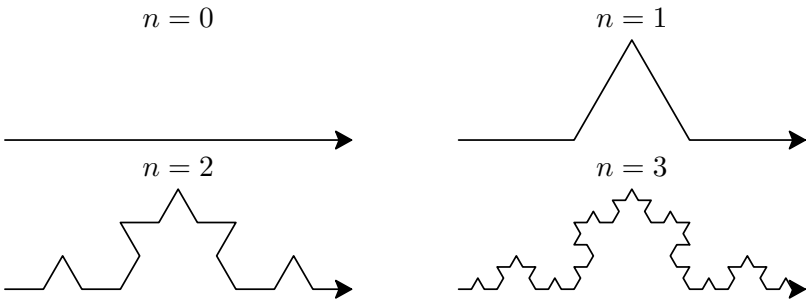
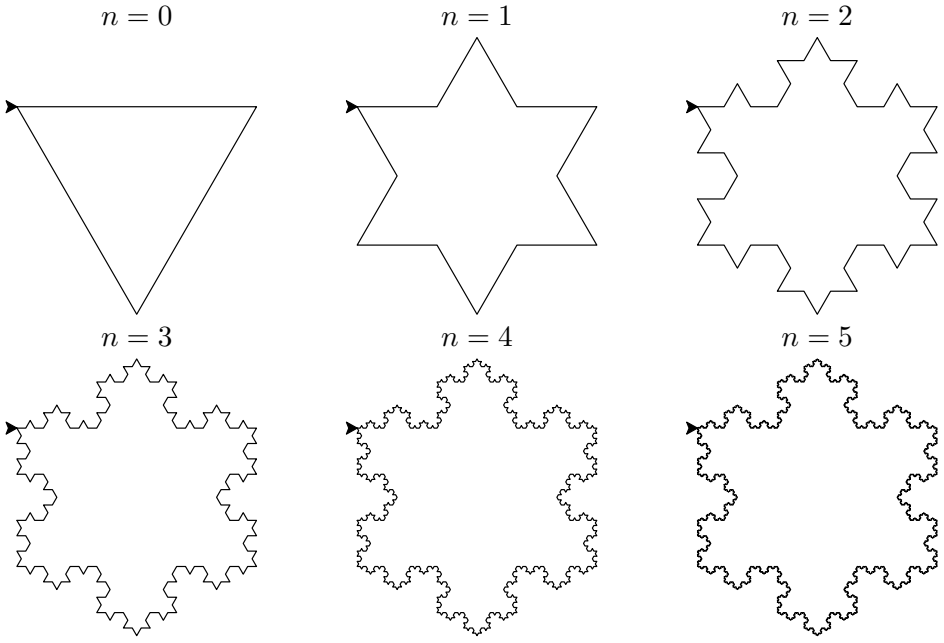
TP 12
Algorithmes récursifs

Le but du TP est de tracer le **flocon de Von Koch**. D'abord on trace la courbe de Von Koch : c'est une courbe qui est obtenue par étapes successives en partant d'une ligne droite puis en la coupant en trois segments égaux et en remplaçant le segment central par un triangle équilatéral. Chaque étape est numérotée par un entier n :



Puis on en déduira le flocon complet :



I Introduction au module `turtle`

Le module `turtle` implémente en Python une méthode ancienne d'apprentissage de la programmation (remontant au début des années 70 !) basée sur une interface graphique appelée **la tortue**.

Il faut imaginer qu'au démarrage du programme se tient au centre de l'écran une tortue tenant un crayon. Les instructions du programme donnent notamment à la tortue l'ordre d'avancer en ligne droite d'un certain nombre de pas, ou bien de tourner à gauche ou à droite. La tortue laisse donc un trait derrière elle, ce qui permet de tracer des figures intéressantes, surtout quand on combine ces instructions avec des fonctions et des boucles Python. Éventuellement, d'autres fonctions permettent de configurer l'épaisseur ou la couleur du tracé. Le crayon peut aussi être relevé — auquel cas la tortue se déplace sans laisser de tracé derrière elle — jusqu'à ce que, éventuellement, il soit de nouveau abaissé.

Le système de coordonnées utilisé est tel qu'au démarrage la tortue est au centre de l'écran aux coordonnées $(x, y) = (0, 0)$, et tournée vers la droite. Les dimensions de la fenêtre (donc les absisses et ordonnées minimales et maximales, c'est à dire les coordonnées du bord de la fenêtre) peuvent dépendre des configurations de l'utilisateur, on gardera donc la tortue dans une zone raisonnable au centre de la fenêtre.

