TP5 - Boucles while

1. Conversions entre types

Dans nos TP nous manipulons des valeurs de types divers : `int` (nombre entier), `str` (chaîne de caractères), `bool` (booléen), `float` (nombre à virgule), etc.

On a parfois besoin de **convertir** une valeur d'un type vers un autre. Pour cela on utilise le nom du type souhaité comme une fonction de conversion :
Exercice 1. Exécuter les instructions ci-dessous dans le shell et expliquer ce qui se passe :

```
a=3
b=[1,4,7,10]
phrase = "La raison de la suite " + b + " est " + a
```

Expliquer le message d'erreur : .....
Continuer avec les instructions suivantes :

```
c= str(a)
d= str(b)
phrase = "La raison de la suite " + d + " est " + c
```

Expliquer pourquoi il n'y a plus d'erreur : .....

2. La fonction `input`, ou quand Python pose les questions

En général un programme est écrit par quelqu'un, le *développeur*, ou *programmeur*, pour être **plus tard** utile à quelqu'un d'autre, qu'on appelle l'*utilisateur*. Ces distinctions entre le moment où le programme est écrit et le moment où il est utilisé, et entre la personne qui l'écrit et la personne qui l'utilise, ne sont pas évidentes dans nos TPs car tout cela ne fait qu'un, mais il faut les garder en tête pour bien comprendre ce qui suit.

Un programme peut avoir besoin de demander à l'utilisateur des informations que le développeur ne connaît pas au moment où il écrit le programme. Pour ce faire, on utilise la fonction `input` : voici un programme qui demande à l'utilisateur son nom afin de le saluer :

```
nom_utilisateur=input("Quel est votre nom ? ")
s= "Bonjour, "+nom_utilisateur
print(s)
```

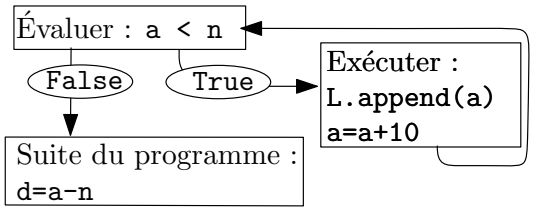
La fonction `input` prend en argument un message qui sera affiché à l'utilisateur avant d'attendre une réponse de sa part. Une fois que l'utilisateur a écrit sa réponse et appuyé sur «Entrée», cette réponse est retournée par la fonction `input` **sous la forme d'une chaîne de caractères**. Si on souhaite utiliser la réponse de l'utilisateur comme un nombre, calculer avec, etc. **il faut la convertir**.

Exercice 2. Écrire un programme qui demande à l'utilisateur son âge et affiche une phrase comme `"Vous êtes donc né(e) en 2008 ou en 2009"`

3. Boucle `while`

Il y a deux types de boucles, `for`, que l'on connait bien maintenant, et `while`, ou en français «tant que». Son exécution est décrite par le schéma suivant :
On évalue d'abord la condition, et si elle est vraie on exécute le corps de la boucle. Puis on évalue de nouveau la condition. Cela peut se répéter un nombre indéfini de fois, et dès que la condition est évaluée en `False`, la boucle se termine et on passe à la suite du programme.

```
a=0
n=37
L=[]
while a<n :
    L.append(a)
    a=a+10
d=a-n
```



Exercice 3.

(1) Prédire étape par étape l'exécution du programme ci-dessus, pour en déduire la valeur de `L` det de `d` après l'exécution. . . . . .

(2) Si on change la valeur de `n`, quel lien existe entre `n` et le résultat `d` ? . . . . . .

Remarque :
— La principale différence avec la boucle `for` est qu'on ne sait pas à l'avance combien d'itérations on fera.
— Il est même possible que la boucle ne s'arrête jamais, on parle de *boucle infinie* (par exemple ci-dessus, si l'on remplace `a=a+10` par `a=a-10`).
— Noter que la condition n'est pas évaluée *en permanence*, mais une fois avant le début de chaque itération.

Exercice 4. Écrire une fonction `puiss2(n)` qui renvoie le plus petit p tel que $2^p > n$.

Exercice 5. Écrire une fonction `mot()` qui demande à l'utilisateur un mot de 8 lettres : si la réponse de l'utilisateur n'a pas exactement 8 lettres, on lui repose la question jusqu'à ce qu'il donne un mot de 8 lettres. La fonction `mot()` renverra le mot choisi.

4. Jeu du nombre secret

On choisit un nombre N, par exemple N=100. Un joueur, le *gardien* choisit un nombre secret entre 0 et N-1. L'autre joueur, l'*attaquant* essaie de le deviner en proposant des nombres. Le gardien doit répondre en indiquant si le nombre à deviner est égal, inférieur ou supérieur au nombre proposé.

Exercice 6.

(1) Écrire une fonction `jeu(N)` qui prend en argument un nombre N et joue le rôle du gardien tandis que l'utilisateur fait l'attaquant :
— Le programme choisit un nombre `x` entre 0 et N-1 sans le communiquer. On pourra utiliser la fonction `randrange` du module `random`.
— Le programme demande à l'utilisateur de faire une proposition.
— Il compare cette proposition au nombre `x` à deviner et lui répond «trop petit», «trop grand», ou «gagné».
— Ceci se répète tant que l'utilisateur n'a pas deviné le nombre `x`.
— Une fois que le joueur a gagné, le programme affiche un message et s'arrête.

Stratégie de l'attaquant

(1) On joue avec N=10 :
   — Peut-on gagner à coup sûr en utilisant au maximum 10 propositions ? Si oui, expliquer comment : .....
   — Peut-on gagner à coup sûr en utilisant au maximum 4 propositions ? Si oui, expliquer comment : .....
   — Peut-on gagner à coup sûr en utilisant au maximum 3 propositions ? Si oui, expliquer comment : .....

(2) Quelle est la plus grande valeur de N pour laquelle l'attaquant peut gagner à coup sûr en 2 coups ?
   En 3 coups ? ..... En 4 coups ? .....

(3) Expliquer le raisonnement utilisé et trouver une formule générale qui donne u_k le plus grand nombre N pour lequel l'attaquant peut être certain de gagner en au plus k coups. .....

Exercice 7. ** Jeu du Mastermind

Dans ce jeu, le gardien choisit une suite de 4 chiffres **distincts**, que l'attaquant tente de deviner en proposant des suites de 4 chiffres distincts. Le gardien doit répondre en indiquant combien de chiffres sont exacts, et combien sont bons mais pas à la bonne place.

Par exemple si le code à deviner est 2794 et l'attaquant propose 7324, la réponse sera «1 exact, 2 déplacés».

(1) En cherchant une fonction adaptée dans le module `random`, tirer aléatoirement quatre chiffres distincts.

(2) Écrire une fonction `reponse(proposition,code)` qui prend en argument deux listes de 4 chiffres distincts et renvoie un couple d'entiers, le nombre de chiffres «exacts» et le nombre de chiffres «déplacés».

(3) Écrire une fonction qui joue le rôle du gardien en choisissant un code aléatoire, tandis que l'utilisateur fait l'attaquant et essaie de le deviner. Il faudra penser à utiliser les conversions nécessaires entre le type d'objets manipulés par le programme (une liste de 4 entiers ?), et ce que l'utilisateur saisit dans le shell (une chaîne de 4 caractères ?).