

On voit dans ce chapitre comment créer des boucles avec l'opérateur for.

1- STRUCTURE : `for i in range(...)`

Code python :

```
1 for i in range(3,13,2) :  
2     print(i)
```

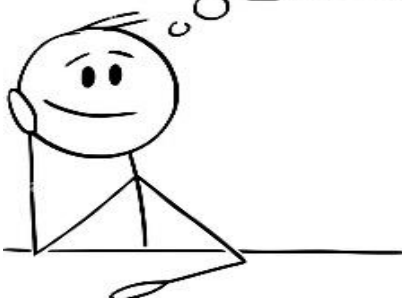
Console après exécution de ce code :

```
>>> (executing file "cours.py")  
3  
5  
7  
9  
11
```

Point Cours : Syntaxe d'une boucle « Pour » en python :

```
for i in range(3,13,2) :  
    print(i)
```

*Doit-on toujours
mettre 3 arguments
dans la parenthèse
du range() ?*



On peut écrire uniquement 1 ou 2 arguments dans la parenthèse du range().

Par exemple :

```
1 for i in range(3,13) :  
2     print(i)
```

Donne à l'exécution :

Autre exemple, le code :

```
1 for i in range(13) :  
2     print(i)
```

Donne à l'exécution :



2- EXEMPLE 1 : que fait ce code python ?

Code :

```
1 n = input("Afficher la table de : ")  
2 n = int(n)  
3 for i in range(1,11) :  
4     produit = i*n  
5     message = str(i) + " x " + str(n) + " = " + str(produit)  
6     print(message)
```

Exécution dans la console si la valeur saisie est par exemple 5 : Pour bien comprendre ce qui se passe lors de l'exécution, on peut construire un tableau qui indique la valeur des variables au cours de l'exécution précédente :


n	i	produit	message

3- EXEMPLE 2 : que fait ce code python ?

Code :

```
1 nbrEleves = int(input("Combien d'élèves ? : "))
2 somme = 0
3 for i in range(nbrEleves) :
4     message = "note de l'élève " + str(i+1) + " : "
5     note = input(message)
6     note = float(note)
7     somme = somme + note
8
9 moy = somme / nbrEleves
10 print(f"La moyenne est : {moy}")
```

Exemple d'exécution dans la console :



*Faut donc toujours
se mettre à la place
du processeur qui
exécute le code ...*

```
>>> (executing file "cours.py")
Combien d'élèves ? : 4
note de l'élève 1 : 18
note de l'élève 2 : 12
note de l'élève 3 : 6
note de l'élève 4 : 20
La moyenne est : 14.0
```

Pour bien comprendre ce qui se passe lors de l'exécution, on peut construire un tableau qui indique la valeur des variables au cours de l'exécution précédente :

nbrEleves	somme	i	message	note	moy

Point Cours : `a = 10` : une variable nommée a est créée. Sa valeur est initialisée à 10 .

`a = a + 2` : la valeur de cette variable a est remplacée par son ancienne augmentée de 2, soit ici 12.

`b = a + 2` : une variable nommée b est créée. Sa valeur est celle de la variable a augmentée de 2. La valeur de a reste à 2.

4- BIBLIOTHEQUE RANDOM :

Point Cours : En informatique, on a souvent besoin de réaliser des tirages au sort. Des fonctions python permettent de générer des nombres aléatoires. Ces fonctions ne sont pas natives de Python. Elles sont disponibles dans une *bibliothèque* nommée « random ».

