

Boucle for

Reprenons la fonction `dessine_carre(cote)` où les deux instructions `forward(cote)` et `left(90)` sont **répétées quatre fois**

```
def dessine_carre(cote):  
  
    forward(cote)  
    left(90)  
  
    forward(cote)  
    left(90)  
  
    forward(cote)  
    left(90)  
  
    forward(cote)  
    left(90)
```

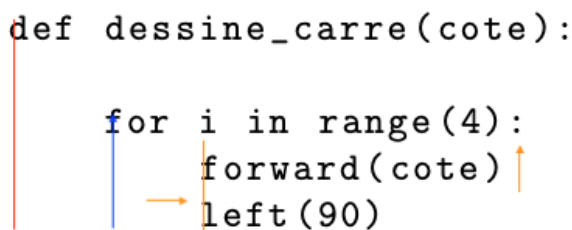
on peut réécrire cette fonction ainsi

```
def dessine_carre(cote):  
  
    for i in range(4):  
        forward(cote)  
        left(90)
```

Regardons un peu en détail le code

1 Répétition d'un nombre fixé d'instructions

```
def dessine_carre(cote):  
    for i in range(4):  
        forward(cote)  
        left(90)
```



1. Les instructions répétées forment le **bloc de la boucle** délimité par les `:` et l'indentation (ligne orange)
2. On peut traduire `for i in range(4)` par **Répéter 4 fois**
3. La variable `i` est appelé un **compteur de boucle** qui parcourant ici l'ensemble des entiers $\{0,1,2,3\}$ crée 4 répétitions
L'ensemble des entiers $\{0,1,2,3\}$ est noté aussi `[[0;4[` **la valeur 4 étant exclue**
4. Puisque le compteur de boucle n'est pas utilisé dans le bloc de la boucle pour insister que ce n'est qu'une répétition sans utilisation du compteur de boucle dans la boucle, à la place d'un nom de variable on met parfois un caractère underscore

```
def dessine_carre(cote):
    for _ in range(4):
        forward(cote)
        left(90)
```

On voit l'importance du symbole : et de l'indentation pour marquer le bloc de répétitions. Dans d'autres langages, comme Javascript, ce sont des accolades qui délimitent le bloc de répétition :

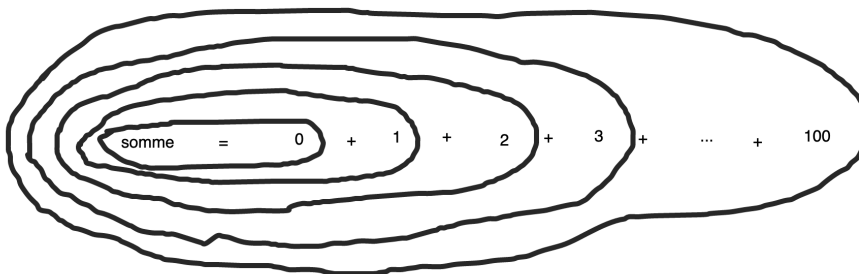
```
for(var i = 0; i < 4; i++){
    /* Le compteur de la boucle
    est la variable i (répéter 4 fois)
    Le corps de la boucle de répétition
    est entre accolades, */
}
```

2 Utilisation du compteur de boucle dans la boucle

Lorsque les actions répétées "varient avec le compteur de boucle", dans ce cas là le compteur de boucle apparaît dans le bloc de la boucle

Exemple 1 :

On veut calculer la somme $1 + 2 + 3 + \dots + 100$



On utilise une variable `somme` initialisée à 0 en plus du compteur de boucle `i` variant de 1 à 100 et on répète l'instruction `somme = somme + i` pour `i` variant de 1 à 100

On généralise de suite en définissant une fonction `somme_entiers(n)` qui calcule la somme $1 + 2 + 3 + \dots + n$ pour $n \geq 1$

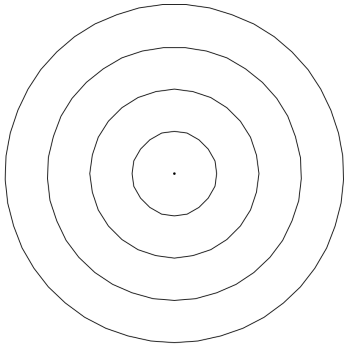
```
def somme_entiers(n):
    somme = 0
    for i in range(1,n):
        somme = somme + i
    return somme
```

`range(1,n)` est associé à l'intervalle d'entiers $\llbracket 1;n \llbracket$ où n est exclu

Exemple 2 :

Construction d'une cible :

On veut construire 4 cercles concentriques de centre (0,0) et de rayon 50,100,150,200



