CHAPITRE 12

LISTES

I. Notion de liste

Définition 1

- En langage Python, une **liste**[1] est objet délimité par des crochets et qui correspond à un ensemble fini d'objets ordonné et séparé par des virgules.
- La **longueur d'une liste** est le nombre d'objets qu'elle contient.
- La liste étant un ensemble ordonné, chaque objet dans celle-ci a une position fixe.
  L'**indice** de l'objet permet de repérer la position de l'objet avec une valeur comprise entre 0 et $\ell - 1$, ℓ étant la longueur de la liste.

Dispositif 1

En langage Python, pour une liste baptisée `liste`, on accède à son élément placé à la k-ème place grâce à `liste[k-1]`.

Exemple 1

```
liste = [5, 3, 2 , 'spam'] #implémentation d'une liste#
len(liste) #renvoie la longueur de la liste#
liste[0] #renvoie le premier élément de la liste#
liste[3] #renvoie le quatième élément de la liste#
```

Arès exécution, la console affiche :

```
4
5
'spam'
```

Dispositif 2

Le mot clef `in` permet de tester si un objet appartient ou non à une liste ; ce test renvoie un booléen.

Exemple 2

```
1 in [7, 4 , 8]
9 in [9, 9, 5]
9 not in [9, 9, 5]
```

Après exécution, la console renvoie :

```
False
True
False
```

Dispositif 3

En langage Python, on peut modifier une liste baptisée `liste` à l'aide de méthodes :

- `liste.append(foo)` rajoute en bout de liste l'élément `foo` ;
- `liste.remove(foo)` enlève l'élément `foo`.

II. Énumération

II.1. Itérable

Définition 2

Un **itérable** est un ensemble que l'ordinateur peut itérer, *i.e.* en parcourir les éléments un à un.

- Le plus classique est fourni par la commande `range`.
  - `range(b)` fournit « l'intervalle semi-ouvert » $[\![0\,;b[\![$ (il contient b nombres entiers) ;
  - `range(a, b)` fournit « l'intervalle semi-ouvert » $[\![a\,;b[\![$;
- Une **liste** est un objet itérable.

Exemple 3

- `range(6)` correspond à l'ensemble $[\![0\,;5]\!]$.
- `range(2, 5)` correspond à l'ensemble $[\![2\,;4]\!]$.

II.2. Mot clef for

Définition 3

Pour programmer une **boucle bornée**, on procède ainsi

```
for variable in itérable:
    effet
```

La variable `variable` prendra alors successivement toutes les valeurs de l'ensemble `itérable`.
L'indentation est nécessaire pour le programme interprète bien l'espace `effet`.

Exemple 4

En rentrant ce code :

```
for k in range(1, 4):
    print(k)
```

la console renvoie après exécution :

```
1
2
3
```

```
cpt = 1
for k in range(3):
    cpt = 3*cpt
    print(cpt)
```

la console renvoie après exécution :

```
3
9
27
```

Exemple 5

Sur un livret, un investisseur met 600 €. Chaque année, les intérêts du livret augmente le montant présent en fin d'année sur le livret de 1,15 %. Des frais de 4,2 € sont prélevés annuellement.
On peut déterminer le montant sur le livret au bout de `n` années par :

```
def livret(n):
    s = 600 #somme à l'ouverture#
    for t in range(n): #sert à décompter les années#
        s = 1.0115*s - 4.2
    return round(s, 2) #arrondit au centime s#

print(livret(5), livret(10), livret(20))
```

et la console renvoie : `613.81, 628.44, 660.30`
Ainsi, le détenteur aura 613,81 € au bout de 5 ans, ...

II.3. Liste par compréhension

Définition 4
Quand les éléments d'une liste sont générables par une formule, on peut définir la liste **par compréhension**.

Exemple 6
```
[k**3 for k in range(10)] #liste de nbr sous la forme k**3 pour k = 0 à 9#
```
renvoie la liste `[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]`.

II.4. Somme d'une liste

Dispositif 4
On peut réaliser la somme d'une liste à l'aide de l'objet intégré `sum`.