- Pour importer la bibliothèque :

```
from random import randint , random
```

- Pour générer un nombre **entier** entre 1 et 5 inclus :

```
randint(1,5)
```

- Pour générer un nombre **réel** entre 0 et 1 inclus : `random()`

5- EXERCICE 1 : *Encore une histoire de moyenne ...*

Le code qui suit est incomplet. En l'exécutant on obtient dans la console :

```
>>> (executing file "cours.py")
Saisir un nombre entier : 4
nombre 1 : -4178
nombre 2 : 7095
nombre 3 : 488
nombre 4 : 2513
La moyenne de ces nombres aléatoires,
tous compris entre -10000 et 10000,
est 1479.5
```

⇒ Compléter ce code :

```
from random import randint , random
```

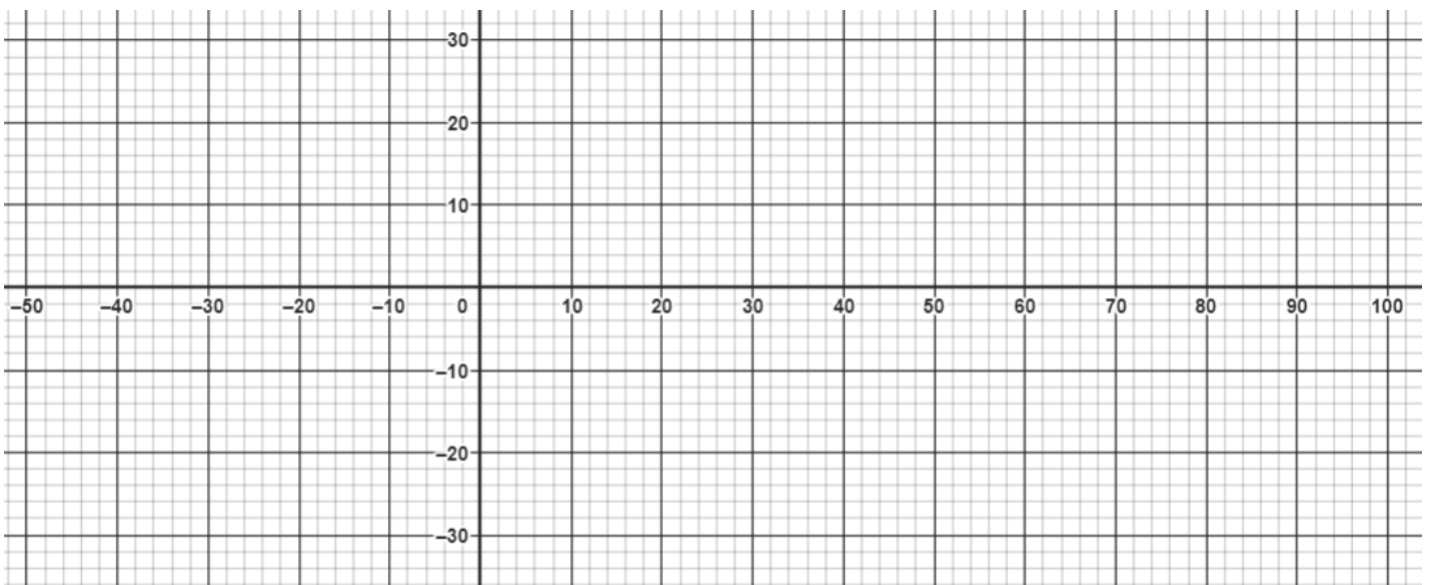
```
print(f"La moyenne de ces nombres aléatoires,\ntous  
compris entre -10000 et 10000,\nest {moy}")
```

6- EXERCICE 2 : *Un peu de graphisme en plus, c'est mieux*

Le code ci-contre permet de tracer une forme géométrique sur une fenêtre graphique « *turtle* ».

⇒ Si cette fenêtre est celle donnée ci-dessous, tracer cette forme :

```
1 from turtle import *
2 up()
3 goto(-50,-25)
4 fillcolor("red")
5 begin_fill()
6
7 for i in range(4) :
8     forward(50)
9     left(90)
10 end_fill()
11
12 mainloop()
```



7- BOUCLE DE BOUCLE : C'est possible ?

On peut tout à fait imbriquer plusieurs boucles les unes dans les autres. Par exemple, dans le code donné ci-contre, on a une boucle qui incrémente une variable i qui se trouve à l'intérieur d'une boucle qui incrémente une variable j . Cela fonctionne parfaitement, il suffit que les 2 indices de boucles aient des noms différents.