Pour commencer on chargera le module `turtle` avec cette ligne **au tout début du fichier** et à exécuter une seule fois :

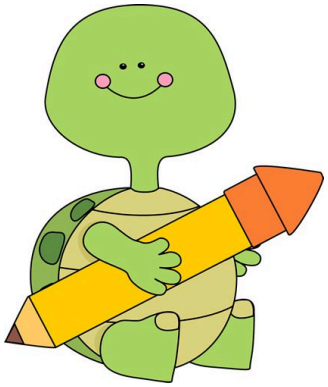


Figure 3. – Source : https://www.mycutegraphics.com/

```
from turtle import *
```

On peut alors utiliser toutes les fonctions du module sans écrire à chaque fois le préfixe `turtle`.

Tester ensuite directement (mode script, dans une cellule, puis exécuter) les lignes suivantes pour vérifier que tout fonctionne correctement :

```
#%%
reset()
forward(200)
done()
#%%
```

La première ligne demande à bien démarrer avec une nouvelle fenêtre blanche. La deuxième donne l'ordre à la tortue d'avancer de 200 pas (ajuster éventuellement ce nombre selon la taille de votre écran et de la fenêtre qui apparait). Enfin la dernière ligne met le programme en pause jusqu'à ce que l'utilisateur ferme la fenêtre. On commencera donc toujours nos programmes par ce `reset()` et on terminera toujours par `done()`.

Pour la liste des commandes disponibles, on pourra se référer à

- https://perso.limsi.fr/pointal/_media/python:turtle:turtleref.pdf : un aide-mémoire (pour les enfants).
- https://docs.python.org/fr/3/library/turtle.html : la documentation officielle (pour les vrais de vrais).

Nous en utiliserons en fait un tout petit nombre, les autres sont à découvrir et tester par vous-même !

- `reset()` : ré-initialise la fenêtre et la tortue à son point de départ, première instruction du programme.
- `done()` : affiche la fenêtre et attend que l'utilisateur la ferme, dernière instruction du programme.
- `forward(n)` : avance de `n` pas.
- `left(r)` : tourne à gauche de l'angle `r` exprimé en degrés.
- `right(r)` : idem mais tourne à droite.
- `goto(x, y)` : déplace la tortue à la position (x, y). Sera utile au début du programme pour tenter de centrer la figure, précédé éventuellement d'un levé de crayon.
- `penup()` : lever le crayon. La tortue se déplace, mais ne laisse pas de tracé derrière elle.
- `pendown()` : rabaisser le crayon.

Exercice 1 *Échauffement*
Tracer un carré, puis un triangle équilatéral.

Exercice 2
Écrire une fonction `polygone(n)` qui trace un polygone régulier à n côtés.

Pour éviter que le dessin ne déborde de l'écran, on pourra diviser la longueur du côté par n (en effet la taille du dessin final sera à peu près proportionnelle à n et à la longueur du côté).

Attention, il est très important que `reset()` et `done()` ne soient pas *dans* la fonction, mais en dehors dans la cellule, comme ceci :

```
#%%
def polygone(n):
    ...

reset()
polygone(7)
done()
#%%
```

Exercice 3
Tracer la première étape ($n = 1$) d'une courbe de Von Koch, en calculant d'abord à la main les angles et les longueurs en jeu.

II Le flocon de Von Koch

Pour tracer un flocon de Von Koch, on va d'abord tracer la courbe de Von Koch et écrire une fonction récursive qui prend deux arguments :

- `n` : le numéro de l'étape qu'on est en train de tracer. Pour $n = 0$ la courbe est une simple ligne droite, pour $n = 1$ c'est la ligne polygonale de quatre morceaux tracée précédemment.
- `L` : un paramètre qui indique la longueur du morceau que l'on est en train de tracer, et qu'il faut diviser par 3 dans les appels récursifs.

Exercice 4
Écrire une fonction `vonkoch(n, L)` récursive qui trace la courbe de Von Koch : pour $n = 0$ elle trace une ligne droite de longueur `L`, et sinon elle s'appelle récursivement en divisant la longueur par 3 et entre les appels récursifs la tortue doit tourner de façon appropriée.

