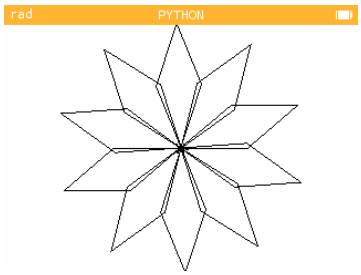
Du flocon à la neige

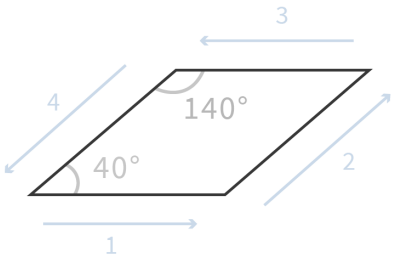
Cette activité s'appuie sur le module Turtle de l'application Python. Après avoir créé un nouveau script, on écrira la commande `from turtle import *` afin de pouvoir utiliser les fonctions de cette bibliothèque qui nous permettront de piloter une petite tortue à l'écran.

Notion de fonction : tout commence avec un losange

Notre objectif dans cette première partie est de piloter la tortue afin qu'elle dessine un flocon de neige. Le dessin de ce flocon de neige sera basé sur la répétition d'une même figure géométrique par rotation.



Lors de l'exécution du script, la tortue est située au centre de l'écran et tournée vers la droite. Nous allons pouvoir la déplacer à l'aide des commandes `forward(l)` et `backward(l)` lui permettant d'avancer ou reculer de l pixels ; et des commandes `left(a)` et `right(a)`, faisant pivoter la tortue sur sa gauche ou sur sa droite de a degrés. Pour se représenter un pixel, il faut savoir que l'écran constitue un rectangle de 320 pixels de largeur sur 222 pixels de hauteur.



Dans un premier temps, on aimerait que la tortue réalise le losange comme indiqué ci-contre.
Comme cette figure est amenée à être répétée plusieurs fois pour réaliser notre flocon, nous allons créer une **fonction** `losange(l)` dans laquelle nous programmerons la série d'instructions qui permet d'aboutir à ce dessin.
Chaque fois que l'on souhaitera répéter cette série d'instructions, on se contentera d'appeler la fonction.

Pour définir la fonction losange, on utilise la syntaxe suivante : `def losange(l) :`
La lettre indiquée entre parenthèses est une **variable** que nous remplacerons, à l'exécution, par la valeur souhaitée. Ici, on souhaite que l désigne la longueur des côtés du losange.
Toutes les instructions qui appartiendront à notre fonction devront être **indentées**, c'est-à-dire décalées par rapport au début de la ligne.

1. Ecrire, à l'aide de `forward(l)` et `left(a)`, la série d'instructions permettant de réaliser le dessin demandé.

2. Exécuter `losange(90)` et vérifier que le dessin correspond bien à l'image ci-dessus.

```
def losange(l):
  forward(l)
  left(40)
  forward(l)
  left(140)
  forward(l)
  left(40)
  forward(l)
  left(140)
```

Il est possible de n'écrire que les 4 premières lignes de la fonction, puis de procéder à un copier/coller pour recopier les 4 suivantes !

Notion de boucle : du losange au flocon

Dans le même script, on cherche maintenant à écrire une nouvelle fonction `flocon(l)` qui permettra de tracer plusieurs fois le losange déjà programmé en suivant une rotation afin d'obtenir le dessin en haut de cette activité. La variable l désigne toujours la longueur des côtés du losange.
Nous allons répéter les mêmes instructions : tracer un losange, puis pivoter. Pour faciliter cette répétition, on utilisera la boucle `for i in range(n)` qui permet de répéter n fois les instructions indentées à la suite.

1. Combien de losanges compte-t-on sur le dessin ? Quel est donc l'argument à indiquer dans la boucle ?

   La boucle est à exécuter 10 fois. On écrira donc `for i in range(10)` en tête de notre programme.

2. Dans la boucle, on pourra appeler la fonction `losange(l)` puis l'instruction pour commander à la tortue de pivoter.
   Quel est le pivot à effectuer entre deux losanges ? On pensera à prendre en compte l'inclinaison avec laquelle la tortue termine son premier dessin.

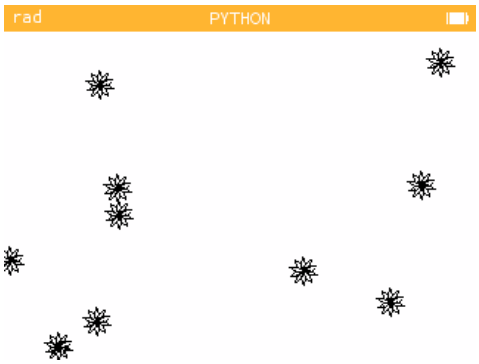
   On doit répartir 360° sur 10 losanges, soit un pivot de 36° entre chaque. Si la tortue termine dans sa position initiale après avoir dessiné le losange, on pourra utiliser la commande `left(36)`.

