Ce TP a pour but de se familiariser avec l'utilisation des listes et des chaînes de caractères.

I Définition de listes

I.1 Définition en extension

Une liste, de type **list**, est une suite finie d'éléments, de types quelconques, rassemblés dans un même objet Python. Autrement dit, les listes sont des tableaux avec une seule ligne qui permettent de stocker des données composites dans une même structure. Pour définir une liste, on place les éléments entre des crochets en les séparant par des virgules, c'est la définition d'une liste en extension : $[x_1, x_2, \cdots, x_n]$
Dans une liste, l'ordre des éléments compte.

Les opérations élémentaires possibles sur les listes sont rassemblées dans le tableau suivant :

Opérations sur les listes	Instruction Python
égalité	==
concaténation	+
répétition n fois	$n*$
taille ou longueur	len
appartenance	in
somme	sum

Exercice 1: Tapez les instructions suivantes dans le *shell* :

☞ $l = [\text{"}AB\text{"}, 4 > 3, 8]$ ☞ type(l) ☞ print(l) ☞ $len(l)$

Une liste peut contenir des objets de différents type.

Exercice 2: Tapez les instructions suivantes dans le *shell* :

☞ $t = 3 * [1, 2]$ ☞ 3 in t ☞ $t = t + [3, 4]$ ☞ print(t) ☞ 3 in t ☞ sum(t)

La liste vide [ ] a son intérêt. Elle est souvent le point de départ d'une liste que l'on veut construire.

Il existe également des **méthodes** sur les listes : ce sont des fonctions directement intégrées à la liste.

•Ajout d'un élément à une liste : la méthode *append* permet d'ajouter un élément en fin de liste.

Exercice 3: Tapez les instructions suivantes dans le *shell* :

☞ $u = [1, 2, 4]$ ☞ $u.append(8)$ ☞ $print(u)$ ☞ $u + [16]$ ☞ print(u)

•Suppression d'un élément d'une liste.
La méthode *remove* : $u.remove(x)$ supprime la première occurence de x de la liste u.
La méthode *pop* : $u.pop(i)$ retourne l'élément de la position i et le supprime de la liste.
Si i est omis, le dernier élément de la liste est retourné et supprimé.

Exercice 4: Tapez les instructions suivantes dans le *shell* :

☞ $v = [5, 6, 4, 5, 2, 1, 6]$ ☞ $v.remove(6)$ ☞ $print(v)$
☞ $v.pop(2)$ ☞ print(v) ☞ $v.pop()$ ☞ print(v)

I.2 Création de liste avec la commande range

Avec la commande $range(a, b, r)$, on parcourt une liste d'entiers compris entre a et $b-1$ avec un pas de r, ie les entiers $a, a+r, a+2r, \cdots$ en s'arrêtant au plus à $b-1$. On peut alors définir des listes correspondant à ces entiers.

Exercice 5: Tapez les instructions suivantes dans le *shell* :

☞ $t = list(range(1, 17, 2))$ ☞ $u = list(range(2, 10))$ ☞ $print(t, u)$

I.3 Définition en compréhension

La description d'une liste par la donnée exhaustive de ses éléments n'est pas la seule façon de définir un tel objet. On peut aussi procéder en compréhension, de façon similaire à ce que l'on fait en mathématiques. Pour cela, on utilise la syntaxe suivante :

[*expression* **for** *indice* in *liste*]

Il est même possible d'utiliser plusieurs indices en écrivant plusieurs instructions du type for *indice* in *liste*.

Exercice 6: Tapez les instructions suivantes dans le *shell* :

☞ $[k**3 \text{ for } k \text{ in } range(1,10)]$ ☞ $sum([k**3 \text{ for } k \text{ in } range(1,10)])$
☞ $((9*10)/2)**2$ # Que constate-t-on ?
☞ $[10*i+j \text{ for } i \text{ in } range(3) \text{ for } j \text{ in } range(10)]$
☞ $[10*i+j \text{ for } j \text{ in } range(10) \text{ for } i \text{ in } range(3)]$

II Accès aux éléments d'une chaîne ou d'une liste

II.1 Cas d'une chaîne de caractères

Pour définir une chaîne de caractères, on rappelle qu'il faut la délimiter par des côtes ou des guillemets :

$$ch =' Python', \quad cha = "J'aime\ l'info"$$

Si ch désigne une chaîne de caractères donnée de longueur n, Python numérote les caractères de la chaîne (i.e. les lettres de la phrase) de 0 à $n-1$. Dès lors, pour accéder au caractère numéroté k, on utilise tout simplement la commande : $ch[k]$.

