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

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...

 \triangle 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[...]}")
```

Propriété

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

• Exercice 1.53

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.

• On rappelle la formule qui permet de calculer la distance AB connaissant les coordonnées de A et de $B:AB=\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}$

• Exercice 1.54

```
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.
```

1.4.2 Les tableaux en Python

Création d'un tableau vide

```
Il existe deux moyens :
Avec la commande list :
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

On utilisera ici la méthode append :

```
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

On utilise ici la commande del:

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

Longueur d'un tableau

```
On utilise ici la commande len:
```

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

Parcourir un tableau version 1

Parcourir un tableau version 2

1.4.3 Des tableaux et des chaînes

Des tableaux aux chaînes

• On utilise ici la commande join :

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

    Vérifiez que a est un str
```

Des chaînes aux tableaux

• On utilise ici la commande **split**:

```
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.

Propriété

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

Exercice 1.55

Ecrire une fonction ajout(tableau, entier) qui prend un tableau d'entiers et un entier en argument et qui renvoie le tableau où l'entier a été ajouté à la fin.

assert tableau([1,2,3,4],5) == [1,2,3,4,5]

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

Exercice 1.56

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

• Exercice 1.57

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

Exercice 1.58

Ecrire une fonction liste_hasard_n(n) prenant en argument un entier n et qui renvoie un tableau avec n entiers choisis au hasard entre 0 et 100.

™ Ne pas oublier: from random import *

Exercice 1.59

Ecrire une fonction double(tableau) qui prend en argument tableau, de type list et contenant des entiers (type int) et qui renvoie un nouveau tableau où chaque valeur a été doublée :

assert double([1,2,3,4,6]) == [2,4,6,8,12]

Exercice 1.60

Écrire une fonction echange(tableau) qui prend en argument tableau, de type list, et contenant des entiers (au moins 2 éléments, de type int) et qui retourne le tableau dans lequel on a échangé le premier et le dernier élément.

assert echange([1,2,3,4]) == [4,2,3,1]

Exercice 1.61

Écrire une fonction carre(tab) qui prend en argument tab, de type list, un tableau d'entiers (de type int), et qui renvoie un nouveau tableau avec les carrées des entiers du tableau passé en argument.

Exercice 1.62

Écrire une fonction supp(tableau) qui prend en argument tableau, de type list, un tableau d'entiers (avec au moins deux éléments, de type list), et qui renvoie le tableau avec les deux premiers termes supprimés.

Exercice 1.63

Écrire une fonction signe(tab) qui prend en argument tab (type list) un tableau d'entiers (type int) et qui renvoie un tuple (n,p) où n est le nombre d'entiers négatifs et p le nombre d'entiers positifs.

Exercice 1.64

Écrire une fonction grandsmots(tab) qui prend en argument un tableau tab (de type list) contenant des chaines de caractères (de type str), et qui retourne une autre liste contenant les mots de plus de 5 caractères.

Exercice 1.65

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

Exercice 1.66

Programme une fonction rotation(tab) qui décale d'un rang tous les éléments du tableau tab (de type list).

- rela 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 1.67

Programme une fonction inverser(tab) de paramètre tab (de type list), un tableau d'entiers (de type int) et qui renvoie un nouveau tableau avec les éléments dans l'ordre inverse.

Par exemple, assert inverser([1,2,3,4]) == [4,3,2,1].

Exercice 1.68

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) == [8,7,5,4]. (l'élément 6 qui était au rang 2 est supprimé)

Exercice 1.69

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(deb,fin,maxi) 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 :

⚠ 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

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 est la méthode de la copie profonde avec la commande deepcopy :

```
>>> 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]]
```