Institut National Universitaire Champollion

Chapitre 3

Boucles & Listes

Plan du chapitre

Boucle for

Boucle for

Exemple introductif : ou comment répéter une instruction.

écrire un programme qui dessine un carré de 4 étoiles.

Entrée [2]:
```
1 for i in range(4):
2     print("****")
3
```

```
****
****
****
****
```

Compteur : variable i
range(4) : signifie que i va varier de 0 à 3
Notez les mots clés **for** et **in.**
Notez aussi le : et l'indentation utilisée par le corps de boucle

Boucle for

Syntaxe :

for *compteur* **in range(n) :**
    instruction1
    instruction2
    …..
suite des instructions en dehors de la boucle

Si on veut que le compteur démarre à p et se termine à n-1 :

range(p,n)

Boucle for

Exemple :

Entrée [11]:

```
1 n = 10
2 for i in range(4,n):
3     if i == n-1:
4         print(i)
5     else:
6         print(i,end="")
7 print("en sortie de boucle i vaut : ",i)
```

```
456789
en sortie de boucle i vaut :  9
```

remarques :

en sortie de boucle le compteur vaut bien n-1

notez le end = ‘’ ‘’ dans l’instruction print pour éviter le retour à la ligne

Boucle for

Exemple :

écrire un code qui permet de sommer tous les entiers de 1 à un entier donné n.

il s'agit donc de faire le calcul 1 + 2 + 3 + … + n

on remarque de suite qu'on doit répéter une opération d'addition n-1 fois, et que le nombres à additionner s'indentent de 1 à chaque nouvelle opération ...

Boucle for

D'où la solution :

Entrée [1]:
```
1 n = 10
2 somme = 0
3 for i in range(n+1):
4     somme = somme + i
5 print("la somme des entiers de 1 à ",n," est : ",somme)
```

```
la somme des entiers de 1 à  10  est :  55
```

on peut modifier « le pas » du compteur, par exemple indenter i de 2 en 2 : range(1,n,**2**)

:
```
1 n = 10
2 somme = 0
3 for i in range(1,n+1,2):
4     somme = somme + i
5 print("la somme des entiers impairs de 1 à ",n," est : ",somme)
```

```
la somme des entiers impairs de 1 à  10  est :  25
```

Vous remarquerez qu'on a besoin d'initialiser la boucle avec l'instruction somme = 0.

Boucle while

Boucle while

Avec la boucle for, on doit connaître le nombre de répétitions à effectuer avant l'exécution de celle-ci.
Suivant les problèmes à résoudre ce n'est pas toujours le cas.

Par exemple : donnez le premier nombre qui est divisible par 131, à partir de 23421 …

idée : on va tester la divisibilité par 131 des nombres successifs à partir de 23421.

attention : il faut être sûr que l'on a bien un nombre fini de tests à faire, sinon on va faire une boucle infinie.
Ici pas de soucis on va effectuer au pire 130 tests ...

On pourrait faire un for de longueur 130, mais ...

Boucle while

Une solution (technique du drapeau) :

Entrée [18]:

```
1 inconnu = 23421
2 trouve = False
3 while not trouve:
4     inconnu = inconnu+1
5     if (inconnu%131 == 0):
6         trouve = True
7 print("le nombre inconnu est : ",inconnu)
```

```
le nombre inconnu est :  23449
```

Boucle while

en plus concis :

Entrée [20]:

```
1 inconnu = 23421
2 while inconnu%131 != 0:
3     inconnu = inconnu+1
4 print("le nombre inconnu est : ",inconnu)
```

```
le nombre inconnu est :  23449
```

Dans une boucle while on a besoin :

- d'une **initialisation** contenant une **variable de contrôle**, ici inconnu = 23421
- d'une **condition de continuation** : celle-ci est la négation logique de la condition d'arrêt.

ici la condition de continuation est :

(inconnu%131 != 0) == True

Boucle while

Syntaxe :

initialisation
While *condition de continuation* **:**
    instruction1
    instruction2
    ….
terminaison éventuelle (affichage résultat ….)

Dans l'exemple précédent la condition de continuation faisait intervenir un compteur : ce n'est pas toujours le cas.

Boucle while

Un exemple typique élémentaire est le contrôle d'une saisie par un utilisateur :

Entrée [2]:
```
1 note = float(input("saisissez une note comprise entre 0 et 20   "))
2 while(note < 0 or note > 20):
3     note = float(input("saisissez une note comprise entre 0 et 20   "))
4 print("la note saisie est : ",note)
```

```
saisissez une note comprise entre 0 et 20   -6.5
saisissez une note comprise entre 0 et 20   21
saisissez une note comprise entre 0 et 20   14.5
la note saisie est :  14.5
```

La fonction input(String) permet de lire une chaîne de caractère saisie au clavier.
La fonction float(String) permet de traduire la chaîne de caractère en réel.

Boucle while

Boucle **for** *vs* boucle **while**

quelle(s) différences entre ces deux codes ?

