Solution de l'exercice No 1. *[Programme à réaliser dans l'environnement `turtle`]*
Ecrire un programme qui affiche un carré de 100 de côté. Proposer une version sans boucle et une version avec boucle.

```
from turtle import *

forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
```

Version avec boucle :

```
for i in range(4):
   forward(100)
   left(90)
```

Solution de l'exercice No 2. *[Programme à réaliser dans l'environnement `turtle`]*
Ecrire une fonction qui affiche un carré dont la longueur du côté est passée en paramètre.

```
from turtle import *

def carre(l):
   "Affiche un carré de côté l"
   for i in range(4):
      forward(l)
      left(90)
```

Solution de l'exercice No 3. *[Programme à réaliser dans l'environnement `turtle`]*
Ecrire une fonction qui affiche un polygone dont le nombre de côtés, et la longueur de chaque côté sont passés en paramètre.

```
from turtle import *

def polygone(n,l):
   "Affiche un polygone de n côtés, de longueur l"
   for i in range(n):
      forward(l)
      left(360/n)
```

Solution de l'exercice No 4. *[Programme à réaliser dans l'environnement `turtle`]*
Ecrire une fonction qui affiche un triangle équilatéral dont la longueur du côté est passée en paramètre.

```
from turtle import *

def triangle(l):
   "Affiche un triangle équilatéral de côté l"
   for i in range(3):
      forward(l)
      left(360/3)
```

Solution de l'exercice No 5. *[Programme à réaliser dans l'environnement `turtle`]*
Ecrire une fonction qui dessine une étoile à 5 branches.
On peut remarquer que dessiner une étoile à 5 branches revient à passer par tous les sommets d'un pentagone mais en les parcourant de 2 en 2 (ou de 3 en 3) au lieu de les parcourir consécutivement.

```
from turtle import *

def étoile():
   for i in range(5):
      forward(100)
      right(2*360/5)
```

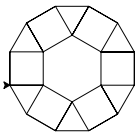
Solution de l'exercice No 6. *[Programme à réaliser dans l'environnement `turtle`] Ecrire une fonction qui dessine une étoile à 6 branches en plaçant judicieusement deux triangles équilatéraux.*
Une fois qu'on a dessiné le premier triangle, il faut non seulement « lever le crayon » pour se positionner plus haut, mais aussi choisir l'angle dans lequel on va se placer pour dessiner le second triangle. Il n'était pas nécessaire de définir une autre fonction triangle qui dessine dans l'autre sens.

```
from turtle import *

def triangle(l):
   "Affiche un triangle équilatéral de côté l"
   for i in range(3):
      forward(l)
      left(360/3)

# Dessin du premier triangle
triangle(100)
# positionnement pour dessiner le 2e triangle
left(90)    # maintenant on est oriente vers le haut
penup()     # on leve le crayon
forward(60) # on avance de 60
right(150)  # on tourne pour etre oriente vers le bas
pendown()   # on repose le crayon
# Dessin du 2e triangle
triangle(100)
```

Solution de l'exercice No 7. *[Programme à réaliser dans l'environnement `turtle`]*
Ecrire une fonction qui dessine une spirale ayant 6 côtés (spirale hexagonale).

```
from turtle import *
angle = 360/6
for i in range(40):
   forward(10+i*3)
   left(angle)
```

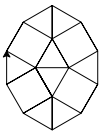
Solution de l'exercice No 8. *[Programme à réaliser dans l'environnement `turtle`]*
Ecrire un programme qui dessine la figure suivante (indice : commencer par dessiner une "maison").



```
from turtle import *

def carre():
```

```
   for i in range(4):
      forward(50)
      right(90)

def maison():
   carre()
   forward(50)
   right(30)
   forward(50)
   right(120)
   forward(50)

def diamant():
   for i in range(6):
      left(90)
      maison()
      backward(50)

diamant()
```

Solution de l'exercice No 9. *[Programme à réaliser dans l'environnement* `turtle`*]*
Ecrire un programme qui dessine la figure suivante (indice : commencer par dessiner une "maison").



La solution proposée ici n'est pas la plus élégante.

```
from turtle import *

def carre():
   for i in range(4):
      forward(50)
      right(90)

def maison():
   carre()
   forward(50)
   right(30)
   forward(50)
   right(120)
   forward(50)

def diamant_l():
   left(60)
   maison()
   right(30)
   forward(50)
   left(60)
   maison()
   right(120)
   forward(50)
   left(90)
   forward(50)
   right(120)
   forward(50)
   left(120)
   maison()
```

```
    right(30)
    forward(50)
    right(30)
    forward(50)
    right(30)
    maison()
    left(30)
    forward(50)
    left(90)
    forward(50)
    right(120)
    forward(50)

diamant_l()
```