Si le compteur de boucle est désigné par i alors le rayon vaut $50*i$ pour i variant de 1 à 4 d'où la fonction

```
from turtle import *

def dessine_cercle(rayon):
    pu()
    goto(0,-rayon)
    pd()
    circle(rayon)

def dessine_cible():
    ht()
    dot(3)
    for i in range(1,5):
        dessine_cercle(50*i)
```

On peut dessiner les cercles dans l'ordre inverse du plus grand au plus petit

```
from turtle import *

def dessine_cercle(rayon):
    pu()
    goto(0,-rayon)
    pd()
    circle(rayon)

def dessine_cible():
    ht()
    dot(3)
    for i in range(4,0,-1):
        dessine_cercle(50*i)
```

`range(4,0,-1)` est associé à l'intervalle d'entiers $\llbracket 4;0 \rrbracket$ où 1 est exclu mais on n'écrit pas en général les intervalles en mathématiques de cette façon, mais ici c'est pratique de le voir ainsi

-1 signifie que l'on passe d'un entier au suivant en retranchant 1

3 Exercices

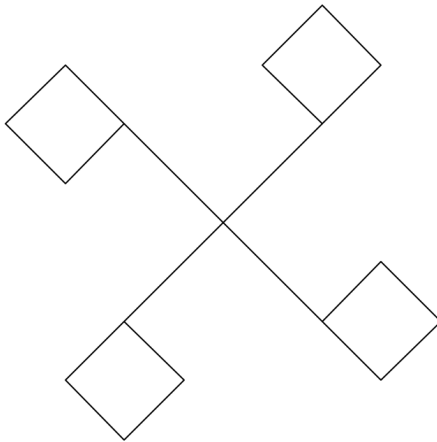
Ex 1

Réécrire la fonction suivante avec une boucle for

```
from turtle import *
def dessin():
    forward(70)
    left(216)
    forward(70)
    left(216)
    forward(70)
    left(216)
    forward(70)
    left(216)
    forward(70)
    left(216)
    forward(70)
    left(216)
```

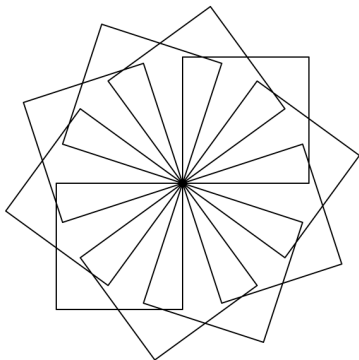
Ex 2

Dessiner un "moulin" avec la tortue :



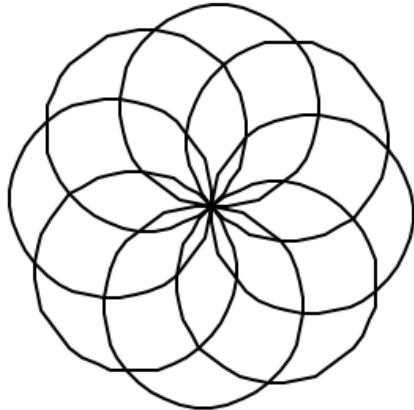
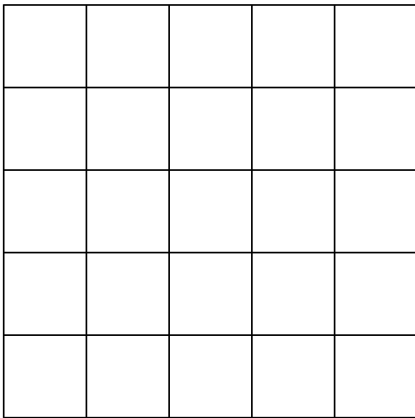
Ex 3

Dessiner une "rosace" par la tortue (faire "tourner" un carré sur son coin inférieur gauche) :



Ex 4

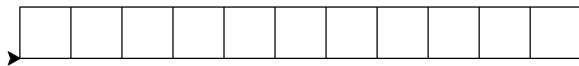
Dessiner une vraie "rosace" en faisant tourner un cercle

**Ex 5**

Dessiner une grille $n \times n$ en procédant ainsi :

1. En premier définir une fonction `dessine_ligne(n,cote)` qui dessine une ligne de carrés de côté `cote` juxtaposés sur une ligne

A la fin du dessin la tortue revient dans son état initial (coordonnées et angle)



2. En réutilisant la fonction précédente définir `dessine_grille(n,cote)` qui dessine une grille $n \times n$ de carrés de côté `cote`

Ex 6

Définir une fonction `somme_carres(n)` qui calcule la somme $1^2 + 2^2 + \dots + n^2$ pour $n \geq 1$

Ex 7

Définir une fonction `produit_entiers(n)` qui calcule le produit $1 \times 2 \times \dots \times n$ pour $n \geq 1$