Les listes

Faites une copie avec la commande ci dessus "File>Make a Copy...", et renommez la en cliquant sur le titre (à côté de jupyter TD2...), par exemple en rajoutant votre nom.

Les listes sont utilisées en permanence en programmation. C'est un outil absolument fondamental, qui vous prendra un peu plus de temps à maitriser que les notions précédentes. On dit "listes" en Python, on dit aussi "tableaux". Une liste est une suite de valeurs, séparées par des virgules, entre crochets. Les valeurs sont indicées (ou indexées) de 0 à *n* - 1, où *n* est la longueur de la liste. On accède à un élément de la liste par son indice (ou index).

In [ ]:
```
liste = [42,23,'oui',"la réponse d"]
print ("la liste comporte ",len(liste)," éléments")
for i in range(4):       # parcours de la liste par ses indices
    print("l'élément d'indice",i,"vaut",liste[i])
```

Dans l'exemple précédent, on constate que : *(compléter, appeler le professeur si nécessaire)*

- pour trouver la longueur d'une liste on utilise la fonction...
- le premier indice dans une liste est...
- le dernier indice dans une liste est...
- la fonction `len(liste)`...
- l'instruction `liste[i]`...

Remarque : les éléments de la liste peuvent être de type différent. On évite cependant de mélanger les types dans une liste, d'autres types de variables sont plus efficaces pour cela, comme les dictionnaires que l'on verra ultérieurement.

Si l'on veut accéder à plusieurs éléments consécutifs de la liste, on utilise la syntaxe `[a:b]`, où a et b sont les bornes. Bien comprendre le fonctionnement sur les bornes `a` et `b`.
Testez les commandes ci-dessous. Commentez le programme afin de préciser l'effet de chaque instruction.

In [ ]:
```
liste = [i for i in range(0,11)]     # une manière simple de cré
er une liste, dite construction par compréhension
print(liste)
print(liste[0])      # extraction de...
print(liste[:])      # extraction de...
print(liste[:3])     # extraction de...
print(liste[3:])     # extraction de...
print(liste[3:7])    # extraction de...
print(liste[-1])     # extraction de...
print(liste[-4:])     # extraction de...
```

Copie de listes

Tester le programme suivant. Le résultat obtenu vous paraît-il normal ?

In [ ]:
```
a = [1,2,3,4]
b = a        # on copie la liste a dans b
print("la liste b est :",b)
a[0]="bonjour"        # on modifie le premier element de a
print("la liste b est :",b)
```

Que s'est-il passé ?
Une variable est stockée en mémoire à une adresse donnée (comme une adresse postale, sauf que dans un ordinateur c'est un en gros un numéro de ligne). Par exemple, on suppose que la variable `a` est un entier, et que la variable `b` est égale à `a`. Pour l'ordinateur, ça ne sera pas `a` mais "adresse #FE240AC6", ni `b` mais adresse #F000001. Si un programme veut changer la valeur de `a`, il va acceder à l'adresse #FE240AC6 et en modifier le contenu. L'adresse #F000001, soit la variable `b`, sera inchangée.
Si maintenant la variable `a` est une liste, c'est une suite de valeurs stockée à partir d'une adresse comme ci-dessus, toujours par exemple #FE240AC6. L'instruction `b = a` ne va pas copier la liste à partir d'une autre adresse, mais signifie que `b` pointe aussi sur #FE240AC6. Toute modification de la liste `a` entrainera aussi celle de la liste `b` ! Le commentaire dans le programme précédent `# on copie la liste a dans b` peut être modifié par `# on copie l'adresse de la liste a dans b`.
On utilise les méthodes d'extraction d'éléments d'une liste, que l'on a vu ci-dessus, pour les copier :

In [ ]:
```
a = [1,2,3,4]
b = a[:]        # on copie la liste a dans b, première méthode
print("Avant modification, la liste a est :",a)
print("Avant modification, la liste b est :",b)
print()
a[0]="bonjour"        # on modifie le premier element de a
print("Après modification de a[0], la liste a est :",a)
print("Après modification de a[0], la liste b est :",b)
print()
c = a.copy()        # on copie la liste a dans c, deuxième metho
de
print("Avant modification de a[1], la liste c est :",c)
print()
a[1]="à tous"        # on modifie le deuxième element de a
print("Après modification de a[1], la liste a est :",a)
print("Après modification de a[1], la liste c est :",c)
```

Remarque : pour vraiment être convaincus, je vous conseille de copier/coller le programme précédent ci-dessous, en remplaçant les lignes `b = a[:]` par `b = a` et de même `c = a.copy()` par `c = a`.

Création de listes et de tableaux

Première méthode : rentrer des éléments un par un

Dans les exemples précédents, les listes étaient données dans le corps du programme. On peut aussi créer une liste en rentrant les éléments un par un.

- Soit on connait à l'avance la longueur de la liste, et on modifie les éléments un par un comme dans cet exemple qui crée une liste de nombres aléatoires (à conserver dans vos archives) :

In [ ]:
```
from random import *

"""
Fonction de création d'un tableau de nombres entier aléatoires

@param : un entier positif ou nul n
@result : un tableau de longueur n, rempli aléatoirement d'entie
rs compris entre 1 et 50
"""

def tableauAlea(n):

    tab = [0]*n        # creation d'un tableau vide de longueur
n
    for i in range(len(tab)):         # remplissage du tableau p
ar des valeurs
        tab[i] = randrange(1,50)      # aléatoires entre 1 et 50
    return(tab)

print(tableauAlea(10))  # exemples
print(tableauAlea(0))
```

- Soit on ne connaît pas à l'avance le nombre d'éléments de la liste, on les ajoute alors un par un à l'aide la méthode `liste.append(element)`, qui rajoute à la fin de `liste` l'item `element`. Exemple ci-dessous, également à conserver.

