TD : le module `turtle`

Le but de cette séance est de continuer à vous habituer à la programmation en Python et notamment aux notions de :

- structures conditionnelles ;
- boucles ;
- fonction.

Pour cela, nous manipulerons le module `turtle` dont le principe est de suivre dans une fenêtre l'évolution d'une tortue (symbolisée par une flèche) qui suivra vos instructions à la lettre.

1 Travail à préparer chez vous avant la séance

1. Quels sont les prototypes (liste d'arguments et leur type, liste de valeurs de retours et leur type) des fonctions `rectangle`, `carre` et `immeuble` présentées dans la section L'immeuble ci-dessous ?

2 Présentation du module `turtle`

Lors de cette séance de TD, vous serez ammenés à faire appel aux fonctions de base du module `turtle` suivantes :

- `turtle.forward(dist)` : faire avancer la tortue de `dist` unités ;
- `turtle.left(alpha)` : faire tourner la tortue sur la gauche d'un angle de `alpha` degrés ;
- `turtle.right(alpha)` : faire tourner la tortue sur la droite d'un angle de `alpha` degrés ;
- `turtle.up()` : faire léviter la tortue (sa trace ne s'écrira donc plus à l'écran) ;
- `turtle.down()` : stopper la lévitation de la tortue ;
- `turtle.goto(x, y)` : faire se déplacer la tortue jusqu'à la position `(x, y)`.

3 Le château de cartes

Dans cet exercice, vous allez tenter de dessiner à l'écran un château de cartes (fait de triangles superposés) similaire à celui-ci :

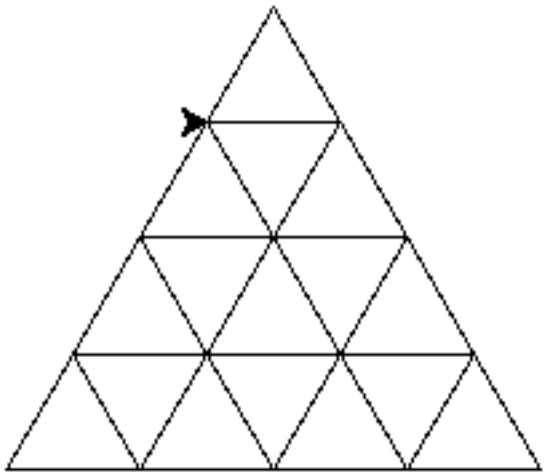

FIGURE 1 –

Pour cela, vous devrez tout d'abord être capable de tracer un triangle équilatéral à une position donnée.

2. Écrivez une fonction qui prenne en entrée une position (sous la forme de deux entiers `x` et `y`) et une taille `c` et trace à l'écran un triangle équilatéral de côté `c` ayant son bord inférieur gauche situé à la position `(x, y)`.
3. Écrivez une fonction qui prenne en entrée un nombre `n` et trace à l'écran un château de cartes dont la base est constituée de `n` triangles.

4 L'immeuble

Pour cet exercice, votre code final devra ressembler à :

```
import turtle

# [...]

n_etages = 5
n_fenetres = 3
```

```
immeuble(n_etages, n_fenetres)
turtle.exitonclick()  # Attend un clic avant de fermer la fenetre
```

L'exécution de ce code devra faire dérouler à l'écran une animation se terminant sur le dessin suivant :

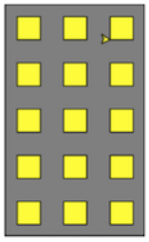

FIGURE 2 –

Pour cela, vous définirez 3 fonctions (à l'emplacement des points de suspension dans l'extrait de code ci-dessus) :

- `rectangle` permettra de dessiner un rectangle à l'écran ;
- `carre` permettra de dessiner un carré à l'écran ;
- `immeuble` permettra de dessiner un immeuble tel que celui représenté ci-dessus à l'écran.

4.1 La fonction rectangle

4. Écrivez une fonction rectangle qui permette de tracer à l'écran un rectangle de taille et de position spécifiées lors de l'appel de la fonction.

4.2 La fonction carre

5. Écrivez une fonction `carre` qui permette de tracer à l'écran un carré de taille et de position spécifiées lors de l'appel de la fonction. Est-il nécessaire de réécrire la fonction dans son ensemble (avec les appels successifs aux fonctions du module `turtle`) ou est-ce possible de s'en sortir en une ligne de code ?

4.3 La fonction immeuble

6. Écrivez une fonction `immeuble` qui permette de tracer un immeuble à l'écran, connaissant son nombre d'étages et le nombre de fenêtres par étage. N'hésitez pas à faire un schéma de l'immeuble sur papier pour vous rendre compte des dimensions à utiliser.

4.4 Un peu de *tuning*

7. Ajoutez aux fonctions nécessaires un paramètre facultatif qui permette de spécifier la couleur de remplissage des formes géométriques tracées. Utilisez ce paramètre facultatif pour demander de tracer l'immeuble en gris (`"grey"`) et les fenêtres en jaune (`"yellow"`). Pour cela, vous aurez besoin des fonctions `turtle.fillcolor(couleur)`, `turtle.begin_fill()` et `turtle.end_fill()` qui s'utilisent comme suit :

   ```
   turtle.fillcolor(couleur)
   turtle.begin_fill()
   # Ici, tracer le polygone
   turtle.end_fill()
   ```