L'utilisation d'un indice négatif permet d'accéder aux caractères à partir de la fin de la chaîne. Ainsi $ch[-1]$ désigne le dernier élément d'une chaîne, $ch[-2]$ l'avant dernier,etc.

On peut effectuer des coupes : $ch[i:j]$ désigne la chaîne constituée des caractères de ch dont les numéros sont entre i et $j-1$.

Exercice 7: Tapez les instructions suivantes dans le *shell* :

☞ $ch = "Python"$ ☞ $ch[0]$ ☞ $ch[1]$ ☞ $ch[2]$ ☞ $ch[-1]$ ☞ $ch[2:6]$
☞ $ch[5] = "u"$ ☞ $ch[6]$ ☞ surnom=$ch[0:5] + "u"$ ☞ print(surnom)

Il est possible de consulter les caractères d'une chaîne, il est en revanche impossible de les modifier au sein même de la chaîne. On dit que les chaînes de caractères sont **non-mutables**.
Pour modifier certains caractères d'une chaîne, il convient par conséquent de créer une nouvelle chaîne dans laquelle on recopie ce que l'on veut garder de la chaîne de départ avant de rajouter les caractères modifiés.

II.2 Cas des listes

L'affectation d'une liste à une variable permet d'accéder individuellement aux éléments de la liste. Les commandes sont très semblables à celles introduites pour les chaînes de caractères.

Si l désigne une liste donnée de longueur n, Python numérote les éléments de la liste de 0 à $n-1$. Dès lors, pour accéder à l'élément numéroté k , on utilise tout simplement la commande : $l[k]$.

Comme pour les chaînes, l'utilisation d'un indice négatif permet d'accéder aux éléments à partir de la fin de la liste. Ainsi $l[-1]$ désigne le dernier élément d'une liste, $l[-2]$ l'avant dernier,etc.

On peut effectuer des coupes : $l[i:j]$ désigne la liste des éléments de l dont les numéros sont entre i et $j-1$.

Exercice 8: Tapez les instructions suivantes dans le *shell* :

☞ $sport = ["tennis","badminton","football","natation","judo"]$ ☞ $len(sport)$
☞ $sport[0]$ ☞ $sport[1]$ ☞ $sport[2]$ ☞ $sport[3]$ ☞ $sport[4]$ ☞ $sport[5]$
☞ $sport+=["rugby"]$ ☞ $sport$ ☞ $sport[5]$ ☞ $sport[0:2]$ # sports de raquette
☞ $sport[4] = "kayak"$ ☞ $sport$ ☞ del($sport[2], sport[4]$) ☞ $sport$

Au contraire des chaînes de caractères, il est possible de modifier (ou d'effacer) un élément d'une liste au sein même de cette liste. On dit que les listes sont **mutables**.

La mutabilité des listes oblige à la prudence lorsqu'on souhaite dupliquer une liste. Par exemple, en écrivant $x = [5,2,9]$ suivi de $y = x$, on dispose en fait d'une seule et même liste qui possèdent deux noms. Si l'on décide alors de muter le dernier élément de la liste x en écrivant $x[2] = 10$, la liste devient $[5,2,10]$ mais son adresse ne change pas en mémoire. Les variables x et y pointent donc toujours vers la même liste. Si l'on demande les valeurs de x et y, on obtient $[5,2,10]$ dans les deux cas. Tout se passe donc comme si la mutation de x s'était répercutée sur y.
La bonne syntaxe pour réellement dupliquer la liste (ie créer une autre variable de même valeur que x) est $y = x[:]$ ou d'utiliser la méthode $y = x.copy()$

II.3 Parcourir les éléments des chaînes de caractères ou des listes

Un itérable est une donnée informatique susceptible d'être parcourue. On pense bien sûr aux intervalles d'entiers (range) mais aussi aux chaînes de caractères (parcourues caractère par caractère) ou aux listes (parcourues élément par élément).

On a deux possibilités :
•on parcourt élément par élément d'un itérable en utilisant une boucle **for** : la syntaxe, très naturelle, est alors de la forme :

```
for  k  in iterable :

    Bloc d'instructions à exécuter      # Indentation obligatoire

# On sort de la boucle en stoppant l'indentation
```

Exemple : Dans l'éditeur, créez la fonction suivante :

```
1 def nombredelettredansmot(lettre, mot) :
2     """ Cette fonction calcule et renvoie le nombre d'occurence de lettre dans mot """
3     c=0 # Initialisation
4     for le in mot :
5         if le==lettre : # on teste si le lettre le en cours est égale à la lettre cherchée
6             c+=1 # Le compteur augmente d'une unité
7     return c
```

Exécutez le fichier et tapez les instructions suivantes :

☞ $nombredelettredansmot('a','abracadabra')$ ☞ $mot='abracadabra'$
☞ $mot.count('a')$

•Au lieu de "boucler" sur les éléments de la liste, on va utiliser les indices de ces éléments.

