TD 10 — Listes et chaines de caractères

Le langage Python propose de nombreuses manières de manipuler des données structurées. Dans ce TD, nous allons étudier les listes. Ces structures sont *itérables*, ce qui signifie qu'il est possible de manipuler les éléments qu'elles contiennent avec une boucle for.

1 Un peu de théorie

Les listes permettent de réunir des… listes de variables (quel que soit leur type). Les éléments dans une liste sont *mutables* (ils peuvent changer de valeur), munis d'un indice (le premier indice étant `0`). Les listes elles-mêmes sont des objets mutables.

1.1 Définir une liste

On peut définir une liste de plusieurs façons différentes. La première consiste à donner de manière exhaustive tous ses éléments :

```python
ma_liste = [3, 'a', 12.5', "Bonjour']
print(ma_liste)
print(f'"La longueur de la liste est {len(ma_liste)})
```

On crée la liste vide avec l'instruction `liste_vide = []`. On construit une liste d'entiers avec l'instruction `L = list(range(1, 10))`. Attention, cela construit la liste des entiers de `1` à `9`.

Enfin, on peut construire une liste par *compréhension* : c'est une syntaxe agréable et très efficace, qu'il convient d'utiliser dès que possible.

```python
L1 = [n**2 for n in range(1, 11)]             # Quels sont les éléments de L1 ?
L2 = [17*n for n in range(1, 500) if n%17==0] # Quels sont les éléments de L2 ?
```

Enfin, on ajoute un élément `e` en dernière position d'une liste `L` au moyen de l'instruction `L.append(e)`.

Différence entre liste et tableau

Les listes et les tableaux ne sont pas les mêmes objets en informatique. Les tableaux sont formés de variables *toutes de même types* et *contiguës* dans la mémoire. Une liste peut contenir des valeurs de plusieurs types, et elles ne sont pas stockées côte-à-côte en mémoire.

En langage C, il n'existe pas de structure native pour les listes : on ne manipule que des tableaux. C'est le contraire en Python, qui fournit une structure de listes mais pas de tableaux.

1.2 Accéder aux éléments d'une liste

Les éléments d'une liste sont *indexés*. On accède à l'élément d'indice `i` dans une liste `L` avec la syntaxe classique `L[i]`. Les indices commencent à partir de `0`.

Il est très facile de parcourir les éléments d'une liste avec une boucle `for` :

```python
for x in L:
    print(x)
```

Notez que l'on n'utilise pas ici les indices des éléments. Si l'on souhaite utiliser les indices, on procède comme suit :

```python
for i in range(0, len(L)):
    print(l[i])
```

On peut facilement *extraire une sous-liste* d'une liste donnée, comme le montre le code suivnat :

```python
L1 = ["a","b","c","d","e","f"]
L2 = ma_liste[:3]
print(L) # Qu'affiche cette ligne ?
```

1.3 Opérations élémentaires sur les listes

Comme les listes sont des objets mutables, contenant des éléments mutables, il est possible d'effectuer toutes les opérations élémentaires usuelles : ajout, suppression d'éléments, etc. Il n'est pas possible [1] de connaitre toutes les opérations. Consultez la documentation pour en savoir davantage : Compléments sur les listes. On donne ci-desous quelques opérations couramment utilisées :

```python
L = ['a', 'b', 'c', 'd', 'e', 'f']
M = ['g', 'h', 'k']
N = L + M          # Que fait cette ligne ?
L.append('g')      # Ajoute 'g' en dernière position.
L2 = L[2:5]        # L2 = ['c', 'd', 'e']

x = L.pop()       # Supprime l'élément en derniere position et le stocke dans x.
y = L.pop(3)      # Supprime l'élément d'indice 3 et le stocke dans y.
del L[2]           # Supprime l'élément d'indice 2 sans renvoyer la valeur supprimée.
L.remove('e')      # Supprime le premier élément ayant la valeur 'e', sans le renvoyer
L.insert(3, 'z') # Insère 'z' en position d'indice 3
```

Exercice 1. Intégrer les lignes de codes ci-dessus dans Thonny et les tester pour bien comprendre leur fonctionnement.

