouveau Motif à partir du motif de départ

courbe = nouveauMotif # on dit que le nouveau Motif est maintenant le motif de départ

return courbe

#courbe = courbeKoch('F',3)

#dessiner(courbe,50, 120)

def tapis(motifInitial, niter):

courbe = courbeKoch(motifInitial, niter)

tapis = ''

for \_ in range(3):

tapis += courbe

tapis += '--'

return tapis

longueur = 2

angle = 120

niter = 6

dessiner(courbeKoch('F', niter), longueur, angle)

turtle.update() # accélération du tracé

turtle.exitonclick() # permet la fermeture de la fenêtre graphique