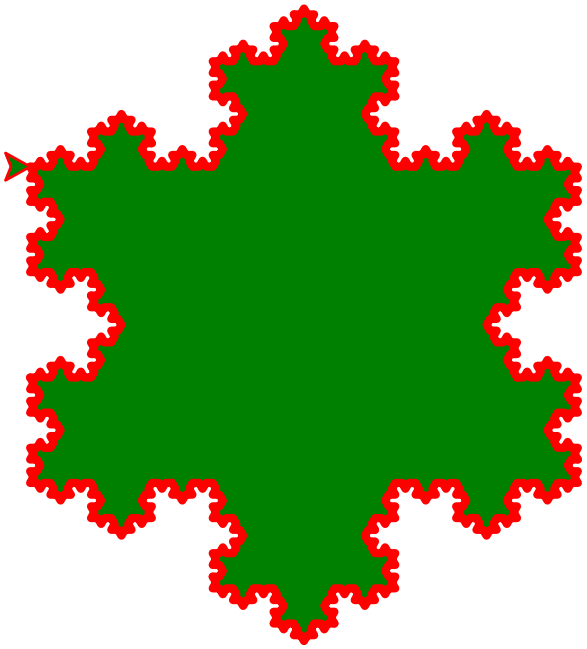
Pour l'appeler proprement, on écrira alors (directement dans une cellule à part) :

```
#%%
reset()
vonkoch(n, L)
done()
#%%
```

Pour dessiner le flocon complet sous forme hexagonale, il suffit… d'appeler trois fois la fonction précédente, en tournant du bon angle !

Exercice 5
Écrire une fonction `flocon(n, L)` (qui n'est plus récursive) qui trace le flocon de Von Koch complet avec n étapes.

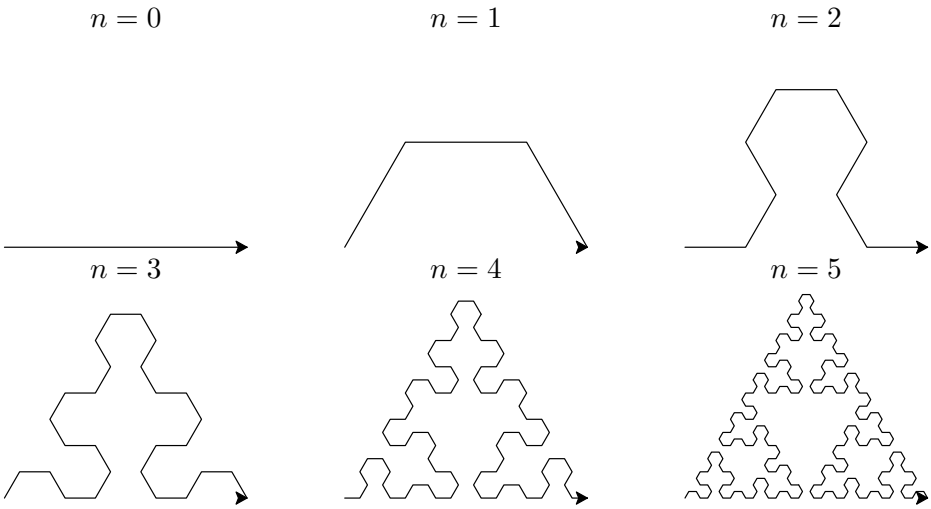
Pour que ce soit plus joli, il faut lire la documentation !



Exercice 6
Lire la documentation ou l'aide mémoire pour ajuster la couleur et l'épaisseur du trait, la couleur de remplissage, le titre de la fenêtre, et pour bien centrer la figure.

III Le triangle de Sierpinski

Il s'agit de la courbe obtenue par la suite de transformations de la figure suivante :



Encore une fois pour $n = 0$ il s'agit d'une ligne droite, et pour $n = 1$ on appellera cela une *tuile*. À chaque étape, les morceaux de ligne droite sont remplacés par des tuiles. Mais attention ! Il y a deux types de tuiles, celles tournant d'abord vers la gauche (comme le cas $n = 1$ ici) et celles tournant d'abord vers la droite (le miroir du cas $n = 1$; la première et la troisième du cas $n = 2$).

Exercice 7
Quels sont les longueurs et les angles en jeu pour tracer une tuile ? Observer que dans le dessin final on voit beaucoup de triangles équilatéraux et d'hexagones réguliers. Puis écrire comme échauffement une fonction `tuile(L)` qui tracer l'étape $n = 1$ d'une tuile de base `L`.

