TP n° 03 – Manipulation des listes et recherche

I Manipulations de listes

Définition. Une liste est une structure de données qui contient une série de valeurs. Pour en déclarer une, on utilise la syntaxe suivante :

```
[1,2,3]
['a','x','mot']
[1,'chiffre',-2**3]
```

Exercice 1. Commandes de base.

1. Créer la liste `L=['a','b','c','d','e','f']`.
2. Afficher le résultat des commandes suivantes : `L[0]`, `L[1]`. Comment faire pour afficher l'élément `'d'` de `L` ?
3. Que renvoie `L[6]` ? Pourquoi ?
4. Que renvoie `L[2:5]` ? Que renvoie en général `L[i:j]` ? Et `L[i:]` ?
5. Que renvoie `len(L)` ? Comment faire pour afficher le dernier élément de `L` ?
6. Opérations sur les listes : `+` et `*`.
   Exécuter les commandes suivantes pour comprendre l'effet de ces opérations sur les listes (il faudra afficher le résultat des opérations pour constater leurs actions) :

   ```
   [1,2,3]+[7,8,9]
   [1,2]+[6,7,8,9]
   [0]*10
   ```

7. Méthodes sur les listes : `pop` et `append`.
   Exécuter les commandes suivantes pour comprendre l'effet de ces méthodes sur les listes :

   ```
   L=[1,2,3]
   L.pop()
   L.append(7)
   ```

Définition. Liste en compréhension
Une liste en compréhension est une liste dont le contenu est défini par filtrage du contenu d'une autre liste. Sa construction se rapproche de la notation ensembliste en mathématiques. Exemples :

en langage ensembliste :	$\mathcal{S}_1 = \{3n+2 \mid n \in \mathbb{N},\ n < 20\}$
en Python :	`S1=[3*n+2 for n in range(20)]`

Exercice 2.
Utilisez une liste en compréhension pour créer la liste des multiples de 5 entre 0 et 100.

Exercice 3. Soit `L1` la liste suivante : `L1=[12,-3,8,1.2,7]`.

1. Ajouter `10` à la fin de la liste. Afficher la nouvelle liste.
2. Afficher l'élément `1.2` de la liste.
3. Retirer l'élément `12` de la liste.
4. Modifier l'élément en position 1 de la liste par l'élément `77`.
   A ce stade, votre liste `L1` doit être : `[-3,77,1.2,7,10]`.
5. Créer la liste `L2=[-3,77,1.2,7,10,0,1,2,3,...,100]`.
6. Créer la liste des 30 premiers termes de `L2` sans ses 3 premiers termes.
7. Créer la liste `L2` suivie de 40 zéros.

Exercice 4. Utilisez une boucle `for` pour afficher un par un tous les termes de la liste `L=[2,8,-7,3]`. (il y a deux syntaxes possibles).

Exercice 5. Ecrire une fonction `renverser` qui à une liste renvoie la liste renversée.
On reprogramme ainsi la méthode `reverse` déjà implémentée dans Python.

```
>>> L=[2,8,-1,7]
>>> renverser(L)
[7,-1,8,2]
```

II La bibliothèque copy

Définition. Une bibliothèque est un ensemble de fonctions. Celles-ci sont regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire. Celles-ci permettent de faire : du calcul numérique, du graphisme,... Quelques exemples : les bibliothèques `math`, `matplotlib.pyplot`, `copy`.
Il existe plusieurs syntaxes pour importer une bibliothèque et pour ensuite dans le corps du programme, appeler les fonctions :

```
import math as m
m.cos(m.pi/3)
```

```
import math
math.cos(math.pi/3)
```

```
from math import cos, pi
cos(pi/3)
```

Exercice 6. Importez la bibliothèque `copy` sous l'alias `copy`.
Testez les fonctions suivantes. Expliquez les résultats :

```
L=[1,2,3,4]
M=L
L[0]=100
print(L,M)
print(id(L),id(M))
```

Programme n°1

```
L=[1,2,3,4]
M=copy.copy(L)
L[0]=100
print(L,M)
print(id(L),id(M))
```

Programme n°2

III Recherches

Exercice 7. Recherche dans une liste
Programmer une fonction `position(liste,element)` qui a comme entrée une liste et un élément et qui renvoie la position de l'élément dans la liste et qui renvoie `None` si l'élément n'est pas dans la liste.
On reprogramme ainsi la méthode `index` déjà implémentée dans Python

Exercice 8. Recherche du maximum dans une liste de nombres.

1. Ecrire une fonction `maximum(liste)` qui renvoie le maximum d'une liste de nombres non triée.

   ```
   >>>maximum([2,8,-7,3])
   8
   ```

2. Dans l'algorithme précédent, combien de fois parcourt-on la liste ?
3. Ecrire une fonction `maximum2(liste)` qui renvoie la deuxième valeur maximale d'une liste de nombres non triés, tous distincts.

   ```
   >>>maximum2([2,8,-7,3])
   3
   ```

Exercice 9. Pour les listes, il existe une fonction `max` et une méthode `index`. On en rappelle les syntaxes respectives dans l'annexe.
Testez `max` et `index` sur des exemples pour comprendre ce qu'elles renvoient.
A l'aide de `max` et `index`, déterminer la position du maximum de la liste `L`.

