Corrigé

Graphisme « tortue »

Exercice 1. Script pour dessiner un carré de côté 200 :

```
reset()
forward(200)
left(90)
forward(200)
left(90)
forward(200)
left(90)
forward(200)
```

Exercice 2. On définit la fonction suivante :

```
def carre(l):
    forward(l)
    left(90)
    forward(l)
    left(90)
    forward(l)
    left(90)
    forward(l)
    left(90)
```

Notez que le dernier `left` permet de repositionner l'orientation de la tortue telle qu'elle était avant le tracé du carré. Cette fonction permet de réaliser le dessin demandé à l'aide du script suivant :

```
reset()
carre(50)
left(90)
carre(100)
left(90)
carre(150)
left(90)
carre(200)
```

Exercice 3. Le dessin demandé se réalise à l'aide du script :

```
reset()
for i in range(20, 220, 20):
    carre(i)
```

Exercice 4. Le script qui suit réalise le dessin présenté figure 1.

```
reset()
speed(0)
for i in range(1, 501):
    forward(i)
    left(91)
```

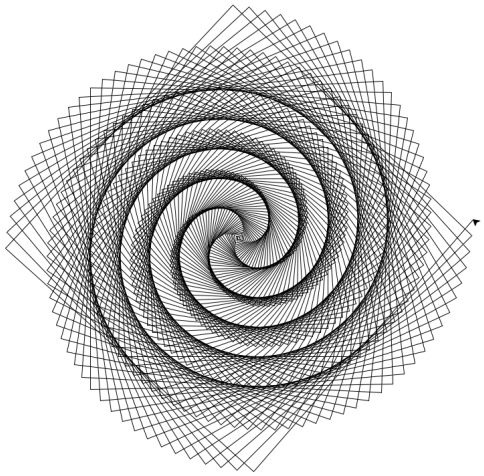

Figure 1 – Le résultat du script de l'exercice 4.

Exercice 5. On définit la fonction suivante, puis on réalise le script :

```
def polygone(n, l):
    a = 360 / n
    for _ in range(n):
        forward(l)
        left(a)
```

```
reset()
for n in range(3, 8):
    polygone(n, 100)
```

Exercice 6. Le dessin demandé se réalise à l'aide du script :

```
reset()
circle(150)
circle(150, extent=720, steps=5)
```

Exercice 7. On réalise un dégradé de couleur entre le bleu et le rouge en utilisant les composantes RGB :

```
reset()
bgcolor('black')
speed(0)
for i in range(72):
    pencolor(i/71, 0, 1-i/71)
    circle(100, extent=180)
    left(90)
    penup()
    forward(200)
    left(85)
    pendown()
```

Exercice 8. On définit successivement les fonctions :

```
def sierp1(l):
    begin_fill()
    polygone(3, l)
    end_fill()
```

```
def sierp2(l):
    sierp1(l/2)
    left(60)
    forward(l/2)
    right(60)
    sierp1(l/2)
    right(60)
    forward(l/2)
    left(60)
    sierp1(l/2)
    backward(l/2)
```

```
def sierp3(l):
    sierp2(l/2)
    left(60)
    forward(l/2)
    right(60)
    sierp2(l/2)
    right(60)
    forward(l/2)
    left(60)
    sierp2(l/2)
    backward(l/2)
```

Exercice 9. La définition récursive demandée est la suivante :

```
def sierpinski(n, l):
    if n == 1:
        begin_fill()
        polygone(3, l)
        end_fill()
    else:
        sierpinski(n-1, l/2)
        left(60)
        forward(l/2)
        right(60)
        sierpinski(n-1, l/2)
        right(60)
        forward(l/2)
        left(60)
        sierpinski(n-1, l/2)
        backward(l/2)
```

Le script suivant réalise alors le tracé de la 5e génération du triangle de Sierpiński :

```
reset()
speed(0)
sierpinski(5, 512)
```

Exercice 10. De même, on définit la fonction :

```
def bin_tree(n, l):
    pensize(n)
    if n == 1:
        pencolor('green')
        forward(l/3)
        backward(l/3)
    else:
        pencolor('brown')
        forward(l/3)
        left(30)
        bin_tree(n-1, 2*l/3)
        right(60)
        bin_tree(n-1, 2*l/3)
        left(30)
        penup()
        backward(l/3)
        pendown()
```

(Les branches les plus fines sont dessinées en vert, les autres en brun.)
Le script demandé s'écrit alors :

```
reset()
speed(0)
left(90)
bin_tree(8, 400)
```

Exercice 11. On commence par placer les deux tortues dans leurs positions initiales en choisissant la couleur de leurs traces respectives :

```
Alice.reset()
Bob.reset()

Alice.pencolor('red')
Bob.pencolor('blue')
```

On les déplace à leurs positions de départ respectives :

```
Alice.penup()
Alice.setposition(0, 400)
Alice.pendown()
Bob.penup()
Bob.setposition(-400, -100)
Bob.pendown()
```

La course rectiligne (appelée encore « courbe du chien ») s'obtient alors à l'aide du script :

```
while Alice.distance(Bob) > 2:
    Bob.forward(10)
    Alice.setheading(Alice.towards(Bob))
    Alice.forward(12)
```

La vitesse légèrement plus grande d'Alice permet à cette dernière de rattraper Bob.
La course circulaire se réalise par le script :

```
Bob.penup()
Bob.setposition(0, -400)
Bob.pendown()

while Alice.distance(Bob) > 2:
    Bob.circle(400, extent=5)
    Alice.setheading(Alice.towards(Bob))
    Alice.forward(25)
```

L'expérience montre que si la vitesse d'Alice est inférieure à celle de Bob (dans le cas des valeurs numériques choisies cela revient à avancer de moins de 35 pas environ), la trajectoire d'Alice tend à être un cercle de diamètre inférieur à celle de Bob.

Enfin, la course à trois (que l'on connait aussi sous le nom de « problème des souris ») peut se réaliser ainsi :

```
Carole = Turtle()

Alice.reset()
Bob.reset()
Carole.reset()

Alice.pencolor('red')
Bob.pencolor('blue')
Carole.pencolor('green')

Alice.penup()
Alice.left(60)
Alice.forward(400)
Alice.pendown()
Bob.penup()
Bob.backward(400)
Bob.pendown()
Carole.penup()
Carole.right(60)
Carole.forward(400)
Carole.pendown()

while Alice.distance(Bob) > 5:
    Alice.setheading(Alice.towards(Bob))
    Bob.setheading(Bob.towards(Carole))
    Carole.setheading(Carole.towards(Alice))
    Alice.forward(5)
    Bob.forward(5)
    Carole.forward(5)
```