Les structures répétitives
La boucle while
La boucle for

Le boucle While

- Comme dans beaucoup d'autres langages, la boucle **while** exécute un bloc tant que la condition d'arrêt est vérifiée.
- Syntaxe:

```
Tanque(condition) faire
        <instructions>
Fintanque
```

Algorithmique

```
while <condition> :
    <instructions>
```

Python

Exemple

```
"""
Le programme ajoute les nombres entiers compris
entre 1 à 20 dans la variable s

"""
s=0
i=0
while (i<20):
    i=i+1
    print(s,"+",i)
    s=s+i

print("La somme est : ",s)
```

```
0 + 1
1 + 2
3 + 3
6 + 4
10 + 5
15 + 6
21 + 7
28 + 8
36 + 9
45 + 10
55 + 11
66 + 12
78 + 13
91 + 14
105 + 15
120 + 16
136 + 17
153 + 18
171 + 19
190 + 20
La somme est :  210
```

Remarques

- La variable évaluée dans la condition doit exister au préalable (il faut qu'on lui ait déjà affecté au moins une valeur).
- Si la condition est fausse au départ, le corps de la boucle n'est jamais exécuté.
- Si la condition reste toujours vraie, alors le corps de la boucle est répété indéfiniment

→ Il faut donc veiller à ce que le corps de la boucle contienne au moins une instruction qui change la valeur d'une variable intervenant dans la condition évaluée par while, de manière à ce que cette condition puisse devenir fausse et la boucle se terminer.

Remarque: break et continue

- Ces deux instructions permettent à l'utilisateur d'avoir un plus grand contrôle de la boucle.
- L'instruction break provoque une ***sortie brutale de la boucle, mais le programme*** continue son exécution après la boucle
- L'instruction continue permet **de passer à l'itération suivante**

```
while <condition 1> :
    --- instructions diverses ---
    if <condition 2> :
        break
```

Exemple (break)

```
"""
Le programme ajoute les nombres entiers compris
entre 1 à 20 dans la variable s mais dès que la somme
dépasse 100 on sort de la boucle

"""
s=0
i=0
while (i<20):
    i=i+1
    s=s+i
    if (s>=100):
        break

print("Le nombre est : ",i)
print("La somme est : ",s)
```

```
Le nombre est :  14
La somme est :  105
```

Exemple (continue)

```
"""
Le programme ajoute à la variable s tous les entiers compris
entre 1 à 20 sauf les valeurs 10 et 11. L'instruction continue
est exécutée lorsque la valeur de la variable i devient 10 ou 11.

"""
s=0
i=0
while (i<20):
    i=i+1
    if (i==10 or i==11):
        continue
    s=s+i
    print("i=",i)
print("La valeur du compteur est : ",i)
print("La somme est : ",s)
```

```
i= 1
i= 2
i= 3
i= 4
i= 5
i= 6
i= 7
i= 8
i= 9
i= 12
i= 13
i= 14
i= 15
i= 16
i= 17
i= 18
i= 19
i= 20
La valeur du compteur est :  20
La somme est :  189
```

Conclusion

L’instruction **continue** termine une itération, alors que le

l’instruction **break** termine une boucle toute entière

Remarque

- Dans de nombreux langages, il existe une instruction **do…while (Repeter … jusqu'à)** qui permet de créer une boucle pour laquelle on ne connaît pas à l'avance le nombre de répétition, mais qui doit s'exécuter au moins une fois.
- Cette instruction n'existe pas en Python, mais on peut facilement reproduire son fonctionnement de la façon suivante :

Exemple

```
while (True):
    x=int(input("Entrez un entiez positif non nul : "))
    if (x>0):
        break
print("saisie correcte")
```

```
>>>
= RESTART: /Users/oumaira/Documents/Python 2019/cours 2020/exemples/break.py =
Entrez un entiez positif non nul : 0
Entrez un entiez positif non nul : -12
Entrez un entiez positif non nul : -3
Entrez un entiez positif non nul : 12
saisie correcte
```

Exercice

- Un entier n est toujours divisible par 1 et n, qui sont ses ***diviseurs triviaux. Par exemple 12 est divisible par 1 et 12 (et par d'autres)...***
- **Un entier n est premier s'il est ≥ 2 et si ses seuls diviseurs sont les diviseurs triviaux. Par exemple 13 est premier, mais pas 12.**
- ALGORITHME.
  - Pour savoir si un nombre n ≥ 2 est premier, il suffit donc d'examiner les nombres entiers d de [2,n-1] à la recherche d'un diviseur de n. Si l'on en trouve un, on s'arrête au premier trouvé avec le résultat False. Sinon, le résultat sera True

Solution

```
n=int(input('Entrer une valeur : '))
premier=True
if n>=2:
    d = 2
    while d < n :
        if n%d == 0 :
            premier=False
            break
        else:
            d = d + 1
    if premier==True :
        print('nombre premier')
    else:
        print('nombre non premier')
else:
    print('le nombre doit être supérieur strictement à 1')
```

La boucle for

- **Syntaxe**

```
for i in range(imin,imax+1,pas):
        instructions
```

La valeur maximale de i sera imax

Exemple

```
x=int(input("Entrer un nombre : "))
for i in range(1,11):
    print(i, '*', x ," = ", i*x)
print("c'est terminé!")
```

```
Entrer un nombre : 9
1 * 9  =  9
2 * 9  =  18
3 * 9  =  27
4 * 9  =  36
5 * 9  =  45
6 * 9  =  54
7 * 9  =  63
8 * 9  =  72
9 * 9  =  81
10 * 9  =  90
c'est terminé!
```

