Chap. 1 Introduction à la programmation Python ACTIVITE

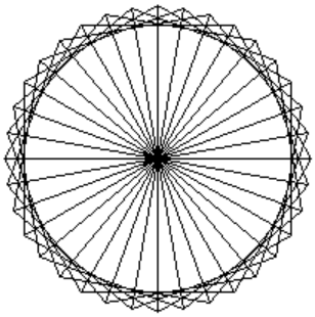
La tortue Python

Objectifs

- ▷ Utiliser le langage Python
- ▷ Définir et utiliser les notions de variable, boucle et fonction
- ▷ Utiliser la bibliothèque `turtle` de Python
- ▷ Tracer une figure géométrique

Introduction

Une « tortue » est un robot éducatif, initialement créé dans les années 40 et popularisé par Seymour Papert, capable de dessiner sur une feuille placée en dessous, et qui peut être contrôlé par des instructions programmatiques simples : avancer, reculer, tourner, ... Dans cette activité, nous utilisons une simulation informatique de la tortue réalisée avec le langage Python afin de tracer la figure ci-dessous :



Le module `turtle` de Python est un ensemble d'outils permettant de déplacer une tortue munie d'un crayon à la surface d'une feuille virtuelle permettant ainsi de dessiner à l'aide d'instructions simples.
Pour pouvoir utiliser la tortue Python, il faut **importer** la **bibliothèque** `turtle` :

```
1 from turtle import *
```

La tortue Python est contrôlée par quatre commandes de base : avancer (`forward` ou `fd`), reculer (`backward` ou `back`), tourner à gauche (`left`) et à droite (`right`). Chacune de ces commandes est invoquée en écrivant le nom de la fonction correspondante, suivi d'un nombre entre parenthèses, qui est le **paramètre** de la fonction, par exemple : `forward(100)`.

Tracer un carré

Le programme suivant est une suite de trois **instructions** :

```
forward(100)
right(90)
forward(100)
```

Lors de ses déplacements, la tortue laisse une trace, ce qui permet de dessiner des figures.

☞ **Compléter le programme ci-dessus pour tracer un carré.**

Remarques

Si l'on exécute le programme depuis un fichier, il faut ajouter l'instruction `exitonclick()` à la fin du programme, sinon la fenêtre d'affichage disparaît dès que l'exécution est terminée.

Utiliser une variable

Pour changer la taille du carré dessiné par le programme précédent, il faut changer plusieurs fois la valeur `100`. Cela fonctionne, mais il existe une solution plus efficace : stocker cette valeur dans une **variable** et utiliser cette variable plusieurs fois. Ici, on donne la valeur `50` à la variable `cote`. Cette opération s'appelle une **affectation** et est notée en Python avec le signe « = ».

```
cote = 50
forward(cote)
right(90)
forward(cote)
...
```

☞ **Compléter le programme ci-dessus pour dessiner un carré de côté 50.**

Utiliser une boucle

On remarque que ce programme répète quatre fois les mêmes deux instructions. Pour automatiser cette répétition, on peut réaliser une **boucle** à l'aide de l'instruction `for` :

```
cote = 50
for i in range(4) :  # exécuter le corps de la boucle 4 fois
    forward(cote)    # première instruction du corps de la boucle
    right(90)        # deuxième instruction du corps de la boucle
print("Fini !")
```

Le corps de la boucle (lignes 3 et 4) est un **bloc d'instructions** qui est délimité par son décalage vers la droite, ou **indentation**. La ligne 5 ne fait pas partie de la boucle et est exécutée une seule fois. Le texte qui suit un « croisillon » ♯ est appelé un **commentaire**, et n'est pas interprété par Python.

