Informatique : devoir n°5 (non surveillé)

Tous les programmes doivent être écrits dans un unique fichier nommé `DM5_votrenom.py`. Les noms des fonctions doivent être respectés. Les fichiers non exécutables ne seront pas corrigés. Les commentaires expliquant le code, en nombre modéré, seront appréciés (utiliser `#`). Les noms des variables doivent être explicites.

Le langage Python dispose d'un module `turtle` permettant de réaliser des figures en déplaçant une tortue symbolisée par un triangle. Au départ, la tortue est tournée vers l'est et l'unité de longueur est le pixel. Trois instructions seront utiles (x et a pouvant être de type entier ou flottant) :

> `forward`(x) : fait avancer la tortue de x pixels,
>
> `left`(a) et `right`(a) : font tourner la tortue respectivement de a degrés vers la gauche ou vers la droite (la tortue n'avance pas).

On pourra tester par exemple le code suivant :

```
from turtle import *
from math import sqrt

forward(100)
left(45)
forward(100*sqrt(2))
right(135)
forward(100)
mainloop()   # à mettre à la fin d'un code Turtle
```

Pour les autres instructions, se référer à la documentation : `https://docs.python.org/3.9/library/turtle.html`.

1) Écrire un programme permettant de réaliser la maison ci-dessous, sachant que :

$$AB = BC = CD = DE = EA = 100,\ (AB) \perp (AE) \text{ et } (AB) \perp (BC).$$

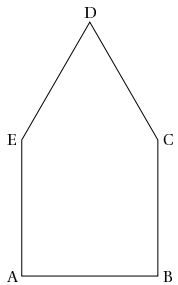


Figure « maison »

2) Écrire une fonction `polygone` qui, recevant un réel ℓ et un entier naturel $n \geqslant 3$, trace un polygone régulier à n côtés où chaque côté est de longueur ℓ.

Comme on réalise régulièrement les mêmes instructions, on se propose de décrire les figures selon les règles suivantes. Une figure est définie par la donnée d'un triplet (ℓ, a, m) où ℓ est un réel représentant une longueur en pixels, a un réel correspondant à l'angle de rotation et m est une chaîne de caractères (appelée motif) permettant de tracer la figure et dans laquelle `F` signifie avancer de ℓ, `+` tourner de a degrés vers la gauche et `-` tourner de a degrés vers la droite.

Par exemple, la figure $(50, 90,$ `'F+F+F+F'`$)$ représente un carré de 50 unités de côté.

3) Donner un triplet décrivant la maison (on commencera au point A).

4) Écrire une fonction `dessiner` qui, recevant un triplet (ℓ, a, m), trace la figure correspondante.

Par exemple, `dessiner(50, 90, 'F+F+F+F')` doit tracer un carré de 50 unités de côté.

5) Écrire une fonction `spirale` qui, recevant un entier naturel n, renvoie une chaîne de caractères de la forme `F+FF+FFF+FFFF+...` (la dernière séquence doit comporter n fois `F`).

```
>>> spirale(5)
'F+FF+FFF+FFFF+FFFFF+'
```

On pourra ensuite tracer cette spirale avec la fonction `dessiner` pour différentes valeurs de l'angle. Pour accélérer la tortue, on pourra ajouter `delay(0)` et `speed(0)` au début de la fonction `dessiner`. On peut aussi ajouter `hideturtle()` pour ne pas afficher la tortue. Enfin, `reset()` efface le dessin et remet la tortue au centre de l'écran.

Dans la suite on se donne deux motifs, appelés axiome et règle, permettant de décrire l'évolution d'une figure en remplaçant à chaque étape les `F` de l'axiome par la règle. Considérons par exemple l'axiome `'F+F+F+F+'` et la règle `'F+F-F'`.Le motif initial de la figure (à l'étape 0) est donc `'F+F+F+F+'`. À la première étape on remplace chaque `F` du motif par `F+F-F`: on obtient le motif `'F+F-F+F+F-F+F+F-F+F+F-F+'`. À la deuxième étape on obtient le motif `'F+F-F+F+F-F-F+F-F+F+F-F+F+F-F-F+F-F+F+F-F+F+F-F-F+F-F+F+F-F+F+F-F-F+F-F+'` , et ainsi de suite.

6) Écrire une fonction `suivant` qui, recevant un motif et une règle, renvoie le motif obtenu à l'étape suivante.

