1.4 Les séquences

NSI 1ère - JB Duthoit

Il est possible de "stocker" plusieurs grandeurs dans une même structure, ce type de structure est appelé une **séquence**.

Définition 1.1
De façon plus précise, nous définirons une séquence comme un ensemble fini et ordonné d'éléments indicés de 0 à n-1 (si cette séquence comporte n éléments).

Nous commencerons par étudier un premier "type" de séquence, les tuples, puis nous enchainerons sur les tableaux.

1.4.1 Les tuples en Python

```
mon_tuple = (5, 8, 6, 9)

type(mon_tuple)
```

Chaque élément du tuple est indicé.

☞ Retrouver comment accéder au premier élément, au second..etc...

⚠ dans les séquences les indices commencent toujours à 0 (le premier élément de la séquence a pour indice 0)

☞ Compléter le programme suivant :

```
mon_tuple = (Duthoit,Jean-Baptiste)
print(f"Mon nom est {mon_tuple[...]} et mon prénom {mon_tuple[...]}")
```

Définition 1.2
Un tuple est une séquence **immuable** (c'est à dire qui ne peut pas être modifiée), pouvant contenir plusieurs autres objets.

Exercice 1.48
Créer une fonction nommée distance, qui prend en argument deux tuples (Dans chaque tuple, se trouvent les coordonnées (x ;y) d'un point) et qui renvoie la distance entre ces deux points, dans un repère orthonormé.

Exercice 1.49
On considère un tuple de tuple :
(('Alim',17),('Pierre',34),('Sophie',19),('Angel',23))
Afficher 'bonjour Alim tu as 17 ans', et ainsi de suite pour les autres.

Exercice 1.50
Les commandes sum(tuple),len,min,max donnent respectivement la somme, le nombre d'éléments, le min et le max d'un tuple.
Ecrire une fonction qui renvoie le tuple composé du min, max, moyenne.

1.4.2 Les tableaux en Python

Création d'un tableau vide

Il existe deux moyens :

```
mon_tableau = list()
```

ou bien

```
mon_tableau = []
```

⚠ Python utilise le terme "list", mais il s'agit en réalité d'un tableau (tableau dynamique en réalité)

Création d'une tableau non vide

```
mon_tableau_1 = [1, 2, 3, 4]
mon_tableau_2 = [36, "voiture", 2.8, True]
mon_tableau_3 = [36, "texte", 2.8, mon_tableau_1]
```

Accès au éléments

```
mon_tableau_1[2] # pour obtenir 3
```

Modification de tableau

⚠ Ce que ne permet pas un tuple :-)

```
mon_tableau_1[2] = 120 # pour remplacer 3 par 120
```

Ajout d'un élément

```
mon_tableau = [1,2,3,4]
mon_tableau.append(5)
```

⚠ La méthode "append" est très utilisée et permet d'ajouter un élément **à la fin** du tableau.

Suppression d'éléments

```
mon_tableau = [1,2,3,4,5,6]
del mon_tableau[1] # suppression du second élément
```

Longueur d'un tableau

```
mon_tableau = [1,2,3,4,5,6]
len(mon_tableau) # affiche 6
```

Parcourir un tableau

```
mon_tableau = [1,2,3,4,5,6,7,8,9,10]
for element in mon_tableau:
        print(element)
```

1.4.3 Des tableaux et des chaînes

Des tableaux aux chaînes

```
mon_tableau = ['Bonjour','à','tous']
a = " ".join(mon_tableau)
```

☞ Vérifiez que a est un str

Des chaînes aux tableaux

```
ma_chaine = " Bonjour tout le monde !"
a = ma_chaine.split(" ")
```

☞ Vérifiez que a est du type list ☞ La méthode split() découpe donc la chaîne en fonction du paramètre donné.

Définition 1.3

Un tableau (list en Python) est une séquence **mutable** pouvant contenir plusieurs autres objets.

Exercice 1.51

Ecrire une fonction qui prend un tableau et un entier en argument et qui renvoie le tableau où l'entier a été ajouté à la fin

⚠ Remarquez bien que la liste a été modifiée en place.