Exemple : Dans l'éditeur, créez la fonction suivante :

```
1 def sommeliste(L) :
2     """ Cette fonction calcule et renvoie la somme des éléments d'une liste L """
3     s=0 # initialisation de la somme
4     n=len(L) # taille de la liste
5     for k  in  range(n) : # k va prendre les valeurs de 0 à n-1
6         print(L[k]) # Affichage de l'élément L[k] de L
7         s=s+L[k] # on ajoute à s chaque élément de L
8     return s
```

Exécutez le fichier et tapez les instructions suivantes :

☞ $L=list(range(1,10))$ ☞ $S=sommeliste(L)$ ☞ $print(S)$ ☞ $S==(10*9)/2$

III Exercices :

1. Définir la liste suivante : $li=[14,58,23,15,16,27,19,15]$. Afficher la liste.
   Ajouter 34 en fin de liste et afficher li.
   Retirer 58 de la liste, afficher li.
   Donner l'indice de l'occurrence 16 dans li. On pourra utiliser la méthode *index*.
   Afficher la sous-liste du 1-er au 4-ième élément.
   Tester l'appartenance de 26 et 27 à cette liste.
   Compter le nombre d'occurrence de 15, 27 et 58 dans li. On pourra utiliser la méthode *count*.

2. Ecrire une fonction **ajoute_indice(L)** qui étant donné une liste L renvoie une liste M telle que chaque élément de M correspond à celui de L auquel on a ajouté la valeur de son indice. Par exemple, si L=[1, 4, 8, 5], M =[1, 5, 10, 8] = [1+0, 4+1, 8+2, 5+3]

3. Allez dans l'éditeur pour traiter cet exercice :

   1) À l'aide de la définition en compréhension des listes, créer la liste L des inverses de tous les carrés des nombres entiers compris entre 1 et 1000000.

   2) En utilisant sum , calculer la somme S de la liste L puis donner la valeur de la racine carrée de $6*S$.

   3) Quelle semble être la limite de $\sum_{k=1}^{n}\frac{1}{k^2}$ quand $[n\to+\infty]$.

4. Utiliser une liste en compréhension pour obtenir la liste de tous les mots de 2 lettres formés à partir de "ABCD". On doit récupérer ['AA', 'AB', 'AC', 'AD','BA', 'BB', 'BC', 'BD','CA', 'CB', 'CC', 'CD','DA', 'DB', 'DC', 'DD']
   Indication : utilisez deux boucles **for** imbriquées.

5. Créer une fonction **secret**(C) qui prend en argument une chaîne de caractère C et renvoie une liste contenant :
   •comme premier élément le premier caractère de C,
   •comme deuxième élément le dernier caractère de C,
   •comme troisième élément la longueur de C.

   Par exemple, **secret**("BCPST") doit renvoyer la liste ["B","T",5]

6. Créer une fonction **censure**(mot) qui prend comme argument un (gros) mot, et remplace toutes les lettres par * sauf la première et la dernière.

   Par exemple, **censure**("bachibouzouk") doit renvoyer la chaîne de caractères "b**********k"
7. Ecrire une fonction **listeentier**(n) qui étant donné un entier n renvoie la liste de ses chiffres dans son écriture décimale.
   Ainsi listeentier(452) doit renvoyer la liste [4,5,2].
8. Ecrire une fonction **sommedeschiffres**(n) qui étant donné un entier n renvoie la somme de ses chiffres dans son écriture décimale.

   Par exemple, **sommedeschiffres**(452) doit renvoyer 11 (=4+5+2)

9. Ecrite une fonction **retourne**(mot) qui étant donné la chaîne de caractère mot renvoie une chaîne de caractères correspondant à celle de mot lue de droite à gauche.

   Par exemple **retourne**("par") doit renvoyer "rap"
10. Ecrire alors une fonction **palindrome**(mot) qui renvoie **True** si la chaîne de caractère mot est un palindrome et **False** sinon.

    Par exemple **palindrome**("Laval") renverra **True**.

.