Algo & Prog, avec Python
(L1-Sciences)
TP n° 4, Automne 2016

N.B. Vous avez téléchargé le fichier `turtle.cfg` dans le dossier de vos programmes tortue ! Il sera utilisé pour initialiser les dimensions (*width*, *height*) de la fenêtre graphique, la forme (*shape*) de la tortue (`'arrow'`, `'classic'`, `'turtle'`, `'circle'`, `'square'`...), le *mode* indiquant le cap 0 (*logo* pour le Nord, *standard* pour l'Est), etc. Si vous ne l'avez pas fait (*et vous avez tort*), faites suivre l'instruction `import` de votre fichier de TP des instructions `mode('logo')` et `shape('arrow')`. Terminez votre fichier par l'instruction `mainloop()`.

Graphisme polaire

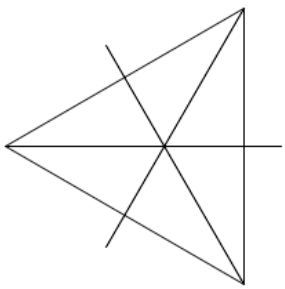
Exercice 4.1 Lorsque vous écrivez plusieurs programmes graphiques dans l'éditeur, vous voudrez sans doute faire une pause d'une ou deux secondes entre chaque dessin. Pour cela, il existe une fonction primitive `sleep(dt)` qui prend un nombre réel `dt` et suspend l'exécution du programme pendant `dt` secondes. Par exemple, `sleep(2)` réalise une pause de 2 secondes.

a) Cherchez (docs, Google) dans quel module se trouve la fonction `sleep`.

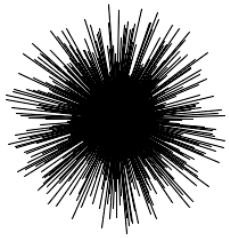
b) Vérifiez son fonctionnement en exécutant le code ci-dessous :

```
angle = 360 // 5            # pour ne pas le recalculer 5 fois
for i in range(5) :         # dessin d'un pentagone régulier
  forward(100)
  sleep(0.5)
  right(angle)
  sleep(0.5)
```

Exercice 4.2 Programmez une fonction `equi(c)` dessinant un triangle équilatéral de côté dans la direction du cap courant de la tortue, sur sa gauche. Mais vous tracerez aussi, au passage, une portion des bissectrices intérieures du triangle, de même longueur que le côté.



Exercice 4.3 Programmez une fonction `oursin(n,c)` modélisant un oursin ayant `n` épines, chaque épine étant de longueur aléatoire dans `[0,c]`. Vous utiliserez la fonction `randint(a,b)` dont vous trouverez l'explication dans les docs…



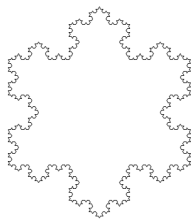
Graphisme cartésien

Exercice 4.4 Programmez une fonction `cercle(a,b,r)` dessinant un cercle de centre le point (a,b) et de rayon r. L'état de la tortue en sortie n'est pas spécifié. *Il vous faudra un peu de trigonométrie...*

Exercice 4.5 a) Définissez une fonction `line(x1,y1,x2,y2)` traçant le segment reliant les points de coordonnées (x1 ; y1) et (x2 ; y2). L'état de la tortue après l'exécution de cette fonction est non spécifié.
b) Demandez à l'ordinateur de résoudre l'exercice pour enfants proposé sur la page Web :
`http://jlsigrist.com/images/fils2.jpg`

Exercices optionnels

Exercice 4.6 a) Programmez le *flocon de Von Koch*, obtenu en greffant une courbe de Von Koch [cours page 19] sur chaque côté d'un triangle équilatéral :



b) Expérimentez l'alternative de placer la courbe de Von Koch à l'intérieur plutôt qu'à l'extérieur du flocon. Vous obtiendrez *l'anti-flocon* cristallin…

N.B. Vous trouverez des informations sur les courbes fractales avec Google. Un petit livre de Benoit Mandelbrot, l'inventeur de cette théorie, est à la B.U. : ***Les Objets Fractals****.*

Exercice 4.7 a) Prolongez l'exercice 4.4 avec l'image `fils3.jpg`
b) Programmez un joli travail en fils tendus (*string art*) :
`http://www.mathcats.com/crafts/stringart.html`

Exercice 4.8 Programmez le dessin de la page Web :
`http://en.wikipedia.org/wiki/File:Turtle-Graphics_Polyspiral.svg`
You will find some pseudocode to be implemented in Python…

Algo & Prog, avec Python
(L1-Sciences)
TD n° 4, Automne 2016

Exercice 4.1 Quelles sont les différences entre la graphisme *cartésien* et le graphisme *polaire* ? Quelles sont les primitives essentielles Python de chacun d'eux ? Donnez un exemple de petit dessin utilisant chacun de ces deux volets de la programmation graphique.

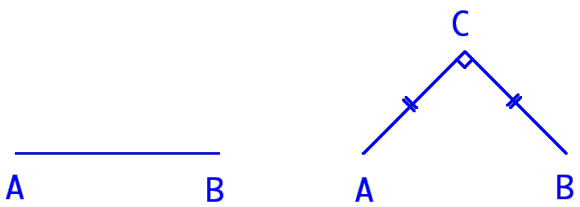
Exercice 4.2 Le *Jeu du Chaos*[1]. Il s'agit d'un programme célèbre montrant la possible émergence d'une figure régulière à partir de l'aléatoire. On considère les sommets A(0 ; 200), B(-200 ; -200) et C(200 ; -200) d'un triangle isocèle, et le processus suivant. On part d'un point quelconque du canvas $M(x_0 ; y_0)$ et :

1. Soit H l'un des sommets A, B, C au hasard.
2. Soit I le milieu du segment MH. On dessine le point I.
3. M devient I. Continuer à l'étape 1.

Programmez une fonction `chaos(n)` sans résultat répétant n fois l'affichage du point I décrite ci-dessus, en utilisant une tortue pour dessiner. Pour n grand, on voit une figure bien connue des chaoticiens émerger du brouillard !

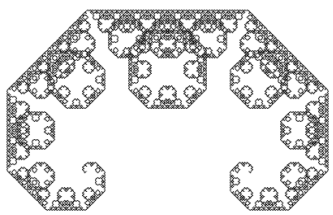
N.B. i) Pour afficher un point, vous vous documenterez sur la fonction `dot` du package `turtle`. ii) Utilisez tracer(False) lorsque n est grand (5000), et tracer(True) pour voir la construction pas à pas lorsque n est petit (n = 50). Passez ce programme sur machine après le TD !!

Exercice 4.3 Programmez l'approximant de niveau `n` et de taille `T` de la célèbre courbe fractale du dragon. Elle est construite de la manière suivante. La courbe de niveau 0 est un segment AB de longueur T. La courbe de niveau 1 est la ligne brisée ACB, toujours avec AB = T. Cette transformation est itérée sur chacun des sous-segments AC et CB, et ainsi de suite.



a) Dessinez à la main les niveaux 3, 4 et 5.

b) Programmez la fonction `dragon(n,T)`, en traduisant la description ci-dessus en <u>récurrence</u> sur `n`. L'argument `T` [la taille] représente la distance entre le point de départ A et le point d'arrivée B.



le dragon de niveau 13

[1] http://www.cut-the-knot.org/Curriculum/Geometry/SierpinskiChaosGame.shtml