Les tableaux en Python (types `list` et `tuple`)

Définition : tableau

En informatique, un **tableau** est une structure de données qui contient un nombre fixé d'éléments, tous de même type. Chacun de ces éléments est repéré par son **indice**.
La longueur d'un tableau est le nombre d'éléments qu'il contient.
Un langage de programmation doit permettre, dans un tableau :

- d'**accéder** (efficacement) à un élément d'indice donné ;
- de **modifier** un élément d'indice donné.

Le type `list` de Python

Le langage Python fournit une forme très souple de tableau appelée « liste » (type `list`) :

- Un tableau de type `list` est déclaré à l'aide de crochets, et les éléments sont séparés par des virgules :
  `tab = [] # un tableau vide`
  `tab = [18, 13, 8, 20, 16]` ou `tab = ["pomme", "poire", "kiwi"]`
- Il est possible d'ajouter ou de retirer des éléments à un tableau ;
- On peut accéder à un élément par son indice ;
- La longueur d'un tableau est donnée par la fonction `len` ;
- Un tableau *peut* contenir des éléments de types différents (ce qu'il faut toutefois éviter).

Exercice 1

Quels sont les affichages produits par le script ci-dessous ?

```
notes = [12, 18, 14, 20]
print(notes[1]) # affiche
n = len(notes)
x = notes[n-1]
print(x) # affiche
x = 15
print(notes) # affiche
```

1 Parcours de tableau

Deux façons de parcourir un tableau Python

Afin de parcourir les éléments d'un tableau dans une **boucle bornée**, on peut soit utiliser les **indices**, soit tirer profit du fait qu'un tableau est un **itérable**.

Les syntaxes ci-dessous sont donc équivalentes.

```
for indice in range(len(tab)):
    element = tab[indice]
    # traitement
```

```
for element in tab:
    # traitement
    #
```

On a ainsi deux manières de parcourir un tableau Python.

→ Ces deux façons de parcourir ressemblent beaucoup aux parcours de quel autre type Python ?

Les chaînes de caractères

/!\ Il est interdit d'utiliser la syntaxe `if x in list` à moins que ça ne soit précisé.

Exercice 2

1. Écrire une fonction `moyenne` qui prend en argument un tableau de notes et en renvoie la moyenne.
2. Écrire une fonction `meilleure_note` qui prend en argument un tableau de notes et renvoie la meilleure.

Exercice 3

1. Écrire une fonction `contient(tab, el)` qui prend en argument un tableau de nombres entiers et un nombre `el`, et renvoie `True` si le tableau contient `el` (et `False` sinon).

   /!\ Rappel : il est interdit d'utiliser la syntaxe `if x in list`.
2. Écrire une fonction `position_de(tab, el)` qui prend en argument un tableau de nombres entiers et un nombre `el`, et renvoie l'indice du premier élément de le tableau égal à `el` si le tableau contient `el` (et -1 sinon).

2 Les méthodes

Définition

En informatique, une **méthode** est une fonction propre à un type de variable.
Les tableaux disposent d'un certain nombre de méthodes, parmi lesquelles :

- `append(el)` : ajoute un élément `el` à la fin du tableau ;

Pour appliquer une méthode à un tableau, la syntaxe est :
`nom_du_tableau.nom_de_la_methode(params)`
où `params` désignent les paramètres que la méthode prend en argument.

Exercice 4

Étant donnée la fonction `foo` ci-dessous, quelle est la valeur renvoyée par l'appel `foo([12, 5, 14, 17])` ?

```
def foo(tab):
    tab2 = []
    for el in tab :
        if el % 2 == 0 :
            tab2.append(el)
    return tab2
```

Exercice 5

On propose d'étudier le programme suivant. Quels sont les affichages produits ? Que se passe-t-il de spécial ? Peut-on laisser un tel programme en l'état ?

```
def bar(tab):
    idx = 0
    while idx < len(tab):
        el = tab[idx]
        if el % 2 == 0 :
            tab.append(el)
        idx = idx + 1

tab1 = [11, 5, 13, 17]
bar(tab1)
print(tab1)

tab2 = [12, 5, 14, 17]
bar(tab2)
print(tab2)
```