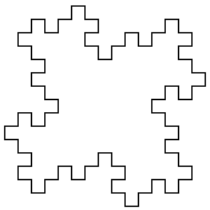
```
>>> suivant('F+F+F+F+', 'F+F-F')
'F+F-F+F+F-F+F+F-F+F+F-F+'
```

7) Écrire une fonction `evolution` qui, recevant un axiome, une règle et un entier naturel n, renvoie le motif obtenu à l'étape n.

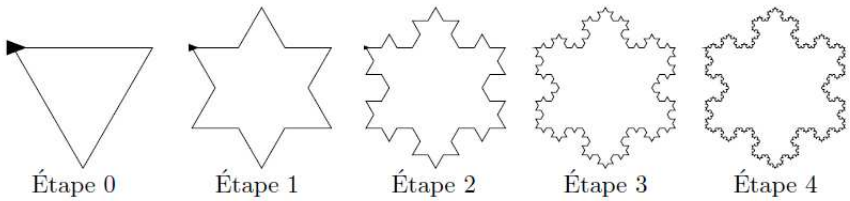
```
>>> evolution('F+F+F+F+', 'F+F-F', 0)
'F+F+F+F+'
>>> evolution('F+F+F+F+', 'F+F-F', 1)
'F+F-F+F+F-F+F+F-F+F+F-F+'
>>> evolution('F+F+F+F+', 'F+F-F', 2)
'F+F-F+F+F-F-F+F-F+F+F-F+F-F+F+F-F+F+F-F-F+F-F+F+F-F+F+F-F-
F+F-F+'
```

On peut alors dessiner la figure, par exemple à l'étape 3 :

```
>>> dessiner(10, 90, evolution('F+F+F+F+', 'F+F-F', 3))
```



8) Déterminer un angle, un axiome et une règle permettant de dessiner le **flocon de Koch** (l'échelle des images a été ajustée pour une meilleure visibilité) :



Autres exemples :

Le dragon de Lévy : $a = 45$, axiome `'F'`, règle `'-F++F-'`.

Le triangle de Sierpiński : $a = 120$, axiome `'F'`, règle `'F+F-F-F+F'`.

Le carré de Koch : $a = 90$, axiome `'F+F+F+F+'`, règle `'F-F+F+F-F'`.

9) La tortue est amusante mais pas très rapide. Pour des dessins un peu évolués, il vaut mieux utiliser les modules `numpy` et `matplotlib` qu'on importe comme suit :

```
import numpy as np
import matplotlib.pyplot as plt
```

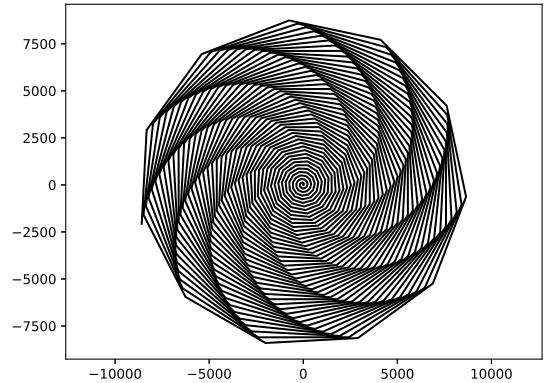
La fonction `plot` prend comme arguments deux listes et trace une ligne brisée reliant les points dont les abscisses sont les éléments de la première liste et les ordonnées les éléments de la deuxième.

Par exemple, pour tracer le carré dont les sommets ont pour coordonnées $(1, 1)$, $(2, 1)$, $(2, 2)$ et $(1, 2)$, on écrira :

```
abscisses = [1, 2, 2, 1, 1]
ordonnees = [1, 1, 2, 2, 1]
plt.axis('equal')   # pour avoir un repère orthonormal
plt.plot(abscisses, ordonnees)
plt.show()
```

Écrire une fonction `tracer` qui, recevant comme la fonction `dessiner` un triplet (ℓ, a, m), trace la figure correspondante en utilisant `matplotlib`.

Il faut donc créer deux listes formées respectivement des abscisses et des ordonnées des points par lesquels passe la tortue.

Indication : on pourra définir trois variables `x`, `y` et `t` représentant respectivement l'abscisse, l'ordonnée et la direction de la tortue.

```
>>> tracer(10, 33, spirale(500))
```