Exercice 1

[Programme à réaliser dans l'environnement `turtle`]
Ecrire une fonction qui affiche un carré dont la longueur du côté est passée en paramètre.

Corrigé

```
# Version sans boucle:
def carre(l):
    forward(l) ; left(90)
    forward(l) ; left(90)
    forward(l) ; left(90)
    forward(l) ; left(90)

# Version avec boucle:
def carreb(l):
    for i in range(4):
        forward(l)
        left(90)
```

Exercice 2

[Programme à réaliser dans l'environnement `turtle`]
Ecrire une fonction qui affiche un rectangle dont la grande longueur est passée en paramètre, ainsi que le rapport entre la grande longueur et la petite.

Corrigé

```
def rectangle(long,rap):
  larg = long/rap
  fd(long)
  lt(90)
  fd(larg)
  lt(90)
  fd(long)
  lt(90)
  fd(larg)
```

Exercice 3

Ecrire une fonction qui dessine une rangée de carrés juxtaposés, dont le côté est passé en paramètre.

Corrigé

```
def carre(long):
  c = 0
  while (c < 4):
    forward(long)
    left(90)
    c = c + 1
```

```
# Hyp: une rangée a 3 carrés
# Version sans boucle
def rangee_3(long):
  carre(long)
  penup()
  forward(1.2 * long)
  pendown()
  carre(long)
  penup()
  forward(1.2 * long)
  pendown()
  carre(long)
```

```
# Le nombre de carrés de la rangée
# est un paramètre
# (boucle obligatoire)
def rangee(long,n):
  c = 0
  while (c < n):
    carre(long)
    penup()
    fd(long+10)
    pendown()
    c = c + 1
```

Exercice 4

Ecrire une fonction qui dessine une spirale en dessinant 10 demi-cercles successifs, dont le rayon est augmenté du même paramètre à chaque demi-tour.

Corrigé

```
# 10 demi-cercles, paramètre : écart
def spirale(delta):
  circle(10+0*delta,180)
  circle(10+1*delta,180)
  circle(10+2*delta,180)
  circle(10+3*delta,180)
  circle(10+4*delta,180)
  circle(10+5*delta,180)
  circle(10+6*delta,180)
  circle(10+7*delta,180)
  circle(10+8*delta,180)
  circle(10+9*delta,180)
```

```
  # on peut ajouter un paramètre
  # qui serait le point de départ
  def spirale(init,delta):
    circle(init+0*delta,180)
    circle(init+1*delta,180)
    circle(init+2*delta,180)
    circle(init+3*delta,180)
    circle(init+4*delta,180)
    circle(init+5*delta,180)
    circle(init+6*delta,180)
    circle(init+7*delta,180)
    circle(init+8*delta,180)
    circle(init+9*delta,180)
```

```
# et bien sûr on peut utiliser une boucle avec compteur:
# à condition d'utiliser la valeur du compteur:
def spirale(init,delta,max):
    c = 0
    while (c < max):
        circle(init+c*delta, 180)
    c += 1
```

Exercice 5

Pour chacun des cas suivants, déjà vus en exercice, proposer une fonction avec les paramètres pertinents :

- dessiner un cerf-volant
- dessiner une grille de 9 carrés
- dessiner une croix (type croix rouge)
- dessiner un polygone

Corrigé

```
def cerf_volant():
    left(60)
   forward(100)
   left(60)
   forward(100)
   left(90+30)
   forward(100)
   left(60)
   forward(100)
   left(90+60)
   forward(170)
   penup()
   backward(170/2)
   right(90)
   pendown()
   forward(50)
   backward(2*50)
```

```
def cerf_volant(long):
   left(60)
   forward(long)
   left(60)
   forward(long)
   left(90+30)
   forward(long)
   left(60)
   forward(long)
   left(90+60)
   forward(1.72*long)
   penup()
   backward(1.72*long/2)
   right(90)
   pendown()
   forward(long/2)
   backward(long)
```

```
def carre(long):
   c = 0
      while (c < 4):
         forward(long)
         left(90)
         c = c + 1
```

```
def rangee(long):
   c = 0
   while (c < 3):
      carre(long)
      penup()
      fd(1.2*long)
      pendown()
      c = c + 1
```

```
def grille(long):
   c = 0
      while (c < 3):
        rangee(long)
        penup()
        backward(3*1.2*long)
        right(90)
        fd(1.2*long)
        left(90)
        pendown()
        c += 1
      ht()
```

```
def branche():
    lt(90)
    fd(100)
    rt(90)
    fd(100)
    rt(90)
    fd(100)
```

```
def croix_rouge():
   color("white", "red")
   begin_fill()
   branche()
   branche()
   branche()
   branche()
   end_fill()
```

```
# polygone

def polygone(n,l):
  c = 0
  while (c < n):
    fd(l)
    rt(360/n)
    c = c + 1
```