TP n°3 : Listes et chaînes de caractères

Instruction générale : hormis pour les exercices corrigés collectivement, vous ferez valider votre travail par l'enseignant.

1 Listes

Une liste est une donnée de type dit **construit** ; ce n'est pas un type simple comme les booléens, entiers et les flottants (et dans une certaine mesure les chaînes de caractères). Une liste contient une série de valeurs. En Python, ces valeurs peuvent être de types différents – des entiers et des flottants par exemple – et on peut même avoir des listes de listes. Elle est symbolisée par des crochets : `[]` et ses différents éléments sont séparés par des virgules ; le type est `list`. Enfin les listes sont **modifiables**.

Exemples :

```
listeVide=[]
listeEntiers=[0,3,7,4]
listeMixte=[2.2,5,"a",True]
listeListes=[[1,2],[3,4]]
```

Une liste est indexée, autrement dit ses éléments sont numérotés. Attention ! En Python, la numérotation commence à 0 donc pour une liste de taille 10, le numéro du dernier élément est 9 et non 10. Pour accéder à l'élément numéro `num` d'une liste en Python, il suffit d'utiliser la commande `liste[num]` (à condition que $num < taille(liste) - 1$). La taille d'une liste est donnée par la commande `len(liste)` (`len` vient de length qui signifie longueur en anglais).

Exemples : en reprenant l'exemple précédent, on a `listeMixte[0]=2.2`, `listeEntiers[3]=4` et `listeListes[1]=[3,4]`.

Exemples : les deux programmes suivants permettent de parcourir et d'afficher tous les éléments d'une liste donnée de deux façons différentes.

```
for element in liste:
    print(element)
```

```
indice=0
while indice<len(liste):
    print(liste[indice])
    indice+=1
```

La première méthode parcourt la liste à travers ces éléments tandis que la seconde le fait à travers les indices de ceux-ci. Toutefois, si la première paraît plus simple, notamment à écrire, dans les faits elle doit utiliser une méthode de parcours équivalente à la seconde, i.e. par les indices.

Exercice 1. Que donne les instructions suivantes lorsqu'on les tape dans l'interpréteur Python ?

Instruction	Résultat et commentaire
`liste1=[2,4]`	
`len(liste1)`	
`liste1[0]`	
`liste1[0]=3`	
`liste1[2]`	
`liste1.append(3)`	
`liste1[2]`	
`del liste1[1]`	
`liste1.remove(3)`	
`liste2=list(range(4))`	
`liste3=liste1+liste2`	
`liste3*2`	
`liste3.clear()`	
`liste4=liste2`	
`del liste2[:]`	
`liste4`	

Exercice 2.

1. Créer de trois façons différentes une liste contenant tous les entiers de 100 à 200 compris.
2. Créer de trois façons différentes une liste d'entiers partant de 42 et allant à 84 dont les valeurs augmentent de deux en deux : 42, 44, 46,..., 82, 84.
3. Créer de trois façons différentes une liste d'entiers partant de 90 et allant à 30 dont les valeurs diminuent de trois en trois : 90, 87, 84, 81,..., 33, 30.

Exercice 3.

1. Créer une liste contenant 100 fois l'entier 0 d'au moins trois façons différentes.
2. Créer une liste de taille 100 dont tous les éléments d'indice pair sont nuls et tous ceux d'indice impair valent 1 d'au moins quatre façons différentes. Par exemple, pour une liste de taille 4, on aurait [0,1,0,1].

Exercice 4. Écrire un algorithme puis le programmer prenant en entrée une liste de nombres et déterminant la moyenne de la liste. On testera l'algorithme avec plusieurs listes.

Exercice 5.

1. Écrire un algorithme puis le programmer prenant en entrée une liste de nombres et déterminant le minimum de la liste ainsi que le nombre de fois où il est atteint et ses positions dans celle-ci. On testera l'algorithme avec plusieurs listes.
2. Faire de même pour trouver le maximum de cette liste.

Exercice 6. Écrire un algorithme puis le programmer prenant en entrée une listes d'entiers et en supprimant tous les nombres pairs. Même question avec le nombres impairs. On testera l'algorithme avec plusieurs listes.

Exercice 7.

1. Écrire un algorithme puis le programmer prenant en entrée une liste contenant des entiers, des flottants, des chaînes de caractères et des booléens puis comptant le nombre d'éléments de chaque type.
2. Même question sauf que le programme devra supprimer tous les éléments d'un type choisi par l'utilisateur.
3. Même question sauf que le programme devra changer tous les booléens en leur contraire.

On pourra tester les programmes ci-dessus avec la liste suivante :

[-2.45,3.14,5.3,-0.3,True,False,True,True,3,4,5,6,"a"," ",",","2"]

Indication : pour tester le type d'une variable `var`, on peut utiliser l'un des deux tests suivants de façon équivalente (pour savoir si c'est un entier par exemple) :

— `type(var) is int`, — `type(var)==int`.