On écrira alors **deux fonctions mutuellement récursives** `tuile_g(n, L)` et `tuile_d(n, L)`. Pour $n = 0$ ce sont toutes les deux des lignes droites, pour $n = 1$ `tuile_g` est la fonction précédente et `tuile_d` est exactement

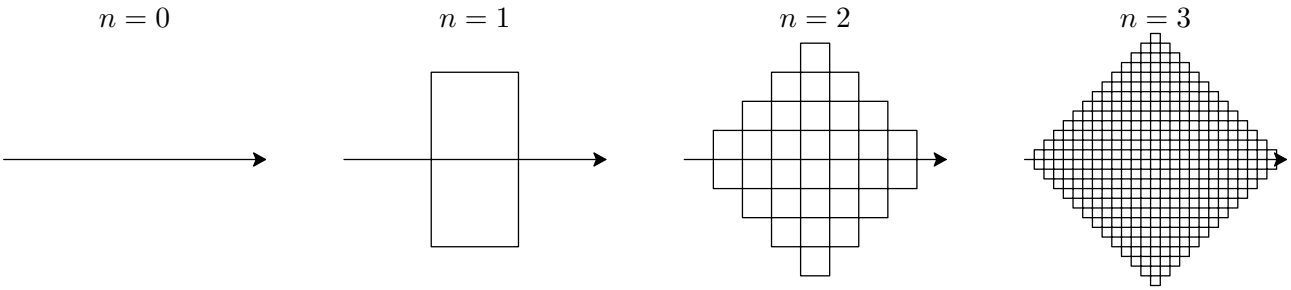
son miroir (ce qui revient à échanger les rotations à droite et à gauche) — mais encore une fois on n'a en fait pas besoin de distinguer le cas $n = 1$ dans la récursivité. Enfin, chacune de ces fonctions fait appel récursivement à elle-même ainsi qu'à l'autre, avec le rang $n - 1$ et la longueur $L/2$.

Attention à un dernier piège : après avoir tracé une tuile, il est important de faire tourner la tortue pour qu'elle soit bien dans le même sens qu'au départ (observez bien la tortue du cas $n = 1$ ci-dessus). Cela est nécessaire pour que l'appel récursif fonctionne correctement, et qu'à chaque étape la tortue se retrouve bien exactement dans la même position, qu'elle ait tracé juste avant une ligne droite ou bien une tuile plus petite.

Exercice 8
Écrire les fonctions `tuile_g(n, L)` et `tuile_d(n, L)`, puis tester en appelant l'une des deux au choix.

IV La courbe de Peano

C'est la courbe obtenue par la succession suivante d'étapes.



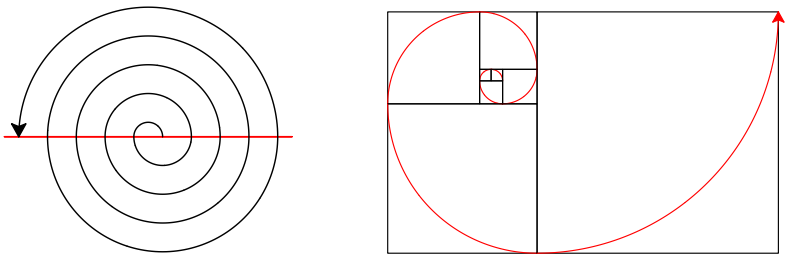
Là encore pour $n = 0$ il s'agit d'une simple ligne droite. Pour $n = 1$ le motif peut être obtenu en parcourant cette ligne dans diverses ordres possibles, quitte à repasser plusieurs fois au même point. À chaque étape, chaque segment est remplacé par une courbe de Peano de taille divisée par 3.

Exercice 9
Écrire la fonction `peano(n, L)` qui trace la courbe de Peano à l'étape n, en partant d'une longueur `L`.

On peut démontrer que « à la limite », la courbe obtenue remplit réellement tout un carré. On trace une courbe de dimension 1 remplissant une surface de dimension 2 !

V La spirale de Fibonacci

Il n'y a ici plus de récursivité. Une spirale simple est obtenue par une succession de morceaux de cercles, dont le diamètre augmente régulièrement. Dans le cas le plus simple, c'est une succession de demi-cercles, dont les rayons forment une progression arithmétique. Un autre cas célèbre est la spirale de Fibonacci, formée de quarts de cercles dont les rayons suivent la suite de Fibonacci.



Exercice 10
1. Écrire une fonction `spirale(n)` qui trace une spirale simple, avec n demi-cercles consécutifs. On pourra par exemple augmenter les rayons de à chaque itération.
2. Écrire la fonction `spirale_fibonacci(n)` qui trace la spirale avec n quarts de cercles dont les rayons sont proportionnels aux termes de la suite de Fibonacci.