INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Leçon sur les structures itératives

Université de Toulouse

Du sélectif à l'itérative

- La boucle TantQue est répétitive : on répète ce que l'on doit faire tant qu'une condition de sortie n'est pas vérifiée.
- Autre variété de boucle : la boucle itérative
  - Classique appelée boucle for (ou boucle pour)
  - On connait a priori le nombre de fois que l'on va parcourir la boucle.
- Exemple classique d'utilisation : calcul de la somme d'une suite géométrique:

$$U_n = \sum_{k=0}^{n} u_0 Q^k = u_0 + u_0 Q + u_0 Q^2 + \cdots + u_0 Q^n$$

Suite géométrique version TQ

- **Calcul de la valeur de la suite à l'instant n+1….**
  - dépend des toutes les valeurs précédentes….

$$U_n = \sum_{k=0}^{n} u_0 . Q^k$$

$$U_n = u_0 . Q^n + \sum_{k=0}^{n-1} u_0 . Q^k$$

$$U_n = u_0 . Q^n + U_{n-1}$$

```
Indice ←0
Un ← U0
Tantque (Indice <= n)
        Un ← Un + U0 * Q(indice)
        Indice = Indice + 1
FinTantque
```

Ca sent la boucle ou je ne m'y connais pas !!!

Suite géométrique version TQ

- **Soit donc en Python :**

```
Q = 0.25
u0 = 1
un = u0
Nmax = int(input('Jusque quel indice doit on aller :'))
indice = 1

while (indice <= Nmax) :
    un = un + u0*Q**(indice)
    indice = indice + 1

print('Au bout de ',Nmax,' la somme de la suite de raison ',Q,'vaut {0:.3f}'.format(un))
```

Version boucle Pour

- **Intérêt : s'épargner un peu de fatigue par une gestion *intégrée* de la progression de la variable (ici Indice) qui lui sert de compteur (*incrémentation*)**

```
Un ← u0
Pour Indice allant de 1 à n
        Un ← Un + u0 * Q^(Indice)
FinPour
```

Pas de gestion de la valeur Indice… son incrémentation est prise en charge par la boucle Pour

Implémentation Python

- En python, la boucle *pour* travaille sur un ensemble d'éléments (que l'on nommera couramment une liste)
- En terme d'algorithme cela pourrait conduire à écrire :

```
Pour tous les entiers de l'intervalle [1 n] faire :
        Un ← Un + u0 * Q(Indice)
```

- Il y a différentes façons de faire cela. La plus simple et la plus courante est d'utiliser la pseudo-fonction range()

La fonction range()

- **C'est une fonction très spéciale…elle ne retourne pas de valeur sauf si on la sollicite dans la boucle pour.**
- **range(start,stop,step) peut prendre 3 arguments d'entrée de type int**
  - **start : valeur initial**
  - **stop : valeur finale (exclue dans l'intervalle)**
  - **step : valeur d'incrémentation**
- **Exemples**
  - **range(1,10) fournira les entiers 1,2,…9**
  - **range(3,22,2) fournira les entiers 3,5,7…21**
  - **range(100,10,-20) fournira les entiers 100,80,60…20**

Utilisation de range()

- Appliqué à la boucle *pour* (for) cela donne par exemple :

```
for indice in range(1,10) :
        print('valeur d\'indice',indice)
```

- Soit à l'exécution :

```
valeur d'indice 1
valeur d'indice 2
valeur d'indice 3
valeur d'indice 4
valeur d'indice 5
valeur d'indice 6
valeur d'indice 7
valeur d'indice 8
valeur d'indice 9
```

Indice est un nom de variable => choisi par le programmeur

On ne va que jusque 9 (10 est exclu)

Application à notre exemple

- **Pour calculer la somme de notre suite géométrique :**

```
Q = 0.25
u0 = 1
un = u0
Nmax = int(input('Jusqu'à quel indice doit on aller :'))

for indice in range(1,Nmax+1) :
    un = un + u0*Q**(indice)

print('Au bout de ',Nmax,' la somme de la suite de raison ',Q,'vaut {0:.3f}'.format(un))
```

A bien noter : indice est « géré » par la boucle for et est directement utilisable à l'intérieur de la boucle

On doit spécifier Nmax+1 pour inclure Nmax

Application directe 1

- **Ecrire une boucle for qui affiche la table de multiplication du nombre B.**
  - B sera saisi par l'utilisateur, a priori 2 ≤ B ≤ 9
  - On affichera la table de Bx1 jusque Bx10

```
B = int(input('Quelle table voulez vous afficher :'))

while (B < 2 or B > 9) :
    print('B doit être compris entre 2 et 9 \n ')
    B = int(input('Redonnez la table que vous voulez afficher :'))

for Parcoure in range(1,11) :
    print('\n\t',B,'x',Parcoure,'=',B*Parcoure)
```

Application directe 2

- Afficher tous les nombres positifs <1000 et divisibles par 191

```
print('Les nombres divisibles par 191 sont :')

for valeur in range(191,1000) :
    if ((valeur % 191) == 0) :
        print('\n',valeur)

##############################################
print('\n\n Version rusée ')
##############################################
for valeur in range(191,1000,191) :
    print('\n',valeur)
```

Autre cas d'utilisation

- range() est utilisable dans la grande majorité des cas et correspond bien à la boucle for
- Autre possibilité : faire une boucle for avec tous les éléments d'un ensemble non forcément entier et continu
- Utilisation de l'appartenance in avec
  - Une chaine de caractères
  - Une liste
- Ce sont deux cas d'utilisation similaire mais il existe une différence structurelle entre eux….notion d'objets non exploitée ici !

Utilisation avec une chaine de caractères

- On sait saisir et afficher une chaine :

```
Chene = "Bonjour !"
print(Chene)
```

- Chene est un ensemble de caractères, on peut les « extraire » un à un dans une boucle for comme :

```
for care in Chene :
        print("\n -->", care)
```

- Ce qui donnera :

```
--> B
--> o
--> n
--> j
--> o
--> u
--> r
-->
--> !
```

Application directe 3

- **Comptabiliser le nombre de ‘e’ qu’il y a dans une chaine préalablement saisie par l’utilisateur.**

```
texte = input("Donne ta phrase : ")
cpt_e = 0
for carac in texte :
    if (carac =="e") :
        cpt_e = cpt_e+1
print("Dans la phrase",texte)
print("\n Il y a ",cpt_e, "e")
```

Utilisation avec une liste

- On verra la notion de liste plus tard…mais on peut l'appréhender comme un ensemble de valeur :

```
ensemble = [0.1, 0.25, -1 ,8 , 45.58]
```

- De cet ensemble, à l'instar de la chaine de caractères, on peut extraire chaque élément dans une boucle for et les traiter :

```
for val in ensemble :
    print("le carré de",round(val,2), " est ",round(val**2,4))
```

```
le carré de 0.10 est 0.0100
le carré de 0.25 est 0.0625
le carré de -1.00 est 1.0000
le carré de 8.00 est 64.0000
le carré de 45.58 est 2077.5364
```

Application directe 4

- Afficher le cosinus de tous les angles d'un cercle modulo $\pi/4$

```
import math

pi = math.pi
angl_div = [ 0, 1/4, 1/2, 3/4, 1, 5/4, 3/2, 7/4, 2]
for angle in angl_div :
    cocosse = math.cos(angle*math.pi)
```

```
# version rusée qui n'utilise pas les listes…
#############################################################
for angle in range(0,9) :
    cocosse = math.cos(angle*0.25*math.pi)
    print('le cosinus de {0:.2f} x pi est
{1:.3f}'.format(0.25*angle,cocosse))
```