Lien entre listes et chaines de caractères

En python, les chaines de caractères se comportent comme des listes de `chr`. Toutes les opérations qui existent sur les listes existent aussi sur les chaines de caractères. Il existes des opérations supplémentaires, spécifiques aux chaines de caractères (telles que la mise en majuscules/minuscules). Lire la documentation sur les chaines de caractères pour en savoir plus.

1. Et pas utile !

2 Exercices d'application

Travail à rendre

Déposer sur EPREL un fichier `r101_td10_<NOMDEFAMILLE>.py`, contenant les réponses aux exercices, convenablement réparties dans des fonctions. Si vous avez besoin de créer des fonctions auxiliaires, ajoutez-les à votre code.

Exercice 2. On considère la liste `[8, 3, 12.5, 45, 15.5, 52, 1]`. Trier cette liste par ordre croissant, sans utiliser la fonction `sort()`. On pourra utiliser librement `min()`, `append()` et `remove()`.

Exercice 3. Écrire une fonction qui reçoit une liste en paramètre, supprime les doublons, la trie par ordre croissant et renvoie la liste obtenue.

Exercice 4. **1)** Écrire une fonction qui reçoit en paramètre une liste `L` de taille n, contenant des entiers appartenant à $[\![1, n]\!]$, et qui renvoie `True` si `L` contient une et une seule fois chaque entier entre 1 et n, `False` sinon.

2) Chercher maintenant une solution qui ne nécessite de parcourir la liste `L` qu'une seule fois.

Permutations

En mathémétiques, une liste qui contient une fois et une seule chaque entier entre entre 1 et n représente ce que l'on appelle une *permutation* (cf. l'article de Wikipedia correspondant).

Exercice 5. Écrire une fonction qui reçoit en paramètre une chaine de caractères contenant un nom commun, et affiche son pluriel.

- Si le mot se termine déjà par un -s, le pluriel et le sungulier sont identiques.
- Si le mot se temine par -al, le pluriel se forme en -aux (on ne prendra pas en compte les exceptions).
- Certains mots en -ou ont un pluriel en -x au lieu d'un pluriel en -s (bijou, caillou, chou, hibou, joujou, genou, pou).
- Dans les autres cas, le pluriel s'écrit avec un -s.

Exercice 6 (Conjugaison !)**.** Écrire une fonction qui reçoit en paramètre un verbe du premier groupe à l'infinitif, et le conjugue au présent de l'indicatif. Si le verbe n'est pas du premier groupe, afficher un message d'erreur. On prendra en compte le cas particulier des verbe en -ger (manger, nager, etc.) à la première personne du pluriel.

Exercice 7 (Analyse linguistique)**.** Nous allons écrire un programme qui, étant donné un texte, détermine dans quelle langue il a la plus grande probabilité d'avoir été écrit. On se limite aux langues latines. Cette méthode est basée sur une analyse statistique : d'une langue à une autre, la fréquence d'apparition des lettres n'est pas la même. On peut donc, en étudiant la fréquence d'apparition de chaque lettre dans un texte, estimer la langue dans laquelle il a été écrit.

Pour simplifier, on ne travaille qu'avec des textes écrits en lettres majuscules et sans diacritiques.

1) Écrire une fonction `nombre_lettres(texte)` qui compte et retourne le nombre total de lettres présentes dans le texte. Ne compter ni les espaces, ni la ponctuation.

2) Écrire une fonction `occurrences_lettre(texte, lettre)` qui compte et retourne le nombre d'occurrences de la lettre dans le texte.

3) Écrire une fonction `pourcentage_lettre(texte, lettre)` calcule et retourne la fréquence d'apparition de la lettre dans le texte. *Rappel — La fréquence f d'une lettre est donnée, en pourcentage, par la formule $f = \frac{n}{d} \cdot 100$ où n est le nombre d'occurrences de la lettre et d le nombre total de lettres.*

4) Écrire une fonction `affiche_table_frequences(texte)` qui affiche, de manière synthétique, la fréquence de chaque lettre de l'alphabet (majuscule) dans le texte.

