TP02 : Boucles – Listes-1

Acquis du T.P. précédent :

- Accès à sa session sur le serveur pédagogique
- Organisation et utilisation des dossiers sur le serveur pédagogique
- Console interactive IDLE
- Écrire et exécuter un script de programme avec IDLE
- Types et valeurs pour les types non mutables (`int`, `float`, `str`, `bool`, `None`)
- Opérateurs sur les types `int`, `float` et `bool`
- Variables, définition, nommage et affection
- Fonctions (syntaxe de base pour la définition d'une fonction, appel à une fonction, récupération des valeurs en sortie, fonctions à effet (de bord), fonctions sans paramètres, fonctions sans valeur de retour.

Préliminaires

- Rappel : instructions pour simuler le **tirage au hasard d'un entier**
  - importer la fonction `randint` du module `random` ;
  - un appel `randint(a, b)` avec a et b entiers, $a \leq b$, renvoie un entier tiré pseudo-aléatoirement dans l'intervalle d'entiers $[\![a, b]\!]$ (en simulant la loi uniforme sur $[\![a, b]\!]$).
    - Exemple d'exécution dans la console interactive :

```
>>> from random import randint
>>> randint(1, 6) # 1 et 6 inclus
4
```

```
>>> randint(1, 6) # 1 et 6 inclus
1
>>> randint(1, 6) # 1 et 6 inclus
4
```

- Complément (utilisé dans ce TP03) : **tirage au hasard d'un flottant dans l'intervalle [0 ; 1[**
  - importer la fonction `random` du module `random` (**attention :** la fonction utilisée à le même nom que le module auquel elle appartient (module dans lequel elle est définie) ;
  - un appel `random()` (cette fonction s'appelle sans argument) renvoie un flottant tiré pseudo-aléatoirement parmi les flottants de l'intervalle [0 ; 1[(en simulant la loi uniforme sur [0 ; 1[).
    - Exemple d'exécution dans la console interactive :

```
>>> from random import random
>>> random()
0.44446191317330386
```

```
>>> random()
0.6198443385100294
>>> random()
0.23041696156611657
```

Remarque : on peut obtenir (après importation) une documentation rapide de la fonction, à l'aide de la fonction `help`

```
>>> help(random)
Help on built-in function random:

random() method of random.Random instance
    random() -> x in the interval [0, 1).
```

Une documentation extensive peut aussi être obtenue en saisissant dans un moteur de recherche la requête « *Python3 random fr* » (rechercher ensuite (Ctrl + F) la chaîne « `random.random` » dans la page atteinte https://docs.python.org/fr/3.9/library/random.html)

Note : selon le programme officiel, il ne peut être demandé d'utiliser les fonctions `random()` et `randint(a,b)` sans qu'une documentation ne soit donnée au candidat.

1 Boucles

1.1 Boucles while :

Exercice 1. Écrire un script **ex1.py**, dans lequel on simule le tirage simultané de deux dés équilibrés à 6 faces, numérotées de 1 à 6, jusqu'à obtenir un double 6.

On affichera les résultats de chacun des tirages effectués.
On rappelle les fonction `randint` et `randrange` du module `random`.
Pour a et b entiers, $(a \leq b)$:

- `randint(a, b)` renvoie un entier aléatoire compris entre a et b inclus ;
- `randrange(a, b)` renvoie un entier aléatoire compris entre a inclus et b exclu.

Exercice 2. Reprendre le script précédent en un script **ex2_3.py**, et ajouter un compteur, nommé `cpt`, afin de compter le nombre de tirages nécessaires pour l'obtention d'un double 6.

Exercice 3. Écrire une fonction `simulation(seuil)`, répétant le tirage aléatoire d'un flottant compris dans l'intervalle $[0\,;1[$, jusqu'à ce que la somme des flottants obtenus soit supérieure strictement à la valeur `seuil`.
La fonction renverra le nombre de tirages effectués et la moyenne des valeurs tirées.

