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 la fonction `inverse(L)` qui prend en argument une liste `L` et retourne un nouvelle liste correspondant à la liste `L` à l'envers.
Note : Ne pas utiliser la fonction `reverse` *de Python*

```
>>> L = inverse([1, 2, 3])
[3, 2, 1]
```

Q1.2 Ecrire la fonction `inverse2(L)` qui prend en argument une liste `L` et modifie la liste `L` en inversant ses éléments.
Note : Ne pas utiliser la fonction `reverse` *de Python*

```
>>> L = [1, 2, 3]
>>> inverse2(L)
>>> print(L)
[3, 2, 1]
```

Q1.3 Ecrire la fonction `separe(L)` qui prend en argument une liste `L` et retourne deux listes, une contenant les éléments de `L` d'indices pairs et l'autre contenant les éléments de `L` d'indices impairs.

```
>>> L = [7, 5, 9, 12, 1, 8]
>>> LP, LI = separe(L)
>>> print(LP)
[7, 9, 1]
>>> print(LI)
[5, 12, 8]
```

Q1.4 Ecrire la fonction `fusionne(LP, LI)` qui prend en argument deux listes `LP` et `LI` retourne une liste dont les éléments d'indices pairs sont les éléments de `LP` et les éléments d'indices impairs sont les éléments de `LI`. En d'autres termes cette fonction fait l'inverse de la fonction `separe`. Si une des deux liste est plus courte que l'autre, les éléments manquants seront remplacés par la valeur `-1`.

```
>>> L = [7, 5, 9, 12, 1, 8]
>>> fusionne(separe(L))
[7, 5, 9, 12, 1, 8]
>>> fusionne([4,5], [2,7,9])
[4, 2, 5, 7, -1, 9]
```

2 Listes et chaînes de caractères

Q2.1 Ecrire la fonction `creeChaine(L)` où `L` est une liste de caractères. Cette fonction retourne la chaîne de caractères formée par les éléments de la liste `L`

```
>>> creeChaine(['b', 'o', 'n', 'j', 'o', 'u', 'r'])
bonjour
```

Q2.2 Ecrire la fonction `creeListe(c)` où `c` est une chaîne de caractères. Cette fonction retourne la liste formée des caractères qui composent la chaîne `c` (en d'autres termes, c'est la fonction inverse de `creeChaine`)

```
>>> creeListe("bonjour")
['b', 'o', 'n', 'j', 'o', 'u', 'r']
```

Q2.3 Ecrire la fonction `decoupe(c)` où `c` est une chaîne de caractères. Cette fonction retourne la liste formée mots qui constituent la chaîne `c`. On suppose que les mots sont séparés les uns des autres par le caractère espace.
Note : Ne pas utiliser la fonction `split` *de Python, qui permet de faire la même chose*

```
>>> decoupe("bonjour les enfants")
["bonjour", "les", "enfants"]
```

3 Ensembles

On considère dans les exercices suivants des ensembles d'entiers positifs représentés sous la forme de listes composées de `0` et de `1`. Le principe est le suivant : si l'entier i appartient à l'ensemble, alors la liste correspondante possède un `1` à la case d'indice i. L'ensemble $E = \{2, 4, 7\}$ par exemple, sera représenté par la liste suivante : `[0,0,1,0,1,0,0,1]`.

Q3.1 Ecrire la fonction `ensemble(L)` où `L` est une liste d'entiers. Cette fonction retourne l'ensemble correspondant à cette liste.

```
>>> ensemble([2,7,4])
[0,0,1,0,1,0,0,1]
```

Q3.2 Ecrire la fonction `appartient(Ens, e)` où `Ens` est un ensemble (au sens défini ci-dessus) et `e` est un entier. La fonction retourne `True` si `e` appartient à `Ens`, et `False` sinon.

```
>>> appartient([0,1,0,1,1], 3)
True
>>> appartient([0,1,0,1,1], 2)
False
>>> appartient([0,1,0,1,1], 12)
False
```

Q3.3 Ecrire la fonction `ajoute(Ens, e)` où `Ens` est un ensemble (au sens défini ci-dessus) et `e` est un entier. La fonction ajoute l'entier `e` à l'ensemble `Ens`.

```
E = [0,1,0,1,1]
>>> ajoute(E, 2)
>>> print(E)
[0,1,1,1,1]
>>> ajoute(E, 8)
>>> print(E)
[0,1,1,1,1,0,0,0,1]
```