5) Sur Wikipedia, on trouve la fréquence des lettres selon la langue. Écrire une fonction qui, étant donné un texte passé en paramètre, détermine la langue dans laquelle il a été écrit.

6) Application — Tester votre programme avec les quatre textes suivants (les lettre des mots ont été volontairement mélangées).

Texte 1 `MAIER BERACUO RSU NU REBRA PRCEEH EIANTT NE ONS EBC NU GAOFREM EIMATR RERNAD APR L RDUOE LAHECLE UIL TTNI A EUP SREP EC LGNGAEA TE RBONUJO ERMNOUSI DU UBRACEO QUE OVSU EEST LIJO UQE OUVS EM MSZELBE BAEU ASNS MIERNT IS RVETO AGRAME ES PRARPTOE A OEVTR AMGUPLE VUOS SEET EL PNIHXE DSE OSHET ED CSE BIOS A ESC MSOT LE OUBRCEA NE ES ESTN ASP DE IEJO TE OUPR ERRNOTM AS BELEL XOVI IL OREVU NU RGLEA ECB ILESSA EBOMTR AS PIOER EL NRDAER S EN ISIAST TE ITD MNO NOB EUSRMNOI NRPEEAZP QEU UTOT EUTLRFTA IVT XUA SPNEDE DE UECIL UQI L TECEOU TECET NEOCL VATU BNEI UN GMAEORF SNAS TUOED LE EOABURC OHENTXU TE NSCOFU UJRA SMIA UN EPU TRDA UQ NO EN L Y ARRPEIDNT ULSP`

Texte 2 `WRE TREITE SO TSPA CUDHR AHNCT UND WIND SE STI RED AEVRT MTI ESEIMN IDNK RE ATH END NEABNK WLOH IN EMD AMR ER AFTSS HIN IHSERC RE AHTL HIN MRWA EINM SHNO SAW SRTIBG UD SO NGBA DNEI EIHSGTC ESISTH RAETV UD DEN LERNIOKG NITHC NDE LOENINKGRE TIM OKRN UDN CHWFSEI NEIM NSOH ES STI IEN BIFTRLSEEEN DU BILESE IKDN OMKM EHG MIT MIR RAG ECHNOS EPELSI EIPSL IHC ITM RDI HNCMA BEUTN MBLUNE DINS NA DEM TNDRAS NMIEE UTETMR AHT CAMHN UDNGEL GDAWEN MIEN EATRV MENI VEART DUN OSTHER DU CINTH SAW KNNOEIREGL RIM ILEES PRSTVRCIEH ISE IHGRU BEEILB RIGUH MNEI KNDI NI RDNEUR NATBRLET STAESUL EDR WNID`

Texte 3 `DSNOACAIF ORP ANU DAEDALRI DNAAEIMTI EQU NNCOSETE EL RSTEOUL SMA AACTFAITNS UQE LE TSVAO OINSRVUE DE US ANIGIICANOM EIORDP TOOD RTEIENS RPO LE ITOABOLRROA ED QIUAMALI USOP A NSSRCAEAD LA TMREAAI NXTADAUEE ROP GOARLS EMESS DE NNAMICLUIAPO Y LOVOIV A RES LE RHMEOB EOMDNEERPRD DE LOS RSOPMRIE OMTSIPE UEQ CIIDADE LE RTDAAOZ ED LSA CELSAL Y LA NICOIOPS ED LAS UESVNA SSACA Y ES ITRMNEEOD QEU AERFU EL UEQIN IIIRDEGAR LA NAIORTREICP DE AL RRTEIA`

Texte 4 `IMTRUESMME DNA TEH LNGIIV SI EYAS SIFH REA GJPNUIM DNA HET TTNOCO IS GHIH OH OUYR DDADY SI IRHC DAN ROUY MA SI DOGO GKOILON OS USHH LTLIET BBYA NDOT OUY CYR NEO OF HESET GNSRONIM YUO RE NANGO SIER PU SNIGING NAD OULLY EPADRS YUOR GINSW DAN LYOLU KATE OT HET KSY TUB ITLL TATH MGNIRNO EREHT NATI INTGOHN ACN AHMR OYU TWIH DADYD NDA MYMMA NSTIDGAN YB`

Copie de listes

Copier une liste dans une autre n'est pas si évident en Python. Il existe *deux manières différentes* de copier des listes.