IV Exercices en plus

Exercice 10. Recherche d'un mot dans une chaîne de caractères.

1. Ecrire une fonction `estIci(motif,texte,i)` qui a comme entrée deux listes (ou deux chaînes de caractères) `motif` et `texte` et un entier `i` et qui renvoie `True` si `motif` est dans `texte` à la position `i` et `False` sinon.

   ```
   >>>estIci('le','Bonjour le monde',8)
   True
   >>>estIci('le','Bonjour le monde',9)
   False
   ```

2. Ecrire une fonction `recherche(motif,texte)` qui a comme entrée deux listes (ou deux chaînes de caractères) et qui renvoie `True` si `motif` est dans `texte` et `False` sinon.

```
>>>recherche('le','Bonjour le monde')
True
>>>recherche('bonjour','Bonjour le monde')
False
```

Exercice 11. Calendrier grégorien.
La manipulation des dates dans les logiciels de gestion, ou encore sur les sites web de réservation, doit s'effectuer conformément au calendrier grégorien (entré en vigueur à la fin du XVI^{e} siècle en France) en précisant pour chaque date le jour de la semaine. Or le calendrier grégorien est un format assez éloigné des formats habituels de stockage des nombres dans un ordinateur.
On cherche à déterminer, pour une date donnée, sa position dans l'énumération des jours depuis le 1er janvier 1600. et le jour de la semaine correspondant.

1. Proposer une fonction `nombre_de_jours(jours,mois,annee)` qui prend en entrée une date de la forme `jours,mois,annee` postérieure au 1,1,1600 et renvoie le nombre de jours écoulés entre le 1er janvier 1600 et la date considérée, sans tenir compte des années bissextiles (c'est-à-dire en supposant que chaque année comporte 365 jours). On pourra utiliser une liste des nombres de jours de chaque mois pour une année non bissextile : `m = [31,28,31,30,31,30,31,31,30,31,30,31]`.
2. Les années bissextiles sont déterminées par la règle suivante [1] :
   — Si l'année est divisible par 4 et non-divisible par 100, c'est une année bissextile (elle a 366 jours).
   — Si l'année est divisible par 400, c'est une année bissextile (elle a 366 jours).
   — Sinon, l'année n'est pas bissextile (elle a 365 jours).
   Proposer une fonction `bissextile(annee)` renvoyant `True` si l'année est bissextile et `False` sinon.
3. Modifier le programme de la fonction `nombre_de_jours(jours,mois,annee)` pour tenir compte des années bissextiles. Par exemple, `nombre_de_jours(1,2,1600)` doit retourner la valeur 31, `nombre_de_jours(1,1,1604)` la valeur 1461 (366+365+365+365) puisque l'année 1600 est bissextile.
4. Le 1er janvier 2001 était un lundi. Déterminer, en utilisant la fonction de la question précédente, quel jour de la semaine tombera le 1er mai 2040. Quel jour de la semaine est tombé le 14 juillet 1789 ?

1. Wikipédia en français, « Année bissextile », consulté le 23 novembre 2015.

Annexe

Fonctions et méthodes sur les listes

En Python, il y a deux façons de faire des opérations sur les objets qu'on manipule : les fonctions et les méthodes.
La différence est, pour vous, surtout d'ordre syntaxique.

Syntaxe fonction :	Syntaxe méthode :
`fonction(objet ,parametres)`	`objet.methode(parametres)`

Pour les listes, on trouve des fonctions et des méthodes. Certaines modifient la liste et ne renvoient rien, d'autres renvoient un résultat sans modifier la liste.

Exemples de méthodes pour les listes.
- **append(élément)** : modifie la liste en ajoutant élément à la fin de la liste
- **pop()** : modifie la liste en supprimant le dernier élément
- **reverse()** : modifie la liste en inversant les valeurs de la liste
- **index(élément)** : renvoie la position de élément dans la liste

Exemples de fonctions pour les listes :
- **len** : renvoie la longueur de la liste
- **max** : renvoie le maximum d'une liste de nombres

Fonctions et méthodes au programme

Opération	Exemple
Création d'une liste vide	`l=[]`
Création	`l=[1,56,13]`
Accès direct	`l[0]`
Extraction de tranche	`l[1:10]` `l[3:]`
Longueur	`len(l)`
Concaténation	`l1+l2`
Répétition	`[0]*k`
Modification par affectation	`l[0]=0`
Ajout en fin de liste	`l.append(1)`
Suppression en fin de liste	`l.pop()`
Création par compréhension	`[k**2 for k in range(n)]`
Copie	`copy(l)`

Fonctions et méthodes sur les chaînes de caractères

On retrouve certaines fonctions et méthodes pour les chaînes de caractères.

Opération	Exemple
Création d'une chaîne vide	`l=''`
Création	`s = 'Ma chaîne'`
Accès direct	`s[0]`
Extraction de tranche	`s[1:10]`
Longueur	`len(s)`
Concaténation	`s1+s2`
Répétition	`'e'*k`