Exercice 1.52

Écrire une fonction (sans paramètre) qui remplit un tableau avec les entiers de 0 à 20

Exercice 1.53

Ecrire une fonction(sans paramètre) qui remplit un tableau avec 20 entiers choisis au hasard entre 0 et 100

Exercice 1.54

Ecrire une fonction prenant en argument un entier n et qui remplit un tableau avec n entiers choisis au hasard entre 0 et 100

☞ Ne pas oublier : from random import *

Exercice 1.55

Ecrire un programme prend en argument un tableau d'entiers et qui retourne un autre tableau où chaque valeur a été doublée :

```
>>> a = [1,2,3,4,5,56]
>>> b = double(a)
>>> b
[2, 4, 6, 8, 10, 112]

>>> a
[1, 2, 3, 4, 5, 56]
```

Exercice 1.56

Ecrire une fonction qui prend en argument un tableau d'entiers d'au moins 2 éléments et qui retourne le tableau dans lequel on a échangé le premier et le dernier élément.

```
>>> a = [1,2,3,4,5,56]
>>> echange(a)
[56, 2, 3, 4, 5, 1]

>>> a
[56, 2, 3, 4, 5, 1]
```

Exercice 1.57

Écrire une fonction qui prend en argument un tableau d'entiers , et qui retourne un nouveau tableau avec les carrées des entiers.

Exercice 1.58

Écrire une fonction qui prend en argument un tableau d'entiers avec au moins deux éléments, et qui retourne le tableau avec les deux premiers termes supprimés.

Exercice 1.59

Écrire une fonction qui prend en argument un tableau d'entiers et qui retourne un tuple (n,p) où n est le nombre d'entiers négatifs et p le nombre d'entiers positifs.

Exercice 1.60

Écrire une fonction qui prend en argument un tableau contenant des chaines de caractères, et qui retourne une autre liste contenant les mots de plus de 5 caractères.

Exercice 1.61

Écrire une fonction Python nommée diviseurs qui prend en argument un nombre entier naturel non nul n et qui retourne la liste de ses diviseurs triés par ordre croissant.

Exercice 1.62

Programme une fonction rotation(liste) qui décale d'un rang tous les éléments d'une liste (le dernier élément devenant le premier). La fonction renvoie une nouvelle liste. Par exemple rotation([1,2,3,4]) renvoie la liste [4,1,2,3].

Exercice 1.63
Programme une fonction inverser(liste) qui renvoie une nouvelle liste avec les éléments dans l'ordre inverse. Par exemple inverser([1,2,3,4]) renvoie la liste [4,3,2,1].

Exercice 1.64
Programme une fonction supprimer_ rang(liste,rang) qui renvoie une liste formée de tous les éléments, sauf celui au rang donné. Par exemple supprimer_ rang([8,7,6,5,4],2) renvoie la liste [8,7,5,4] (l'élément 6 qui était au rang 2 est supprimé)

Exercice 1.65
Écrire une fonction qui permet d'écrire une liste d'entiers aléatoires qui s'arrête dès que la somme des nombres dépasse une certaine valeur. la fonction "makeList" prend en entrée 3 valeurs entières "deb", "fin" et "maxi" correspondant respectivement à la plus petite valeur des nombres aléatoires, à la plus grande et à la valeur à ne pas dépasser en faisant la somme de ces nombres.

1.4.4 Le sujet -délicat- des copies de tableaux

Étudiez soigneusement les lignes suivantes :

```
>>> x = [1, 2, 3]
>>> y = x
>>> y
[1, 2, 3]
>>> x[1] = -15
>>> x
[1, -15, 3]
>>> y
[1, -15, 3]
```

⚠ Vous voyez que la modification de x modifie y aussi ! Pour comprendre ce qui se passe, rendez-vous dans python Tutor !

Copie explicite de la liste initiale

```
>>> x = [1, 2, 3]
>>> y = list(x)
>>> x[1] = -15
>>> y
[1, 2, 3]
```

Deepcopy

⚠ Attention, l' astuce précédentes ne fonctionne que pour les listes à une dimension, autrement dit les listes qui ne contiennent pas elles-mêmes d'autres listes.

La méthode qui fonctionne à tous les coups :

```
>>> import copy
>>> x = [[1, 2], [3, 4]]
>>> x
[[1, 2], [3, 4]]
```

```
>>> y = copy.deepcopy(x)
>>> x[1][1] = 99
>>> x
[[1, 2], [3, 99]]
>>> y
[[1, 2], [3, 4]]
```