La premire méthode, appelée *copie en surface* (*shallow copy*) est rapide et suffisante pour les liste simples :

```python
L1 = [1,2,3]
L2 = L1[:]
print(f"L1 = {L1}") # [1, 2, 3]
print(f"L2 = {L2}") # [1, 2, 3]
L2[0] = 18
print("Après modification de L2, L1 reste inchangée :")
print(f"L1 = {L1}") # [1, 2, 3]
print(f"L2 = {L2}") # [18, 2, 3]
```

En revanche, la copie en surface ne suffit pas pour les listes plus compliquées, par pour les liste de listes :

```python
L1 = [[1,1],2,3]
L2 = L1[:]
print(f"L1 = {L1}") # [[1, 1], 2, 3]
print(f"L2 = {L2}") # [[1, 1], 2, 3]
L2[0][0] = 18
print("Après modification de L2, L1 aussi a changé :")
print(f"L1 = {L1}") # [[18, 1], 2, 3]
print(f"L2 = {L2}") # [[18, 1], 2, 3]
```

On utilise une deuxième méthode, appelée *copie en profondeur* (*deep copy*) :

```python
import copy
L1 = [[1,1],2,3]
L2 = copy.deepcopy(L1)
print(f"L1 = {L1}") # [[1,1],2,3]
print(f"L2 = {L2}") # [[1,1],2,3]
L2[0][0] = 18
print("Après modification de L2, L1 reste inchangée :")
print(f"L1 = {L1}") # [[1,1],2,3]
print(f"L2 = {L2}") # [[18,1],2,3]
```

3 Un mot sur les tuples et les ensembles

Python fournit deux autres structures, proches des listes, mais qui ont des spécifiqucations techniques.

Les tuples sont proches proches des listes. La seule différence, outre la syntaxe, est qu'ils sont *non mutables*, ce qui signifie que les valeurs ne peuvent pas être modifiées. On utilise cette

structure lorsque l'on souhaite manipuler des n-uplets en garantissant que les valeurs resteront inchangées. Les éléments sont indexés (à partir de l'indice `0`).

Les ensembles (*sets*) sont l'implémentation Python de la notion mathématique d'ensemble[2]. Un ensemble en Python ne peut pas contenir de doublons, les éléments ne sont pas indexées et les élements sont mutables.

3.1 Les tuples

Exécuter, analyser et comprendre le code ci-dessous. On pourra le modifier librement pour l'étudier.

```python
t = (1, 2.2, 'a') # Un tuple
print(t)          # Affiche 1
print(t[0])       # Lève une exception : les tuples sont non mutables
for e in t:
    print(e)
```

Exercice 8. Créer un tuple *par compréhension* contenant le carré des entiers de 1 à n, n étant saisi au clavier.

3.2 Les ensembles

En Python, les ensembles se notent entre crochets : `s = {1, 2, 3}` crée un ensemble contenant trois éléments. On crée l'ensemble vide avec l'insrtruction `s = set()`.

Copier et exécuter le code suivant pour comprendre les instructions :

```python
ens1 = {1, 2, 5, 3, 1, 2, 2, 3}
ens2 = {x%13 for x in range(1000)}
ens3 = {2, 5, 6, 7}
print(ens1)        # Les doublons sont éliminés. Notez l'ordre d'affichage
print(ens2)
ens1.add(6)
ens1.add(2)        # Rien en se passe, 2 est déjà dans s1
print(ens1 | ens3) # Union de deux ensembles
print(ens1 & ens3) # Intersection de deux ensembles
print(ens1 - ens3) # Que fait cette instruction ?
print(ens1 < ens3) # Que fait cette instruction ?
```

2. Cf. R1.06 Mathématiques discrètes.