Boucles « While »

1- INTERET DE LA STRUCTURE WHILE :

<u>Exemple</u>: $n=10\,000$ bactéries ont été cultivées dans une fiole de laboratoire médical. Après injection d'un antibiotique, ce nombre n diminue de 5 % toutes les heures.

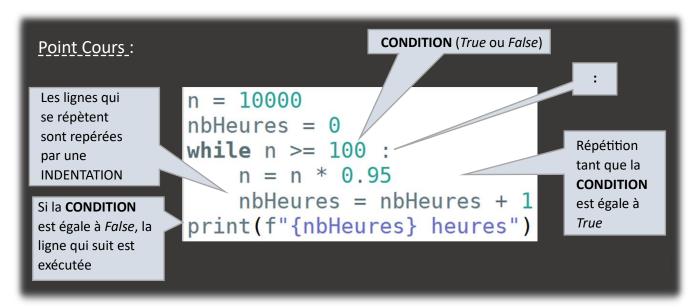
On souhaite réaliser un code python qui donne le nombre d'heures nécessaires pour avoir $n < 100 \ {\rm bactéries}.$



<i>Méthode 1</i> : boucle for			Méthode 2 : boucle while					

<u>Point Cours</u>: La structure WHILE est utilisée lorsque l'on veut réaliser une boucle pour laquelle le nombre d'itérations n'est pas connu au départ.

2- STRUCTURE WHILE EN LANGAGE PYTHON:



3- ATTENTION RISQUE DE BOUCLAGE INFINI :

Les boucles du type **for** i **in** range(10): ou **for** elt **in** liste:, sont dites Le nombre de répétitions est connu au départ.

Par opposition, les boucles du type | while condition == True :, sont dites
Le nombre de répétitions n'est connu au départ.

Point Cours:



Dans une structure WHILE, si la condition ne prend jamais la valeur *True*, le bouclage tourne à l'infini.

Lors de l'écriture de la condition, il faudra toujours s'assurer qu'elle prendra la valeur False à un moment donné.

4- EXERCICES :

Exercice 1 : Qu'obtient-on en exécutant ces codes ?

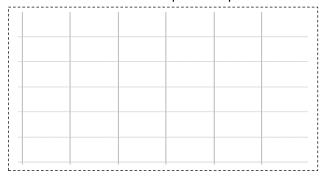
Code	Shell après exécution
print(5!= 2+2)	
<pre>n = 8 condition = (n%2 == 0 and n%4 == 0) print(condition)</pre>	
<pre>mot = "Bienvenue" condition = (mot == "Welcome") print(condition)</pre>	
<pre>while 5 > 4 : print("Vive les vacances")</pre>	
<pre>n = 0 while 3 != 1+1 : n = n + 1 print(n)</pre>	
<pre>condition = (8 > 7) while condition : print(condition)</pre>	

```
while False :
    print("Tu as gagné 100 000 €")

n = 100
while n > 1 :
    n = n // 2
    print(n)
```

Exercice 2:

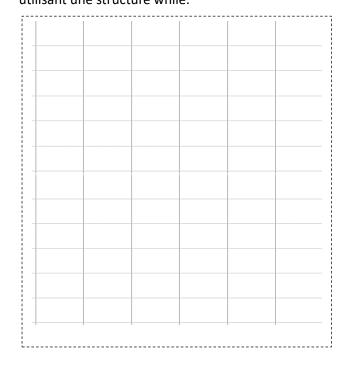
Ecrire un code utilisant une structure while qui permet d'obtenir le même résultat que le script ci-contre :



for i in range(100) :
 print(i)

Exercice 3:

Réécrire le code de la fonction *additionne()* ci-contre, en utilisant une structure while.



```
def additionne(n) :
    somme = 0
    for i in range(n) :
        somme = somme + i
    return somme

s = additionne(4)
print(s)
```

Exercice 4: Exécuter ce code « à la main » et écrire les valeurs prises par les variables.

somme	annee	somme = 1000
		annee = 2024
		while somme < 2000 :
		annee = annee + 1
		somme = somme * 1.2
		print(f"Tu auras {somme:.2f} € en {annee}")

Affichage final dans la console :

Exercice 5:

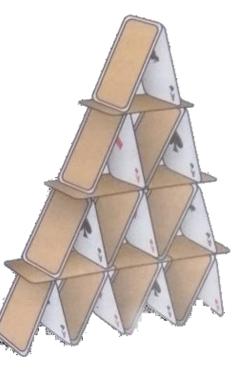
Une ruche contient 5000 abeilles au 1er juillet 2024. Durant tout l'été, le nombre d'abeilles diminue de 1 % chaque jour. Ecrire un code qui permette de calculer le nombre de jours à partir duquel les abeilles seront moins de 3000.

Exercice 5: Pierre s'ennuie. Pour passer du temps, il décide de

construire un château de carte.

Il lui faut 15 cartes pour construire un château de 3 étages, 26 cartes, pour un château de 4 étages et 0 carte pour un château de 0 étage.

1- Combien de cartes lui faut-il pour construire un château de 1 étage, ? de 2 étages ? de 6 étages ?



Voici l'algorithme d'une fonction qui permet de calculer le nombre de cartes nécessaires à la construction d'un château de n étages :

Nom de la fonction : nombreDeCartes

Arguments: n, entier positif

Début

Somme prend la valeur 0

Pour etage allant de 1 à n

Somme prend la valeur somme + etage*3-1

Renvoyer somme

Fin de la fonction

2- Ecrire le

code Python de cette fonction

- 3- Ecrire 2 lignes de test avec l'instruction assert qui permettent de valider le code écrit.
- Ecrire le code Python d'une fonction qui prend en paramètre un nombre N de cartes dont dispose Pierre et qui renvoie le nombre nd'étages du plus haut château qu'il peut construire.
- 5- Ecrire 2 lignes de test avec l'instruction assert qui permettent de valider le code écrit.