2 Chaînes de caractères

Les chaînes de caractères peuvent considérer comme des listes de caractères. Elles sont déclarées à l'aide de guillemets simples ou doubles. Les différents caractères d'une chaîne sont indexés et comme pour les listes leur numérotation commence à 0. Toutefois, contrairement aux listes, les chaînes de caractères ne peuvent être modifiées par leur numéro de caractères comme nous le verrons plus bas.

Exemples :

— On considère la chaîne de caractères `texte="abc"`, on a `texte[0]="a"`, `texte[1]="b"` et `texte[2]="c"`.

— Les deux programmes suivants permettent de parcourir et d'afficher tous les caractères d'une chaîne donnée de deux façons différentes.

```
for carac in chaine:
    print(carac)
```

```
indice=0
for indice<len(chaine):
    print(chaine[indice])
    indice+=1
```

Il existe des caractères spéciaux donnant des résultats particuliers lors de l'utilisation de `print()` sur les chaînes de caractères.

`\n`	effectue un retour à la ligne
`\t`	effectue une tabulation
`\'`	donne une apostrophe
`\"`	donne une guillemet

On peut aussi utiliser astucieusement les guillemets simples et doubles. Par exemple, en déclarant la chaîne de caractères `"J'y suis, t'es où"` avec des guillemets doubles, les simples comptent comme des apostrophes ; tandis qu'en déclarant la chaîne de caractères `'le mot de passe est "sorbet citron"'` avec des guillemets simples, les doubles comptent comme des guillemets de texte.

Exercice 8. Que donne les instructions suivantes lorsqu'on les tape dans l'interpréteur Python ?

Instruction	Résultat et commentaire
`print('a\n b')`	
`print('a\t b')`	
`mot1='a', mot2='b'*2`	
`mot2='c'`	
`mot2[0]='b'`	
`mot1<mot2`	
`mot3=mot1+mot1`	
`mot4=mot1+mot2`	
`mot3==mot4`	
`mot4>mot3`	
`'yoda'.upper()`	
`'DARK VADOR'.lower()`	
`'star wars'.capitalize()`	
`ligne=input("Écrire une phrase : ")`	
`mots1=ligne.split()`	
`mots2=input("Écrire une phrase : ").split()`	
`mots3='1,2,3,4'.split(",")`	
`str(3)`	
`int('2')`	
`ord('a')`	
`chr(98)`	

Exercice 9. Écrire trois programmes prenant en entrée une lettre puis donnant en sortie cent fois cette lettre en minuscule sur une ligne et cent fois cette même lettre en majuscule sur une autre ligne.

Exercice 10. Écrire un programme prenant en entrée une phrase puis donnant en sortie un mot sur sur deux de cette phrase à partir du premier mot. Même chose à partir du second mot.

Exercice 11. Écrire un programme prenant en entrée une phrase puis donnant en sortie le nombre de mots de plus de six lettres de cette phrase. Même chose avec les mots de moins de quatre lettres.

Exercice 12. [Comptage] Écrire un algorithme puis le programmer prenant en entrée une chaîne de caractères et un caractère précis (par exemple "a") puis comptant le nombre d'occurrences et donnant la fréquence de ce caractère dans la chaîne. On testera l'algorithme avec plusieurs chaînes de caractères.

Exercice 13. [Suppression] Écrire un algorithme puis le programmer prenant en entrée une chaîne de caractères et un caractère précis (par exemple "a") puis supprimant ce caractère de la chaîne. On testera l'algorithme avec plusieurs chaînes de caractères.

Exercice 14. [Remplacement] Écrire un algorithme puis le programmer prenant en entrée une chaîne de caractères et deux caractères précis (par exemple " " et "_") puis remplaçant ce le premier caractère par le second de dans chaîne. On testera l'algorithme avec plusieurs chaînes de caractères.

Exercice 15. [Miroir] Écrire un algorithme puis le programmer prenant en entrée une chaîne de caractères et inversant l'ordre de tous les caractères (le dernier devient le premier et réciproquement). Par exemple, "loutre" donnerait "ertuol" et "kayak" donnerait encore "kayak". On testera l'algorithme avec plusieurs chaînes de caractères.

Exercice 16. [Chiffrement de César] Le chiffrement de César consiste à décaler toutes les lettres de l'alphabet. Par exemple, si le A devient J, alors B deviendra K, C deviendra L, etc. Ainsi le mot BAC deviendra KJC. Le changement A en J est appelé clé de chiffrement, il permet de coder le message et aussi de le décoder (chiffrement symétrique).

1. Écrire un programme prenant en entrée un texte et deux caractères constituant la clé de chiffrement puis chiffrant le texte selon cette clé.
2. Écrire un programme déchiffrant un message connaissant sa clé de chiffrement.
3. Écrire un programme vous permettant de déchiffrer un message sans en connaître la clé de chiffrement.

3 Ressources supplémentaires

— Les cours sur Python d'OpenClassRooms
— Le cours du W3 Schools
— Vidéo sur les listes
— Vidéo sur les chaînes de caractères