1.2 Boucles for :

1.2.1 Syntaxe de base en Python : « for i in range(deb, fin + 1) »

Exercice 4. Dans un script **ex4_5_6.py**, écrire une boucle `for` affichant tous les entiers de 7 à 18 inclus.

Exercice 5. Dans le même script, définir une fonction `affiche(n, p)` affichant tous les entiers compris dans l'intervalle $[\![n; p]\!]$.

Exercice 6. Dans le même script, définir une fonction `table(n)` affichant la table de multiplication d'un entier strictement positif n, en utilisant une boucle `for` permettant de faire varier le multiplicateur de n. L'affichage produit devant être de la forme (en prenant $n = 7$ pour exemple) :

```
>>> table(7)
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
10 * 7 = 70
```

1.2.2 Syntaxe pour une répétition à l'identique n fois : « for i in range(n) »

Exercice 7. Dans un script **ex7.py**, définir une fonction `repete(n)`, affichant n fois une même chaîne de caractères qui sera définie dans le corps de la fonction.

Définir ensuite une fonction, `repete2(ch, n)`, affichant n fois la chaîne de caractères `ch`, passée en argument.

1.2.3 Syntaxe avec incrément : « for i in range(deb, fin + 1, pas) »

Exercice 8. Dans un script **ex8.py**, définir une fonction `mult7()` affichant tous les multiples de 7, compris entre 0 et 100 (on utilisera une boucle `for` avec l'incrément 7).

Exercice 9. Dans un script **ex9.py**, écrire une fonction `sommes`, de paramètre entier n et qui calcule et renvoie la somme des entiers de 1 à n, et la somme des carrés des entiers de 1 à n. Inclure dans votre fichier plusieurs tests de la fonction, et la vérification que les formules de calcul de ses sommes données en mathématiques sont bien vérifiées.

2 Listes de valeurs de type non mutable

On suppose connues :

- la fonction, `len`, donnant la longueur d'une liste (`len(L)` renvoie le nombre d'éléments d'une liste `L`) ;
- la numérotation positive des éléments d'une liste (les positions des éléments d'une liste `L`, sont numérotées de zéro à `len(L) - 1`.

2.1 Définition d'une liste en extension

Exercice 10. Dans un script **ex10_11.py**, définir en extension une liste de dix valeurs entières.

Dans la console interactive, afficher les premier, deuxième, avant-dernier et dernier termes de cette liste.

Ajouter dans le script les instructions nécessaires pour afficher ces quatre termes.

Exercice 11. Dans le même script, définir une fonction, `enumereposval(L)`, affichant les termes de la liste L passée en argument, avec leur position dans la liste L.

- Exemple d'appel à la fonction

```
>>> enumereposval(['a', 'b', 'c'])
Valeur a en position 0
Valeur b en position 1
Valeur c en position 2
>>>
```

2.2 Définition d'une liste par ajouts successifs

Exercice 12. Dans un script **ex12_13.py**, définir, en utilisant la méthode `.append()`, une fonction `liste_constante`, prenant en argument un entier n et une valeur v et renvoyant une liste contenant n fois la valeur v.

Exercice 13. Définir dans le même script, une fonction `liste_alea_int` prenant en argument un entier n et deux entiers a et b $(a < b)$ et renvoyant une liste, construite par ajouts successifs, contenant n entiers tirés pseudo-aléatoirement dans l'intervalle $[a\,;b[$.

2.3 Définition d'une liste en compréhension

Exercice 14. Dans un script **ex14_15.py**, définir une fonction `liste_constante2`, prenant en argument un entier n et une valeur v et renvoyant une liste contenant n fois la valeur v, construite à l'aide d'une compréhension de listes.

Proposer une implémentation alternative, sous la forme d'une fonction `liste_constante3`, en utilisant cette fois l'opération de multiplication d'une liste par un entier.

Exercice 15. Dans le même script, définir une fonction `liste_alea_int2` prenant en argument un entier n et deux entiers a et b et renvoyant une liste, construite en compréhension, contenant n entiers tirés pseudo-aléatoirement dans l'intervalle $[a\,;b[$.