Dans ce code, on exécute le script ci-dessous :

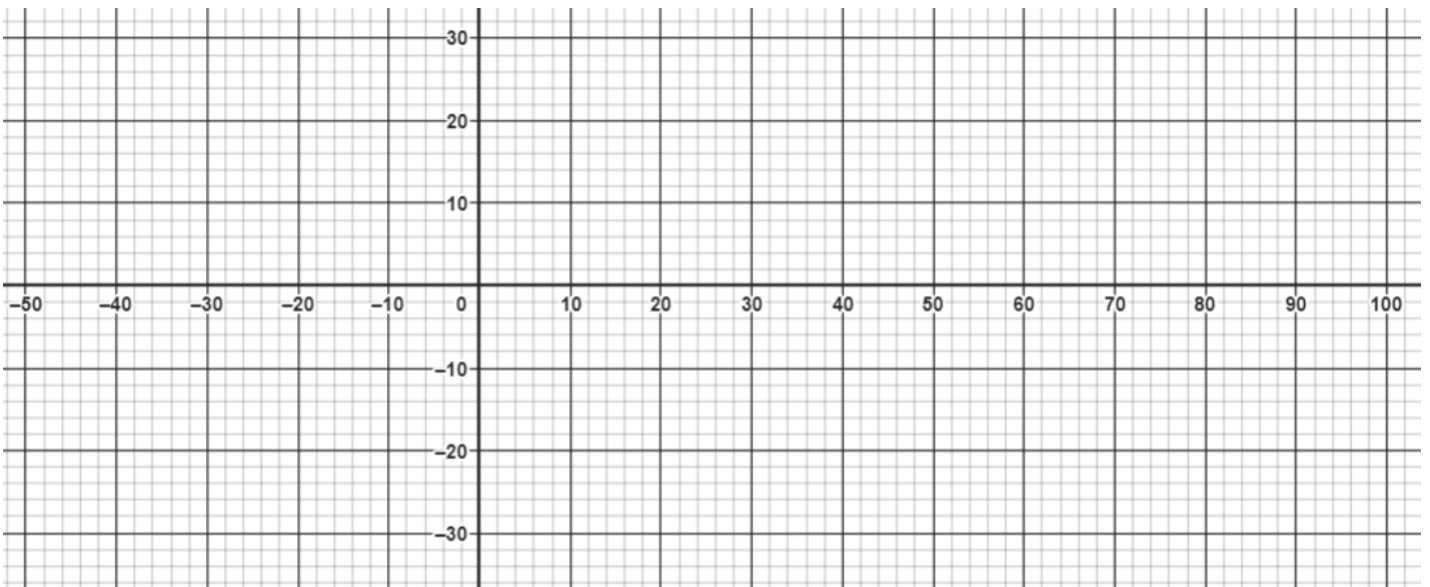
- une première fois avec la variable $j = 0$

- Une seconde fois avec la variable $j = 1$

```
down()
begin_fill()
for i in range(4) :
    forward(50)
    left(90)
end_fill()
up()
forward(100)
```

```
1 from turtle import *
2 up()
3 goto(-50, -25)
4 fillcolor("red")
5
6 for j in range(2) :
7     down()
8     begin_fill()
9     for i in range(4) :
10        forward(50)
11        left(90)
12    end_fill()
13    up()
14    forward(100)
15
16 mainloop()
```

⇒ Si la fenêtre turtle est celle donnée ci-dessous, tracer les formes obtenues lorsque le code est exécutée :



8- PARCOURIR UNE CHAÎNE DE CARACTÈRE :

Point Cours : Syntaxe d'une boucle « Pour » en python :

```
for lettre in "bonjour" :  
    print(lettre)
```

La variable *lettre* prendra successivement comme valeur celle de chacun des caractères du string "bonjour".

Exemple de code :

```
1 aLenvers = ""  
2 for lettre in "bonjour" :  
3     aLenvers = lettre + aLenvers  
4  
5 print(aLenvers)
```

Valeur des variables au cours de l'exécution :

lettre	aLenvers

Affichage dans la console :