Contrôle continu - Programmation (MASCO 1)
Durée : 2h30
Documents autorisés : le polycopié du cours

A lire avant de commencer

— Certaines fonctions que l'on vous demande d'écrire existent déjà en Python. Si c'est le cas, vous **ne devez pas** les utiliser dans votre solution. L'idée est de comprendre comment ces fonctions ont été programmées !

— Ecrivez de manière la plus lisible possible, les copies illisibles ne seront pas lues.

1 Listes

Q1.1 Ecrire une fonction `memeLongueur(L1, L2)` qui prend en arguments deux listes `L1` et `L2` et retourne `True` si `L1` et `L2` ont même longueur et `False` sinon.

Q1.2 Ecrire une fonction `inverse(L1, L2)` qui prend en arguments deux listes `L1` et `L2` et retourne `True` si `L1` et `L2` sont l'inverse l'une de l'autre (`L1` à l'envers est égale à `L2`).
Ne pas utiliser la fonction `reverse` de Python !

```
>>> A = [1, 2, 3]
>>> B = [3, 2, 1]
>>> inverse(A, B)
True
```

Q1.3 Ecrire une fonction `sousListeImpaire(L)` qui prend en argument une liste `L` et retourne une nouvelle liste composée des éléments de `L` d'indice impair.

```
>>> L = [2, 4, 6 , 9, 22, 56]
>>> sousListeImpaire(L)
[4, 9, 56]
```

2 Listes de listes

Q2.1 Ecrire une fonction `decoupe(L, n)` qui prend en argument une liste `L` et un entier `n` et découpe `L` en sous-listes de longeur `n`. Plus précisément, la fonction retourne une nouvelle liste composée elle-même de listes, telle que la première correspond au `n` premiers éléments de `L`, la seconde aux `n` éléments suivants ... Si la longueur de `L` n'est pas divisible par `n`, alors la dernière sous-liste aura une longueur inférieure à `n`

```
>>> A = [45, 43, 67, 2, 59, 34, 11, 45, 35, 10]
>>> decoupe(A, 3)
[[45, 43, 67], [2, 59, 34], [11, 45, 35], 10]
```

Q2.2 Ecrire une fonction `fusionne(L)` qui prend en argument une liste de listes `L` et retourne une liste composée des éléments des sous-listes de `L` dans l'ordre dans lequel ils apparaîssent dans `L`.

```
>>> A = [[45, 43, 67], [2, 59, 34], [11, 45, 35], 10]
>>> fusionne(A)
[45, 43, 67, 2, 59, 34, 11, 45, 35, 10]
```

Q2.3 Ecrire une fonction `complete(L)` qui prend en argument une liste de listes `L` et complète les sous-listes de `L`. L'opération de completion consiste à faire en sorte que toutes les sous-listes de `L` aient la même longueur (qui est la longueur de la sous-liste la plus longue) en ajoutant des zéros à la fin des listes dont la longueur est inférieure à celle de la sous-liste la plus longue.

```
>>> A = [[1, 2], [3, 2, 4], [1]]
>>> complete(A)
[[1, 2, 0], [3, 2, 4], [1, 0, 0]]
```

3 Dictionnaires

On considère dans les exercices suivants des dictionnaires, qui sont des listes de la forme `[clef1, valeur1, clef2, valeur2 ...]` où une clef est une chaîne de caractère et une valeur est une liste, comme dans l'exemple suivant :
`['Mathilde', ['12 rue de Lodi', '0601020304'], 'Julien', ['15 rue Sénac', '0602040201']]`
Ce dictionnaire est constitué de deux clefs (`Mathilde` et `Julien`) et de deux valeurs (`['12 rue de Lodi', '0601020304']` et `['15 rue Sénac', '0602040201']`). Les valeurs peuvent contenir n'importe quoi, ça ne nous importe pas, du moment que ce sont des listes.

Q3.1 Ecrire une fonction `clefs(D)` qui prend en argument un dictionnaire `D` et retourne la liste des clefs de `D`.

```
>>> L = ['Mathilde', ['12 rue de Lodi', '0601020304'], 'Julien', ['15 rue Sénac', '0602040201'
>>> clefs(L)
['Mathilde', 'Julien']
```

Q3.2 Ecrire une fonction `recherche(D, clef)` qui prend en argument un dictionnaire `D` et une clef `clef` et retourne la valeur correspondant à `clef`. Si `clef` n'existe pas dans `D`, la fonction retourne `None`.

```
>>> recherche(L, 'Mathilde')
['12 rue de Lodi', '0601020304']
```

Q3.3 Ecrire une fonction `elimine(D, clef)` qui prend en argument un dictionnaire `D` et une clef `clef` et élimine `clef` et la valeur lui correspondant. Si `clef` n'existe pas dans `D`, la fonction ne modifie pas `D`.

```
>>> elimine(L, 'Mathilde')
>>> print(L)
 ['Julien', ['15 rue Sénac', '0602040201']]
```

Q3.1 Ecrire une fonction `ajoute(D, clef, valeur)` qui prend en argument un dictionnaire `D`, une clef `clef` et une valeur `valeur`. Si `clef` existe déjà dans `D`, sa valeur est remplacée par `valeur`, sinon `clef` et `valeur` sont ajoutés à la fin de `D`.

```
>>> ajoute(L, 'Julien', ['xxx'])
>>> print(L)
['Mathilde', ['12 rue de Lodi', '0601020304'], 'Julien', ['xxx']]
```