☞ **De manière générale, si l'on souhaite dessiner un polygone à n côtés, quel sera la mesure de l'angle qu'il faudra tourner après chaque déplacement ?**

☞ **Compléter le programme ci-dessous et vérifier qu'il fonctionne bien avec plusieurs valeurs de n.**

```
1 n = 12
2 cote = 50
3 angle = ...
4 for i in range(n) :
5     forward(cote)
6     right(angle)
```

Définir et utiliser une nouvelle fonction

Pour créer une nouvelle commande de la tortue qui permette de dessiner un polygone, on définit une **fonction** grâce à l'instruction `def`. Cette fonction a deux **paramètres** : le nombre de côtés `n` et leur longueur `cote`. De la même façon que pour une boucle, le bloc d'instructions qui constitue le corps de la fonction est indenté.

```
1 def polygone(n, cote) :
2     """ Tracer un polygone a n cotés de longueur "cote" """
3     # ici vient le corps de la fonction
4     # il doit être aligné avec sa description ci-dessus
```

☞ **Compléter le corps de la fonction ci-dessus avec les lignes 3 à 6 du code précédent. N'oubliez pas de maintenir l'indentation, c'est-à-dire le nombre d'espaces au début de chaque ligne.**
Une fois la fonction définie, il est possible de l'appeler en indiquant les valeurs de ses paramètres, appelés **arguments**, entre parenthèses :

```
1 polygone(6, 100)
2 polygone(30, 20)
```

☞ **Compléter le code ci-dessous pour obtenir la figure présentée dans l'objectif de cette activité.**

```
for i in range(...) :
    polygone(3, 100)
    right(...)
```

Bibliothèque **turtle**

Les principales fonctions du module `turtle` sont :

- `reset()` Tout effacer et recommencer à zéro
- `goto(x,y)` Aller au point de coordonnées `(x,y)`
- `forward(d)` Avancer de la distance `d` (exprimée en pixel)
- `backward(d)` Reculer de la distance `d` (exprimée en pixel)
- `left(a)` Tourner à gauche de l'angle `a` (exprimé en degré)
- `right(a)` Tourner à droite de l'angle `a` (exprimé en degré)
- `circle(r,a)` Trace un arc de cercle d'angle `a` et de rayon `r`
- `dot(r)` Tracer un point de rayon `r`
- `up()` Relever le crayon et interrompre le dessin (pour pouvoir avancer sans dessiner)
- `down()` Abaisser le crayon et reprendre le dessin (pour pouvoir recommencer à dessiner)
- `color(c)` Sélectionner la couleur `c` du crayon qui peut être une chaîne prédéfinie (`'red'`, `'blue'`, `'green'`, etc)
- `width(e)` Fixer l'épaisseur `e` du trait de crayon
- `begin_fill()` Activer le mode remplissage
- `end_fill()` Désactiver le mode remplissage
- `fillcolor(c)` Sélectionner la couleur `c` pour le remplissage
- `fill(1)` Remplir un contour fermé à l'aide de la couleur sélectionnée (on termine la construction par `fill(0)`)
- `write(t)` Texte doit être une chaîne de caractères délimitée par des `"` ou `'`
- `speed(v)` Définir la vitesse de déplacement de la tortue
- `title(t)` Donner le titre `t` à la fenêtre de dessin
- `ht()` Cacher la tortue (seulement le dessin)

La tortue commence toujours au point de coordonnées `(0,0)`, situé au centre de l'écran et est orientée par l'axe des abscisses. L'axe des ordonnées est orienté vers le haut. Les coordonnées et les distances sont mesurées en pixels et les angles en degrés. Les arcs de cercles sont parcourus dans le sens trigonométrique si le rayon est positif, et sens horaire si le rayon est négatif.

Voici toutes les fonctions de la bibliothèque `turtle` de Python.

Exemple : Taper le programme ci-dessous dans l'interpréteur de commande de IDLE, cela permettra de suivre ce qui se passe au fur et à mesure :

```
from turtle import *
forward(120)
left(90)
color('red')
forward(80)
```

Que voyez-vous à l'écran ?