Chapitre 6

Les modules de la librairie standard

Certaines fonctions comme `print`, `input`, `type`, etc. font partie du langage Python. Cependant, il existe beaucoup de fonctions qui ne font pas directement partie du langage mais qui sont disponibles dans des *modules*.

Par exemple, la plupart des fonctions mathématiques sont présentes dans le module `math`, on peut manipuler des dates avec `datetime`, tirer des nombre au hasard avec `random` ou encore dessiner avec le module `turtle`.

I Importer un module

Pour charger les fonctions d'un module, on utilise l'instruction `import`. Par exemple, l'instruction `import math` importe toutes les fonctions du module `math`. Pour y accéder, on utilise le nom du module suivi d'un point, comme ceci : `math.cos(12)`.

Il est possible de charger seulement une fonction d'un module avec l'instruction `from math import cos`. On peut alors utiliser directement la fonction `cos`, sans faire `math.cos`.

Enfin, il est possible d'importer toutes les fonctions d'un module comme ceci : `from math import *` mais cette pratique n'est pas recommandée.


Attention

L'utilisation de l'instruction `from mon_module import *` est dangereuse car si `mon_module` possède une fonction du même nom qu'une que j'ai déjà définie, alors celle-ci sera écrasée par l'import ! C'est aussi le cas lorsqu'on importe deux modules de cette manière : si certaines fonctions partagent le même nom, les fonctions du dernier import écraseront celles du premier.

Plus tard, vous écrirez vos propres modules que vous importerez.

II Les modules `math` et `random`


Attention

Cette partie est en cours de rédaction.

III Le module `turtle`

Le module Turtle permet de dessiner à l'écran. Sa documentation se trouve ici : `https://docs.python.org/fr/3/library/turtle.html`. Pour comprendre le fonctionnement de Turtle, il faut imaginer un robot sous forme de tortue partant de l'origine d'un repère orthonormé et se déplaçant sur l'écran, en dessinant derrière lui.

III.1 Premiers dessins avec Turtle

Exercice

Exercice 55

Saisir le code ci-dessous et observer le résultat. Modifier le programme pour comprendre le fonctionnement de Turtle.

```
from turtle import *
```

```
forward(100)  # on avance de 100 pixels
left(90)      # on tourne à gauche de 90°
forward(50)   # on avance de 50 pixels
right(90)     # on tourne à droite de 90°
forward(50)   # on avance de 50 pixels
done()        # on indique à Turtle qu'on a fini de dessiner
```

On peut faire en sorte que la tortue soit représentée par une vraie tortue, et pas une simple flèche, avec la commande `shape("turtle")`.

On peut aussi modifier la couleur du trait avec des commandes comme `color("red")` ou `color("blue")`.

Exercice

Exercice 56

Écrire un programme permettant de tracer un carré bleu de côté 150 pixels.

Exercice

Exercice 57

Écrire un programme dessinant un triangle équilatéral vert de côté 200 pixels.

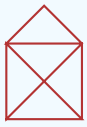
Il est possible d'aller directement en un point en donnant ses coordonnées avec la commande `goto(x, y)`. Heureusement, la tortue peut se déplacer sans écrire, en « levant le crayon » grâce à l'instruction `penup()`. Pour écrire de nouveau, il faut utiliser l'instruction `pendown()`. Par exemple, le code ci-dessous place la tortue au point de coordonnées $(-100\,;50)$ puis trace un segment de 200 pixels de long.

```
penup()          # on lève le crayon
goto(-100, 50)   # on se place au point de coordonnées (-100, 50)
pendown()        # on pose le crayon
forward(200)     # on trace un segment de 200 pixels
done()           # on indique à Turtle qu'on a fini de dessiner
```

Exercice

Exercice 58

Dessiner la forme suivante avec Turtle :



III.2 Des boucles pour des formes plus complexes

Exercice

Exercice 59

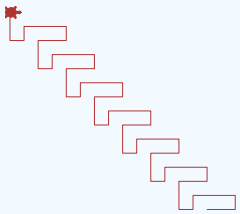
Compléter le programme suivant pour qu'il trace un escalier.

```
for i in range(10):
    forward(20)
    left(90)
    ...
done()
```

Exercice

Exercice 60

Modifier le programme précédent pour qu'il trace la forme ci-contre.



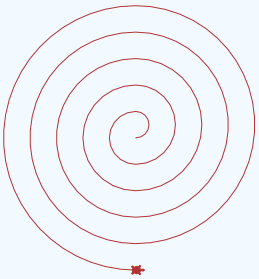
Info

On peut tracer des arcs de cercles avec l'instruction `circle(rayon, angle)`.

Exercice

Exercice 61

Écrire un programme qui trace la spirale ci-contre.



Exercice

Exercice 62

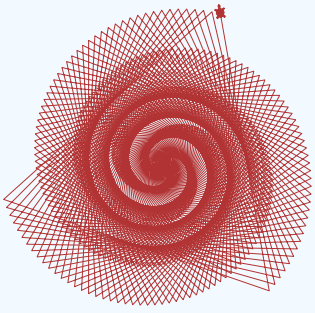
Écrire le code suivant et comprendre ce qu'il trace.

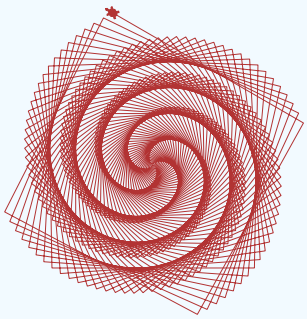
```
speed("fastest")
for i in range(300):
    left(119)
    forward(i)
done()
```

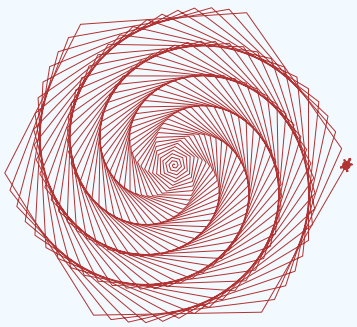
Exercice

Exercice 63

Modifier le code de l'exercice précédent pour obtenir les figures ci-dessous.







III.3 Tracer une fonction avec Turtle

Dans cette partie, on cherche à tracer la représentation graphique d'une fonction avec Turtle. On utilisera la fonction $f : x \mapsto \frac{1}{10}x^2 - 3$, et la fenêtre de visualisation sera réglée avec les paramètres suivants : `xmin = -10`, `xmax = 10`, `ymin = -5` et `ymax = 6`.

La fonction f peut être définie comme ceci en Python :

```
def f(x):
    return x ** 2 / 10 - 3
```

Exercice 64

Compléter le code ci-dessous pour qu'il trace la courbe de la fonction f.

```
xmin = -10
xmax = 10
ymin = -5
ymax = 6

# on ajuste les coordonnées de la fenêtre
setworldcoordinates(xmin, ymin, xmax, ymax)

# la tortue va vite !
speed("fastest")

# on va au début de la courbe, sans tracer
penup()
goto(...)
pendown()

# quel pas ?
pas = 0.1

x = xmin
while x < xmax:
    x = x + ...
    goto(...)

done()
```

Exercice 65

Modifier le programme précédent pour ajouter des axes et des graduations.