Exemple en mode interactif

```
>>> for i in range(1,11,2):
        print(i)


1
3
5
7
9
```

```
>>> for i in range(10,0,-1):
        print(i)


10
9
8
7
6
5
4
3
2
1
```

Syntaxe générale

```
for élément in séquence :
    bloc d'instructions
# suite du programme
```

Exemple

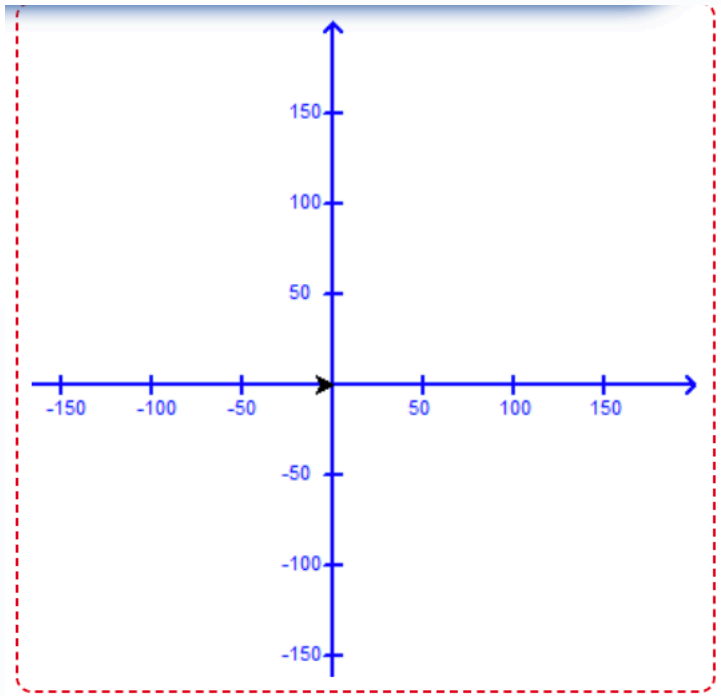
```
ch="ensak"
for i in ch:
    print (i)
print("Fin de la Boucle")
```

```
========== RESTART:
e
n
s
a
k
Fin de la Boucle
>>>
```

Le module turtule

Le module turtule

- Le module graphique turtle permet de piloter un «crayon» afin de tracer dynamiquement des figures géométriques.
- Les dessins sont réalisés dans un repère orthonormé virtuel centré sur la fenêtre d'affichage. L'unité des axes est le pixel. Le repère n'est pas visible à l'écran.
- La forme par défaut du crayon de tracé est une flèche «orientée», placé au départ à l'origine du repère. Le crayon est situé à la pointe, la flèche montre le sens du tracé en cours ou à venir.

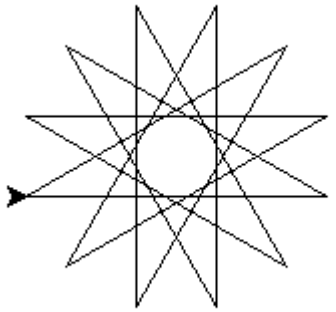


Quelques fonctions du module turtle

- **reset()** On efface tout et on recommence
- **goto(x, y)** Aller a l'endroit de coordonnées x, y
- **forward**(distance) Avancer d'une distance donnée
- **backward**(distance) Reculer
- **up()** Relever le crayon (pour pouvoir avancer sans dessiner)
- **down()** Abaisser le crayon (pour recommencer a dessiner)
- **color(couleur)** *couleur peut être une chaine predefinie ('red', 'blue', etc.)*
- **left(angle)** Tourner a gauche d'un angle donné (exprime en degrés)
- **right(angle)** Tourner a droite
- **width(epaisseur)** Choisir l'épaisseur du trace
- **write(texte)** *texte doit être une chaine de caractères*

Graphiques tortue

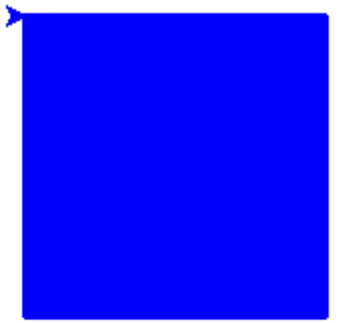
```
from turtle import *
reset()
i=1
while i<=12:
    forward(150)
    left(150)
    i=i+1
```



Graphiques tortue

```
from turtle import *
reset()
color("blue")
width("3")
begin_fill()
for i in range(1,5):
    forward(150)
    right(90)      # Rotation de 90 degré

end_fill()
```



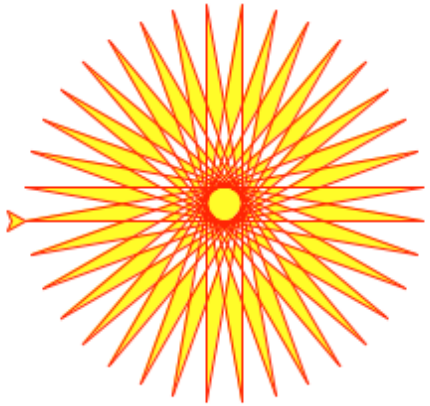
https://docs.python.org/3.3/library/turtle.html?highlight=turtle#turtle.pos

https://zestedesavoir.com/tutoriels/944/a-la-decouverte-de-turtle/

Exemple

Turtle star

Turtle can draw intricate shapes using programs that repeat simple moves.



```
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill()
done()
```