Informatique **Année 2021-2022**

MPSI-PCSI

TP4 : BOUCLES FOR et WHILE

L'objectif de cette séance est de consolider l'utilisation des boucles que nous avons rencontrées dans le TP2. (boucles "for")

Rappelons qu'une boucle est une séquence d'instructions qui se répète et on distingue alors deux types de boucles :

— les boucles inconditionnelles (ou boucles bornées) : les boucles "for" sont adaptées lorsqu'on connaît à l'avance le nombre de fois qu'une séquence devra se répéter.
Exemples :

```
1 for k in range(1,11) : #k prend les valeurs 1,2,...,10
2     print('lecarrede', k,'vaut',k**2)
```

permet d'afficher les carrés des entiers compris entre 1 et 10.
Une boucle for permet également d'accéder aux éléments d'une chaîne de caractères :

```
1 mot = 'Python'
2 for k in mot :
3   print(k)
```

— Les boucles inconditionnelles (ou boucles non bornées) : les boucles " while " qui présentent l'avantage d'effectuer une séquence d'instructions sans en connaître à l'avance le nombre de répétitions.
Une boucle "while" sera utilisée si les instructions doivent être répétées jusqu'à ce qu'une condition cesse, peu importe le nombre de fois.
Chaque répétition des instructions s'appelle un "tour de boucle" ou "itération". Lorsqu'on effectue la première itération, on dit qu'on entre dans la boucle, lorsque l'on a fini d'exécuter la dernière itération, on dit que l'on sort de la boucle.
La syntaxe fait intervenir le mot clé "while" et une condition à valeur booléenne (test booléen) :
while condition :
.....
bloc d'instructions
.....
Ainsi tant que ("while") la condition vaut True, la boucle continue ses itérations.
Exemple :

```
1 x = 1
2 while x!=11: # on aurait pu ecrire while x < 11 :
3   print(x)
4   x = x + 1
```

affiche les entiers de 1 à 10.

Remarque –
— Attention aux boucles "infinies" : il faut toujours s'assurer que la condition peut réellement devenir égale à False quand on utilise une boucle "while ".
— Concernant la condition de la boucle : il est souvent plus commode de chercher une condition de sortie de la boucle puis de déterminer sa négation.
— Il peut être intéressant d'utiliser un compteur initialisé à 0 et incrémenté à chaque passage dans la boucle "while" (il permet alors de connaître le nombre de passages dans la boucle) :
compteur = 0
while condition and compteur < 10 : # on choisit une certaine valeur du compteur pour être sûr de sortir de la boucle
....

bloc d'instructions
.....
compteur+=1

Exercice 1 :

Programme mystère
On considère l'algorithme suivant qui prend en entrée un entier naturel n.

Entrer n
k $\leftarrow$ 1
Tant que n $>$ 9 faire
    k $\leftarrow$ k + 1
    n $\leftarrow$ quotient de la division euclidienne de n par 10
Fin de boucle tant que
Renvoyer k

Écrire un programme traduisant cet algorithme en Python, puis tester le pour différents ordres de grandeur de n. Que calcule cet algorithme ?

Exercice 2 :

On considère la suite (u_n) définie par $u_0 = 2$ et

$$\forall n \in \mathbf{N}, u_{n+1} = \frac{1}{2}(u_n + \frac{2}{u_n})$$

On admet que la suite (u_n) est décroissante et converge vers $\sqrt{2}$

1. On pose $n = 10$. Calculer u_n en utilisant une boucle for. Vérifier que le résultat obtenu est proche de $\sqrt{2}$.

2. On pose $\epsilon = 10^{-2}$. En utilisant une boucle while, calculer le premier terme de la suite pour lequel $(u_n - \epsilon)^2 \leqslant 2$.

Exercice 3 :

La suite de Syracuse est définie de la manière suivante.

— On choisit le terme initial $u_0 \in \mathbf{N}^*$.
— Pour tout $n \geq 0$, $u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{sinon.} \end{cases}$

Par exemple, si $u_0 = 5$, on a $u_1 = 3 \times 5 + 1 = 16$, $u_2 = \frac{16}{2} = 8$, $u_3 = 4$, $u_4 = 2$, $u_5 = 1$, $u_6 = 4$, $u_7 = 2$, $u_8 = 1$, etc.

1. Testez (éventuellement à la main) différentes valeurs de u_0. Que pouvez-vous conjecturer ?
2. Écrire un programme qui détermine les n premiers termes de la suite où u_0 et n sont choisis par l'utilisateur.

Cette suite assez simple à construire pose encore problème aux mathématiciens. Quelle que soit la valeur choisie pour u_0, il semble exister un entier n tel que $u_n = 1$: la suite entre au bout d'un certain temps dans le cycle 1, 4, 2. Mais cette question reste ouverte à l'heure actuelle !... [1]

On appelle *temps de vol* le plus petit entier n tel que $u_n = 1$; on appelle *altitude maximale* la valeur maximale de la suite (il se pourrait que le temps de vol et l'altitude maximale soient infinis...).

3. Créez un programme qui détermine le temps de vol et l'altitude maximale après vous avoir demandé d'entrer la valeur de u_0.

1. Et l'on montrera peut-être un jour que l'on ne connaîtra jamais la réponse.

Exercice 4 :

Jeu du plus ou moins

Créer un programme qui demande à l'utilisateur de deviner un entier aléatoire compris entre 1 et 1000, et qui continue jusqu'à ce que la réponse convienne. À chaque proposition de l'utilisateur, le programme devra indiquer si le nombre saisi par l'utilisateur est plus grand ou plus petit que le nombre à deviner. Et lorsque l'utilisateur a découvert le nombre, le programme affichera le nombre d'essais nécessaire pour y parvenir.

Aide : La commande `randint(a,b)` du module `random` permet de tirer un nombre entier n compris entre `a` et `b` aléatoirement.

Combien d'essais vous faut-il au maximum (la réponse n'est pas 1000) ?