1 Exercice 1 - Dessinez avec la tortue

La module `turtle` intégré à la bibliothéque standard de python permet de réaliser des dessins géométriques à l'aide d'une *tortue virtuelle*.
Après avoir présenté quelques fonctionnalités de base, nous utiliserons ce module pour illustrer les boucles bornées vues en cours.

Mouvements	méthode associée	nature du paramètre	type de paramètre
Avancer	`forward`*(d)*	distance parcourue en pixels	`int` ou `float`
pivoter à droite	`right`*(a)*	angle de rotation de la tête en degré	`int` ou `float`
pivoter à gauche	`left`*(a)*	angle de rotation de la tête en degré	`int` ou `float`
aller au point de coordonnées	`goto`*(x, y)*	coordonnées du point	`int` ou `float`

1) Tester l'exemple suivant :

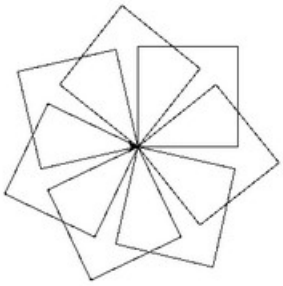
```
from turtle import *
forward(100)
left(90)
forward(100)
right(90)
forward(50)
right(90)
forward(150)
```

a) Commentez la première ligne
Remarques :

- La tortue commence toujours au point de coordonnées(0,0) situé au centre de la fenêtre `Python Turtle Graphics`.
- Pour réaliser un nouveau dessin, fermer d'abord cette fenêtre sinon la tortue repartira de l'état final précédent.

b) Définissez une procédure `carre` permettant de dessiner un carré et acceptant un paramètre `a` de type entier représentant la longueur de l'arête. Codez cette fonction d'abord sans boucle puis avec une boucle bornée `for`

c) À l'aide de votre procédure `carre`, définissez une procédure nommée `carres_tournants` qui dessine n carrés de côté 100 pivotant autour d'un sommet commun pour faire un tour complet. La figure suivante montre l'exemple pour n=7.



carres tournants

[1]

2) On peut également contrôler le crayon à l'aide de plusieurs instructions

Instructions	description
`up()`	relève le crayon
`down()`	abaisse le crayon (*le dessin peut reprendre*)

Testez le code suivant:

```
forward(100)
up()
forward(50)
down()
forward(100)
```

En utilisant les fonctions précédentes et une boucle bornée, définissez les procédures suivantes

a) `trois_carres(a)` dessinant trois carres d'arête `a` alignés horizontalement et séparés d'une distance `a`

b) `grille_horizontale(x, a)` dessinant `x` carres accolés horizontalement.

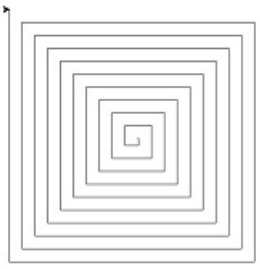
Conseils:

- *utilisez des petites distances de l'ordre de 10 à 20 pixels, et peu d'alternances pour que le dessin "reste" dans la fenêtre*
- *vous réglez la vitesse de la tortue à l'aide de la méthode* `speed` (https://docs.python.org/3.3/library/turtle.html?highlight=turtle#turtle.speed)

c) `grille_verticale(y, a)` dessinant `y` carres accolés verticalement.

d) `grille(x, y, a)` dessinant une grille de `x` sur `y` carrés

[1] Extrait de https://www.fil.univ-lille1.fr/~L1S1Inf



spirale carré

faisceau

2 Exercice 2

1) Definissez une procedure `spirale_carre()` permettant de réaliser le dessin ci-dessous.

A chaque tour le trait est plus long de 10 pixels : il y a 20 traits au total. (*vous pouvez ultérieurement modifier ces paramètres ou bien les demander à l'utilisateur*)

2) Le style du trait réalisé par le crayon peut être modifié :

Instructions	description
`width`*(e)*	fixe à *e* l'épaisseur du trait
`color`*(c)*	sélectionne une couleur *c* (type `str`) pour les traits

