BIlan du travail à la maison

Exercice 1
Avez-vous des question concernant les exercices traités sur codingame ?

Exercice 2
Pour le prochain TD, traiter chez vous les feuilles n°7 et 8 qui reprennent des éléments de ce TD.

I Les itérables en python

Les objets qui nous intéressent aujourd'hui sont plus complexes que les nombres manipulés dans le TD précédent. Ce sont des objets qui en "contiennent" d'autres, dans un ordre bien défini (on peut identifier le premier objet contenu, le deuxième....). En voici plusieurs exemples

1.1 Les listes

Il s'agit du type itérable dont nous nous servirons les plus cette année. On reconnaît une liste aux crochets `[` et `]` qui la délimite. Une liste peut contenir comme élément tout objet python. Pour créer une liste à plusieurs éléments, on sépare ces éléments par des virgules. Voici quelques exemples

```
1  L1 = []
2  L2 = [2, 4, 6, 8]
```

`L1` est une liste vide, qui ne contient aucun élément. `L2` est une liste contenant 4 éléments. On dit qu'elle est de *longueur* 4.

Exercice 3
Créer les listes précédentes dans la console puis trouver leur longueur en utilisant la fonction `len` qui peut prendre une liste comme argument et retourne la longueur de son argument.

Exercice 4
Les éléments d'une liste sont numérotés par convention à partir de 0. Le numéro d'un élément dans une liste est son **indice**. Pour accéder à l'élément d'indice `i` de la liste `L`, on utilise la syntaxe `L[i]`.
Deviner le résultat des commandes suivantes puis tester dans la console.

```
1  L2[2]
2  L1[0]
```

Exercice 5
Lorsque L est une liste déjà créée, on peut lui ajouter un élément par la commande `L.append(element)` où *element* représente l'élément à ajouter.
Ajouter les deux booléens `True` et `False` à la liste `L1` puis vérifier qu'elle est bien de longueur 2. Que vaut `L1[0]` maintenant ?

Exercice 6
Tester les commandes

```
1  L1 + L2
2  L2 + L2
```

Quel est l'effet du + entre deux listes ?

Exercice 7
Rappelons qu'une liste peut contenir comme élément n'importe quel objet python. Quelle est la longueur de la liste `L` ?

```
1  L = [[0, 1], [2, 3]]
```

Que vaut `L[0]` ? `L[1][0]` ?

1.2 Les intervalles

Ces itérables sont plus limités que les listes. Ils ne contiennent que des entiers, spécifiés une fois pour toute lors de la création de l'intervalle. Pour construire un nouvel intervalle, on utilise la fonction `range`. Cette fonction peut s'utiliser de 3 manières distinctes

Spécifier les deux bornes. On utilise `range(debut, fin)` où `debut` et `fin` sont des entiers (donnés par leurs chiffres, ou contenus dans des variables) pour créer l'intervalle d'entiers $[debut, fin[$ (remarquer que l'entier de fin est exclu de l'intervalle)

Exercice 8
Combien d'éléments contient `range(0, 10)` ? On pourra vérifier dans la console grâce à la fonction `len`
Tester ensuite

```
list(range(0, 10))
```

La fonction `list` transforme un autre itérable en une liste.

Seulement la borne de fin la commande `range(entier)` est équivalente à `range(0, entier)` : la borne inférieur (inclue) de l'intervalle est 0 par défaut.

Utiliser un pas La forme la plus aboutie de la fonction `range` prend 3 arguments : `range(debut, fin, pas)`. `debut` et `fin` ont la même signification que dans la première utilisation, mais cette fois les éléments contenus sont $debut, debut + pas, debut + 2 \times pas \ldots$.

Exercice 9
Donner la forme à 3 arguments équivalente à `range(42)`.

Exercice 10
Quels sont les entiers contenus dans `range(0, 8, 2)` ? et ceux dans `range(5, 0, -1)` ? On pourra vérifier les réponses dans la console en utilisant la fonction `list` (mais seulement après avoir calculé la réponse à la main sur une feuille...).

Exercice 11
On considère une liste `L` de longueur n un entier. Donner une commande `range` qui retourne un intervalle contenant exactement tous les indices des éléments de `L`.
Donner la même commande, mais qui ne fait pas apparaître n. On pensera à `len`

II Les boules for

Les boucles permettent, dans un algorithme ou dans un programme, de répéter des opérations bien précises. Nous allons étudier un type particulier de boucle : la boucle `for` (qui se traduit par "pour").