In [ ]:
```
liste = []      # creation d'une liste vide
continuer = True
while continuer:
    elt = int(input("rentrer un entier non nul (0 pour arreter)
:"))
    if elt == 0:
        continuer = False
    else:
        liste.append(elt)
print(liste)
```

Exercice (suite de Syracuse) : modifier le programme de création d'une liste aléatoire pour qu'il crée une liste de longueur 100 dont les éléments sont :

- un nombre entier aléatoire entre 1 et 50 pour `liste[0]`
- le nombre calculé à partir de l'élément précédent (d'index `i - 1`) de la liste suivant la règle:
  - si l'élément précédent est impair, alors on le multiplie par 3 et on ajoute 1
  - si l'élément précédent est pair, alors on le divise par 2

Il peut être nécessaire de faire une liste plus longue pour obtenir un résultat intéressant, suivant la valeur de `liste[0]`

Faire tourner le programme plusieurs fois, vous constaterez ce qui est un grand mystère des mathématiques de notre époque.

Deuxième méthode : remplir la liste en entier directement

La méthode ci-dessous : `liste = [i for i in range(0,11)]` est très pratique et rapide pour créer rapidement une liste de valeurs suivant ce que l'on appelle une suite arithmétique.

In [ ]:
```
liste = [i for i in range(0,11)]
print(liste)
```

Exercice : créer avec cette méthode la liste des multiples de 3 compris entre -6 et 24, vérifier avec un affichage.

Exercice : créer en une seule ligne de code une liste de 20 nombres aléatoires compris entre -10 et 100.

Fonctions et méthodes de base sur les listes

Longueur et ajout d'un élément.

La fonction `len(liste)` donne le nombre d'éléments d'une liste.
La méthode `liste.append(element)` rajoute un élément en fin de liste. *Remarque sur le vocabulaire* :

- `len()` est une **fonction**, on doit donc affecter le résultat dans une variable : `n = len(liste);`
- `.append()` est une **méthode**. Elle modifie l'**objet** donné avant le point. Il n'y a pas d'affectation à faire, l'instruction s'utilise seule.

D'autres méthodes et fonctions utiles.

De nombreuses méthodes existent sur les listes, citons par exemple:

- `liste.sort()` trie la liste dans l'ordre croissant. Les éléments doivent être de nature comparable
- `liste.reverse()` : inverse l'ordre des éléments.
- `liste.index(élément_de_liste)` : renvoie l'indice dans la liste d'un élément donné.
- `liste.remove(élément_de_liste)` : enlève un élément de la liste. Si l'élément est présent plusieurs fois, seule la première occurence sera retirée.
- `liste.insert(index, élément_de_liste)` : ajoute un élément à la liste à l'index précisé.
- `min(liste)` : donne le minimum d'une liste.
- `max(liste)` : *compléter*
- `liste1.extend(liste2)` : rajoute `liste2` à la fin de `liste1`. Il est équivalent de faire `liste1 = liste1 + liste 2`

Tester ces méthodes dans la cellule ci-dessous

Remarque : dans la majeure partie des exercices en début d'année, l'utilisation de ces méthodes est interdite ; en effet elles suppriment la visée pédagogique desdits exercices.

In [ ]:
```
from random import *
liste = [randint(1,50) for i in range(0,11)]
print(liste)
liste.sort()
print(liste)
```

Complément : les chaines de caractères

Un chaine de caractères est une liste.
Exercice : dans une chaine (d'au moins 15 caractères) que vous vous donnez:

1. Extraire la première lettre
2. Extraire la sous-chaine allant de la 3ème à la 10ème lettre.
3. Extraire la dernière lettre. L'instruction doit fonctionner quelque soit la longueur de la chaine, sans avoir à changer le code.
4. Extraire les dix dernières lettres, avec deux méthodes différentes (indices positifs ou négatifs).

Une première approche des tableaux et matrices

Un tableau est en général considéré une liste de listes (de listes de listes... etc, autant que l'on veut). On rappelle qu'il y a ambiguïté, vu que les imprécisions de langage avec le python font que l'on confond tableau et liste. Une matrice est un tableau de listes de réels (ou de complexes), toutes les listes ayant la même longeur. OnUne matrice est dimension $a \times b$ si elle a a lignes et b colonnes. Tester :

In [ ]:
```
tab = [[1,2,3],[4,5,6],[7,8,9],[10,11,12]]
print("un affichage pas très beau :")
print(tab)
print("un affichage plus lisible, pas parfait cependant :")
for i in range(4):
    print (tab[i])
```

Quelle est la dimension de la matrice crée ci-dessus ?

Remarque : on reviendra plus longuement sur les tableaux et matrices ultérieurement

In [ ]:
```
tab = ['a','b','c','d','e','f']
print("la longueur de tab est :",len(tab))
tab.append("bonjour")
print(tab)
```

Exercice

En se basant sur l'exemple précédent, et sur la fonction de création d'une liste d'entier aléatoires, écrire une fonction qui crée une matrice de *n* lignes de *p* nombres aléatoires entre 12 et 45. *n* et *p* seront passés en paramètres.

Frederic Mandon (mailto:frederic.mandon@ac-montpellier.fr), Lycée Jean Jaurès - Saint Clément de Rivière - France (2015-2019)