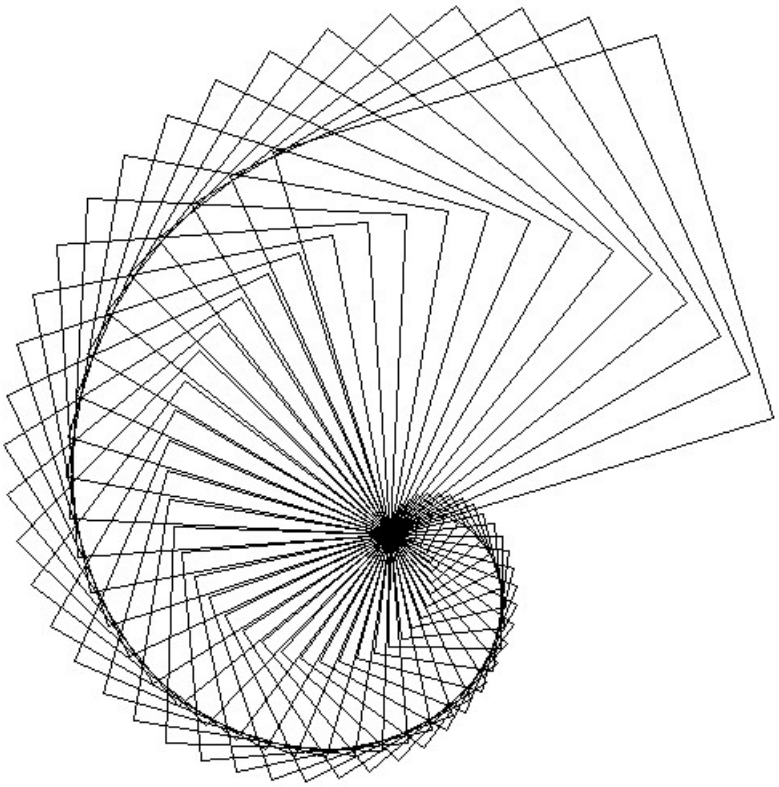
a) Testez le code suivant:

```
forward(100)
up()
forward(50)
down()
color("red")
width(5)
forward(100)
up()
forward (50)
down()
color("green")
width(10)
forward(50)
```

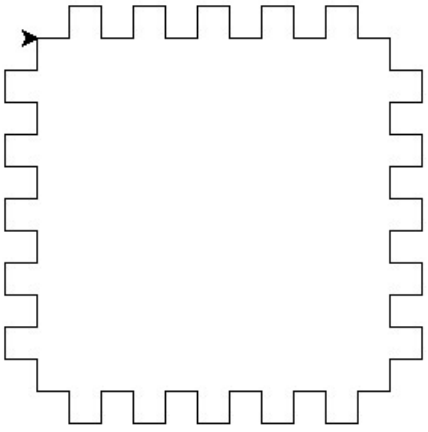
b) Définissez une procédure `faisceau(c)` acceptant comme paramètre `c` une couleur sous forme de chaîne de caractères et permettant de réaliser un trait de plus en plus large de couleur choisie par l'utilisateur.

3 Challenges

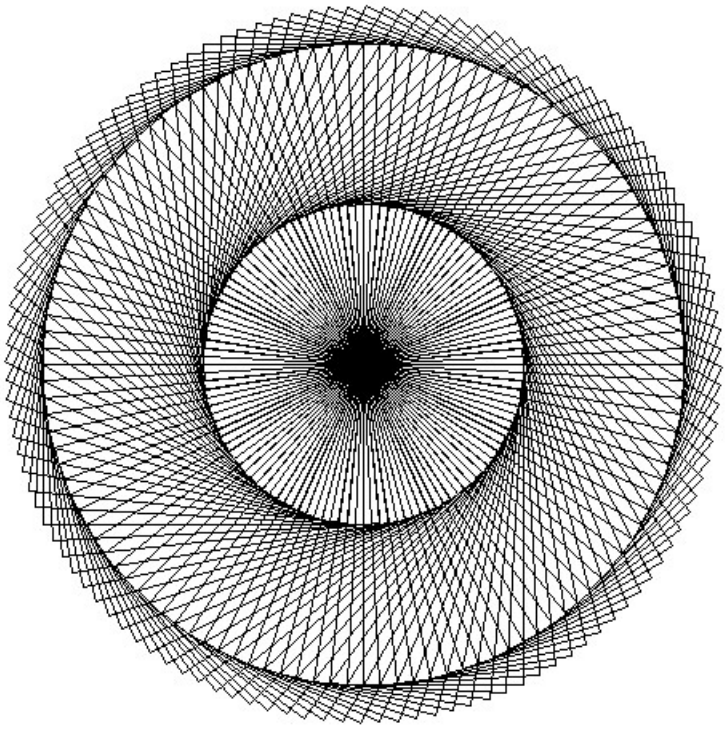
Réaliser les dessins représentés en utilisant le plus possible de boucles for