3. Ecrire la fonction à l'aide des questions précédentes puis exécuter `flocon(60)`.

```
def flocon(l):
  for i in range(10):
    losange(l)
    left(36)
```

Avec de l'aléatoire : de la neige à l'écran !

Notre objectif est maintenant de dessiner plusieurs flocons de petite dimension à l'écran. Nous avons déjà une fonction `flocon(l)` que nous allons pouvoir appeler autant de fois que nécessaire. Nous allons maintenant voir comment générer aléatoirement ces flocons à l'écran.
Pour cela, nous allons créer une nouvelle fonction `neige(k)` qui permettra de dessiner k flocons à l'écran. Cette fois, nous n'allons pas chercher à contrôler par une variable la dimension des flocons, mais plutôt leur nombre.



Se déplacer sur l'écran avec Turtle

Nous avons vu qu'à l'exécution du script, la tortue était positionnée au centre de l'écran, tournée vers la droite. Le centre de l'écran correspond à un point de coordonnées $(0;0)$. En effet, il est possible de se repérer sur l'écran en suivant un axe horizontal numéroté de -160 à 160 (on rappelle que l'écran fait 320 pixels de largeur) et un axe vertical numéroté de -111 à 111 (pour 222 pixels de hauteur).
Pour déplacer la tortue en un point donné, il suffit d'utiliser l'instruction `goto(x;y)`, x et y correspondant bien sûr aux coordonnées dans le repère décrit précédemment.
Attention toutefois puisque, lorsque la tortue se déplace, elle laisse une trace à l'écran : c'est tout le principe de Turtle ! Si l'on veut effectuer un déplacement sans laisser de tracé, il faudra utiliser les commandes `penup()`, pour relever le crayon, et `pendown()`, pour le reposer.

Générer des positions aléatoires

Si l'objectif est de tracer beaucoup de flocons à l'écran, et en nombre variable, il paraît difficile de programmer à l'avance toutes les positions à occuper par les flocons. Nous allons laisser le hasard décider.
Pour utiliser les fonctions qui vont nous permettre de générer des nombres aléatoires, nous allons utiliser le module `random` qui est à importer en tout début de script à l'aide de la commande `from random import *`.
Les coordonnées des flocons seront générées aléatoirement à l'aide de l'instruction `randint(a,b)` qui permet d'obtenir au hasard un entier compris entre les valeurs a et b.

1. Quelles sont les bornes a et b à utiliser pour générer aléatoirement la position de la tortue sur l'axe des abscisses ? Et sur l'axe des ordonnées ? Quelle commande écrira t-on alors pour déplacer la tortue en un point aléatoire de l'écran ?

x peut prendre n'importe quelle valeur entre -160 et 160, y peut prendre n'importe quelle valeur entre -111 et 111. On utilisera donc la commande `goto(randint(-160,160),randint(-111,111))`, ou bien on pourra définir deux variables x et y comme dans le script proposé ci-dessous.

2. Quels sont mouvements à répéter dans une boucle pour pouvoir tracer chacun des flocons ?

   On commence à générer aléatoirement une position, on lève le crayon avant d'aller sur cette position, on le repose, puis on dessine le flocon.
   Des variantes sont possibles bien sûr ! La boucle peut commencer sur le lever de crayon, par exemple.

3. Ecrire le programme `neige(k)` puis l'exécuter pour qu'il dessine 10 flocons. On fixera la dimension des flocons à 5px en appelant la fonction avec `flocon(5)`.

```
def neige(k):
  for i in range(k):
    penup()
    x = randint(-160,160)
    y = randint(-111,111)
    goto(x,y)
    pendown()
    flocon(5)
```

Un peu de couleur pour les plus rapides !

Il est possible de modifier la couleur du tracé de la tortue à l'aide de la commande `color()`. L'argument à indiquer entre parenthèses peut être de plusieurs natures : il peut s'agir d'une couleur pré-enregistrée à entrer entre guillemets (voir le menu du module Turtle pour connaître les couleurs disponibles), soit d'un code couleur RVB composé d'une triplet de trois entiers compris entre 0 et 255 et séparés par une virgule. Chaque nombre désigne une intensité, depuis l'absence de couleur (0) à l'intensité maximale (255), et correspond respectivement aux couleurs rouge, verte et bleue.

Essaye de trouver une couleur qui pourrait se rapprocher de celle de la neige ! Du cyan, par exemple ?
Et si l'on est un peu joueur, pourquoi ne pas générer une couleur aléatoire différente pour chacun des flocons ?