Entrée [4]:

```
1 for i in range(1,5):
2     print("****")
3 print(" i après le for : ",i)
```

```
****
****
****
****
 i après le for :  4
```

Entrée [5]:

```
1 i = 1
2 while (i<5):
3     print("****")
4     i = i+1
5 print("i après le while : ",i)
```

```
****
****
****
****
i après le while :  5
```

Boucle while

Boucle **for** *vs* boucle **while**

- l'écriture du for est plus concise : pas d'initialisation, pas d'expression explicite non plus de la condition de continuation : elles sont contenues dans le range(1,5).
- Pas d'écriture explicite non plus de l'incrémentation du compteur avec le for.
- À la sortie des boucles for et while, les compteurs n'ont pas la même valeur :

Attention si on souhaite utiliser ces valeurs en sortie de boucle !!

Boucle while

Boucle **for** *vs* boucle **while**

Finalement :

- On utilise une boucle for quand on sait par avance combien d'itérations on souhaite faire.
- On utilise une boucle while quand on ne sait pas combien d'itérations on va faire, ce nombre d'itérations étant dépendant d'une condition de continuation.

NE SURTOUT PAS MODIFIER LE COMPTEUR DANS LE CORPS DE LA BOUCLE FOR :
cela est souvent du à une mauvaise conception, et est souvent source d'erreurs (boucles infinies par exemple)

Les listes

Les listes

Une liste est une structure de données contenant une collection d'éléments.

Exemple d'initialisation d'une liste :

Entrée [10]:
```
1 L = [1,4,-4,6,9,-2]
2 print("L = ",L)
```

```
L =  [1, 4, -4, 6, 9, -2]
```

une liste peut contenir des éléments de type différents :

Entrée [11]:
```
1 L = ["une chaine", True, 5, -4, 6.87]
2 print("L = ",L)
```

```
L =  ['une chaine', True, 5, -4, 6.87]
```

à éviter si possible ...

Les listes

On peut connaître le nombre d'éléments qu'il y a dans une liste : fonction **len(**uneListe**)**

Entrée [14]:
```
1 L = [1,5,7,4,9,45,75,-6]
2 print("longueur de L : ",len(L))
```

```
longueur de L :  8
```

on peut accéder à un éléments d'une liste L: ils sont indicés de 0, à len(L) -1 :

Entrée [15]:
```
1 L = [5,7,3,-1]
2 for i in range(len(L)):
3     print("L[",i,"] = ",L[i])
```

```
L[ 0 ] =  5
L[ 1 ] =  7
L[ 2 ] =  3
L[ 3 ] =  -1
```

notez au passage comment utiliser la fonction len, pour faire une boucle for sur les éléments d'une liste

Les listes

Les éléments d'une liste sont bien des variables :

Entrée [17]:
```
1 L = [1,5,7,4,9,45,75,-6]
2 L[3] = L[1]+L[4]
3 print("L[3] = ",L[3])
```

```
L[3] =  14
```

on peut savoir si un élément est présent ou pas dans une liste, fonction **in** et **not in** :

Entrée [1]:
```
1 L = [1,5,7,4,9,45,76,-6]
2 print(" 9 est présent dans la liste : ",9 in L)
3 print("10 n est pas present dans la liste : ",10 not in L)
```

```
 9 est présent dans la liste :  True
10 n est pas present dans la liste :  True
```

Les listes

On peut supprimer un élément d'une liste, fonction **del**

Entrée [20]:

```
1 L = [1,5,7,4,9,45,75,-6]
2 print("L = ",L)
3 del(L[3])
4 print("L = ",L)
```

```
L =  [1, 5, 7, 4, 9, 45, 75, -6]
L =  [1, 5, 7, 9, 45, 75, -6]
```

On peut rajouter en fin de liste un élément, fonction **append** :

Entrée [21]:

```
1 L = [1,5,7,4,9,45,75,-6]
2 print("L = ",L)
3 L.append(10)
4 print("L = ",L)
```

```
L =  [1, 5, 7, 4, 9, 45, 75, -6]
L =  [1, 5, 7, 4, 9, 45, 75, -6, 10]
```

Il existe bien d'autres fonctions sur les listes : vous chercherez ...

Exercices boucles

Exercice 1 : on se donne un entier n, imprimer sur une ligne n étoiles. Exemple si n = 5, vous imprimez *****

Exercice 2 : on se donne un entier n. Calculez la somme : $1^2 + 2^2 + \ldots + n^2$. Vous vérifierez dans vos tests que cette somme vaut n(n+1)(2n+1)/6.

Exercice 3 : on se donne un entier n pair. Chercher le plus grand diviseur impair de n.
(on peut trouver plusieurs façons de le faire : à discuter ...)

Exercices Listes

Exercice 4 : on se donne une liste L. Affichez le premier et le dernier élément de la liste. Faire saisir un entier k, et afficher le k-ième élément de la liste s'il existe. S'il n'existe pas afficher un message l'indiquant.

Exercice 5 : on se donne une liste d'entiers L. Calculez la moyenne des éléments de L.

Exercice 6 : on se donne une liste d'entiers L. Calculez le plus grand et le plus petit élément de la liste.

Exercice 7 : on se donne une liste d'entiers L et un entier k. écrire un programme indiquant si l'entier k appartient ou non à la liste. (sans utiliser la fonction in ….). S'il appartient à la liste précisez sa position dans la liste.
(correction voir cours suivant)