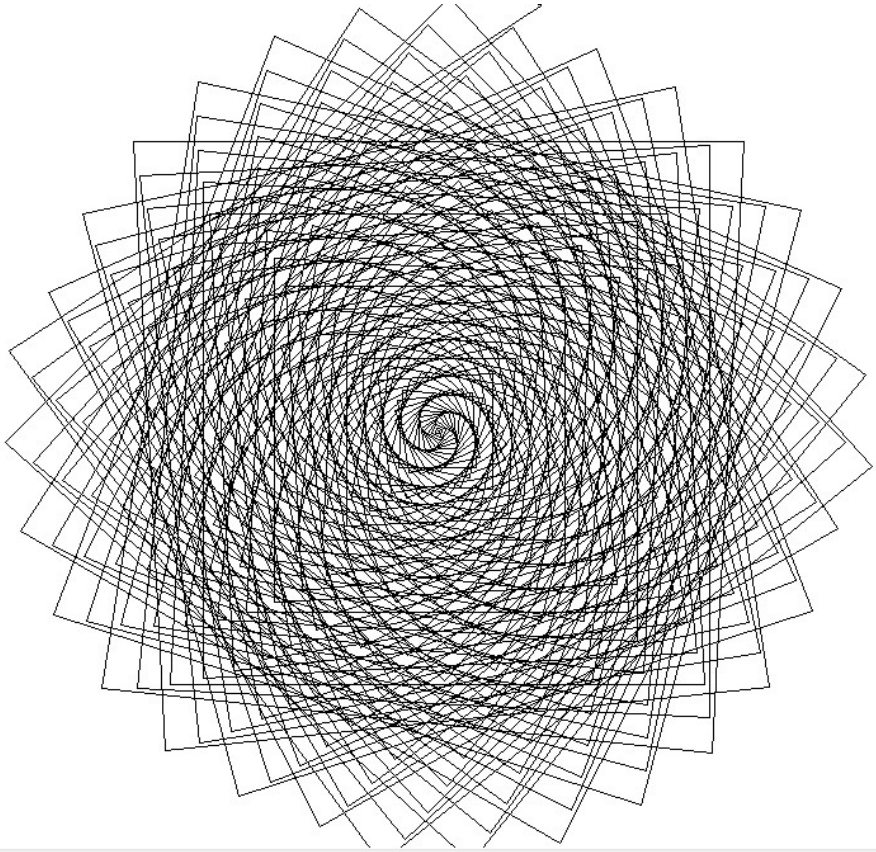
Challenge 1



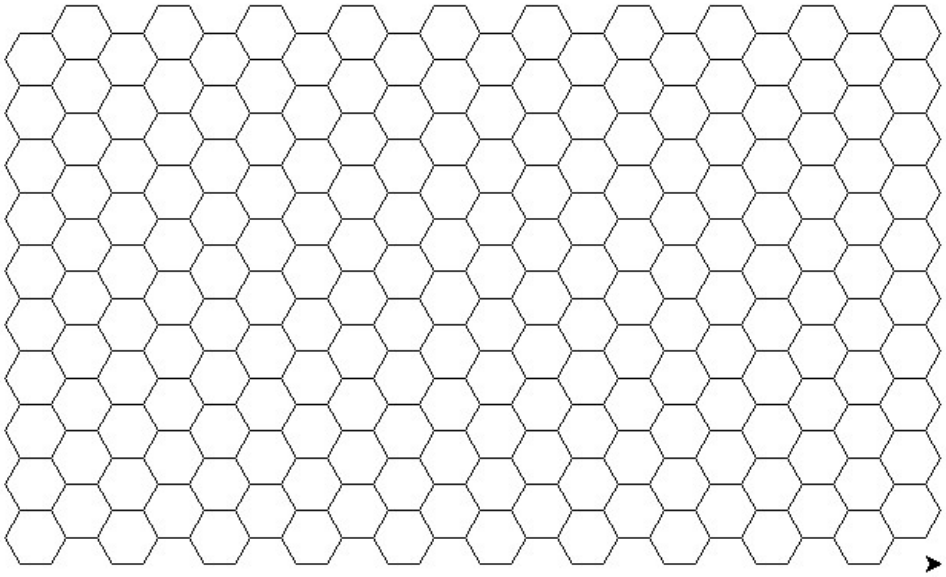
Challenge 2



Challenge 3 : *AIDE* : vous utiliserez une fonction annexe permettant de dessiner un rectangle



Challenge 4



Challenge 5