Les Boucles:

- Une boucle permet d'exécuter plusieurs fois de suite une même séquence d'instructions.
- Cette ensemble d'instructions s'appelle le **corps de la boucle.**
- Chaque exécution du corps d'une boucle s'appelle une **itération.**
- Il existe deux types de boucle :
  - ✓ la boucle ***Tant que***
  - ✓ la boucle ***Pour***
- Chacune de ces boucles a ses avantages et ses inconvénients.
- Que vaut n à la fin des instructions ci-dessous?

```
>>> n=0
>>> n=n+1
>>> n=n+1
>>> n=n+1
>>> n=n+1
>>> n=n+1
>>> n=n+1
>>> n=n+1
>>> n=n+1
>>> n=n+1
>>> n=n+1
>>> que vaut n ???
```

La boucle POUR (FOR)

La syntaxe d'une boucle pour est la suivante :

Pour <variable> **Allant** de <valeur 1> à <valeur 2> **faire**
    <instructions>

*Le bloc <instructions> est **indenté** .*

La boucle Pour fait varier la valeur du compteur <variable> entre < valeur 1>et <valeur 2> .
Après chaque itération, le compteur est incrémenté de 1 (valeur par défaut).

Exemple :

✓Afficher 10 fois bonjour:

```
for i in range (10):
    print('Bonjour')
```

i : compteur
10: nombre d'itération
*Print('bonjour'):*corps de la boucle

✓Compter jusqu'à 10:

```
# les boucles:

n=0 #initialisation de la variable

for i in range (10): # i varie de 0 à 9

    # on incrémente i
    n=n+1

    #message affiché à chaque itération
    print('pour i = ',i,' n vaut :',n)

#message affiché à la sortie de la boucle
print('à la fin de la boucle n vaut :',n)
```

La boucle TANT QUE (WHILE)

La syntaxe d'une boucle tant que est la suivante :

Tant que <condition> **faire**
    <instructions>

*Le bloc <instructions> est **indenté** .*

Les instructions sont exécutées tant que la condition est vraie.

Exemple :

✓Afficher 10 fois bonjour:

```
i=0
while i <10:
    print(i)
    print('Bonjour')
    i=i+1
```

i = 10 : initialisation du compteur
i < 10 : condition
$i=i+1$: incrémentation du compteur.
NB:
Si pas de compteur la boucle sera sans fin car i sera toujours < 10!!

✓Que fait ce programme?

```
nb=int(input('entrez un nombre entier'))
i=1
n=1
while i <=nb:
    n=n*i
    i=i+1
print('le résultat est ...',n)
```

Programmes à réaliser:

➢ **En utilisant les boucles *for* :**

❖ Écrire un algorithme puis réaliser le programme qui affiche les entiers de 1 à 20.

❖ Écrire un algorithme puis réaliser le programme qui affiche les entiers pairs de 1 à 50.

❖ Écrire un algorithme puis réaliser le programme qui calcule la somme des n premiers nombres entiers positifs. S = 1+2+3+4+……..+n

L'algorithme demandera à l'utilisateur d'entrer la valeur de n.

❖ Écrire un algorithme puis réaliser le programme qui calcule la somme S suivante :

$S = 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2$.

L'algorithme demandera à l'utilisateur d'entrer la valeur de n.

➢ **Refaire le même travail en utilisant les boucles *while* .**

➢ **Le mot de passe :**

❖Pour accéder à un compte utilisateur, le mot de passe est : 'Bonjour'.

Réaliser un programme qui permet de vérifier le mot de passe saisi au clavier par l'utilisateur.

Si le mot de passe saisi est correct le programme s'arrête et affiche le message 'Bienvenue',

sinon le programme recommence et demande à l'utilisateur de faire un nouvel essai.

❖ Modifier le programme pour limiter le nombre d'essais à 3.

Mini projet: le juste prix:

❖L'objectif de cet algorithme ***justeprix*** est de retrouver en un nombre d'essais minimal un nombre entier tiré au hasard entre 1 et 1000. Il se décompose suivant les étapes suivantes :

✓Tirage au hasard d'un entier entre 1 et 1000 (module random):

✓ Proposition d'un entier par l'utilisateur et, selon la valeur proposée, un des messages suivants lui est communiqué :

- *Trop grand,*
- *Trop petit*
- *Bravo, vous avez trouvé en k essais !*

L'étape précédente est répétée tant que l'utilisateur n'a pas trouvé , et que le nombre d'essais est inférieur à 20. Si le nombre d'essais atteint 20 alors le message suivant apparaîtra :

- *Perdu, le nombre à découvrir était de …*

❖Compléter l'algorithme précédent en proposant à l'utilisateur de faire une nouvelle partie s'il le souhaite !