2.1 Sytaxe

```
for variable in iterable:
        instruction1
        [instruction2...]
```

Les crochets dénotent ici une partie optionnelle : on peut taper une ou plusieurs instructions dans une boucle for. Une instruction par ligne, comme nous commençons à en prendre l'habitude.
Plusieurs remarques sur la forme :

- `variable` est un nom de variable, qui n'a pas besoin d'être définie avant.
- `iterable` est un itérable, par exemple une liste ou un range.
- remarquez la présence des deux points ":" à la fin de l'instruction `for` (oui, c'est une instruction qui peur donc prendre une place à l'intérieur d'un for...)
- les instructions à répéter sont **indentées** : la ligne de code commence par strictement plus d'espaces que la ligne d'instruction `for`. Toutes les instructions à répéter doivent avoir la même indentation (le même nombre d'espaces au début) : on utilise la touche **TAB** pour introduire 4 espaces qui représentent l'indentation standard.

2.2 Signification

L'exécution de code précédent est assez simple en première approche :

- `variable` prend comme valeur le premier élément de `iterable`.
- on exécute séquentiellement (l'une après l'autre) les instructions contenues dans la boucle.
- retour au début de la boucle, mais `variable` prend comme valeur le second élément de `iterable`.
- sortie de la boucle lorsque `variable` a pris comme valeur le dernier élément de `iterable` et qu'on a exécuté toutes les instructions contenues dans la boucle.

2.3 Au boulot !

Exercice 12
Quelle est la valeur contenue dans la variable `a` à la fun de la boucle ?

```
a = 0
for i in range(1, 5):
    a = a + i
```

On commencera par bien identifier les éléments contenus dans l'itérable.

Exercice 13
Avec la liste `L2` définie précédemment,

```
for i in range(len(L2)):
    L2[i] = L2[i] // 2
```

Que contient maintenant la liste `L2` ?

Exercice 14

```
L = [1, 3, 5, 7]
a = 1
for elt in L:
    a = a * elt
```

Quelle est la valeur contenue dans `a` ?

III Examens des éléments d'une liste

Exercice 15
Expliquer l'éventuel changement de valeur de la variable `M`. Comment nommer cette fonction ? On cherche un nom qui décrit la valeur de retour.

```
def f(a, b):
    M = a
    if b > a:
        M = b
    return M
```

Exercice 16
Pour une liste `L` qui contient des nombres, on cherche à déterminer le plus grand de ses éléments. Notre méthode ne doit pas dépendre de la liste `L` (elle doit marcher quelque soit la liste donnée).

1. Décrire en plusieurs phrases un algorithme qui répond à la question posé.
2. Compléter, dans l'éditeur, la fonction suivante. Les lignes ou morceaux de lignes qui commencent par "#" sont des **commentaires**. Python ne les lit pas, il sont seulement là pour aider à la compréhension du code.

```
def elt_maxi(L):
    M = L[0] # le plus grand élément trouvé pour l'instant
    # compléter ici, en plusieurs lignes
    return M
```

Exercice 17
Créer dans l'éditeur une fonction `compte_element(L, element)` qui prend comme argument une liste `L` et un objet `element` et retourne le nombre de fois où `element` est présent dans la liste `L`.
On pourra utiliser la construction classique de compteur

```
c =0
# plus loin dans le code
c = c + 1
```

Exercice 18
Adapter la méthode de l'exercice 16 pour créer une fonction qui retourne le deuxième plus grand élément de la liste `L`. Si tous les éléments de `L` sont égaux, on retournera le maximum, sinon un élément strictement inférieur au maximum mais supérieur ou égal aux autres éléments.
Exprimer en fonction de n, la longueur de `L`, le nombre de comparaison ($\leqslant, ge, <, >, =$) entre éléments de la liste `L` nécessaires au calcul de ce second maximum.

Pour aller plus loin

Exercice 19

1. Créer une fonction `puissance(a, n)` qui calcule et retourne a^n où `a` est un nombre et `n` est un entier naturel.
2. Créer une fonction `elements_pairs(L)` qui prend en argument une liste d'entiers `L` et retourne la liste composée des éléments de `L` qui sont pairs.
3. Créer la fonction `chiffres(n)` qui renvoie la liste des chiffres de l'écriture décimale de `n`, le chiffre des unités en première position. Adapter ensuite pour obtenir l'écriture binaire.
4. Créer la fonction `palindrome(n)` qui renvoie un booléen (`True` ou `False`) suivant que l'écriture décimale de `n` est un palindrome ou non (on peut lire `n` de gauche à droite ou de droite à gauche, comme 12321 qui est un palindrome).