3 Modifier un élément

Pour modifier la valeur de l'élément d'indice `idx` d'un tableau `tab`, on utilise la syntaxe :
```
tab[idx] = nouvelle_valeur
```

Exercice 6

Quel est l'affichage produit par le script ci-dessous ?

```
tab = [2, 4, 6, 8]
tab[0] = tab[3] * 2
tab[1] = tab[0] - tab[1]
tab[2] += 20
print(tab)
```

4 Modification en place

Définition

Lorsqu'une fonction **modifie un tableau passée en argument (paramètre)**, on dit que la modification s'effectue « *en place* » car le tableau est un tupe *mutable*. C'est-à-dire que le tableau passé en paramètre sera modifiée à l'extérieur de la fonction si elle est modifiée dans la fonction.
Si on ne veut pas modifier le tableau passé en paramètre, il faut en créer une copie et la renvoyer.

Les méthodes comme `append` sont des exemples de modification en place du tableau passé en argument.

Exercice 7

1. Écrire une fonction `bonnes_notes(notes)` qui prend en argument un tableau de notes, et renvoie un **nouveau** tableau constituée des notes supérieures ou égales à 10.
2. Écrire une fonction `supprime_mauvaises_notes(notes)` qui prend en argument un tableau de notes, et la modifie **en place** pour en supprimer les notes inférieures à 10. On autorise à utiliser exceptionnellement la méthode `remove` sur les tableaux (voir documentation Python en ligne).

Exercice 8

Les coordonnées d'un point du plan peuvent être représentées par un tableau de longueur 2.

Dans chacun des cas ci-dessous, on demande de programmer la fonction qui applique la transformation indiquée. Les modifications des coordonnées passées en argument doivent être effectuées **en place**.

1. `symetrie_axe_x(p)` : symétrique du point `p` par rapport à l'axe des abscisses.
2. `symetrie_axe_y(p)` : symétrique du point `p` par rapport à l'axe des ordonnées.
3. `symetrie_origine(p)` : symétrique du point `p` par rapport à l'origine.
4. `symetrie_premiere_bissectrice(p)` : symétrique du point `p` par rapport à la droite d'équation $y = x$.

   $\rightarrow$ Cette transformation consiste simplement à échanger abscisse et ordonnée du point.
5. `translation(p, u)` : image du point `p` par la translation de vecteur `u`.

5 Opérations sur les tableaux

Concaténation

Pour les tableaux, les opérateurs + et * sont surchargés de la manière suivante :

- `tab1 + tab2` est le tableau formé des éléments de `tab1`, suivis des éléments de `tab2` ;
- `tab * n` est le tableau $\underbrace{\texttt{tab} + \cdots + \texttt{tab}}_{\texttt{n fois}}$.

Exercice 9 À la fin de l'exécution du script ci-dessous, quelles sont les valeurs contenues dans le tableau `tab` ?

```
tab = []
for idx in range(5):
    tab = tab + [idx] * idx
```

Exercice 10

1. Que contient le tableau `tab` à l'issue de l'exécution des scripts suivants ?

```
tab = []
for n in range(50000):
    tab.append(n)
```

```
tab = []
for n in range(50000):
    tab = tab + [n]
```

```
tab = []
for n in range(50000):
    tab += [n]
```

2. Toutefois, ces scripts ne sont pas équivalents : comment l'expliquer ?

Exercice 11 Sans utiliser la méthode `insert`, écrire une fonction `inserer(tab, el, idx)` qui :

- prend en argument un tableau `tab`, une valeur `el` et un entier `idx` ;
- renvoie une copie du tableau `tab`, où l'élément `el` a été inséré à l'indice `idx`.

6 Les *tuples* ou *n-uplets*

Définition

En Python, un *tuple* (*n-uplet*) est un tableau qu'il n'est pas possible de modifier.
(C'est un type de variable *immutable*, tout comme les chaînes de caractères, c'est-à-dire qu'elles ne sont pas modifiables *en place*.)
Un tuple est déclaré par des parenthèses, et les éléments sont séparés par des virgules :
`equipe = ("Tiéoulé", "Esmé", "Claire", "Mehdi")`

Les tuples permettent notamment d'effectuer des **affectations simultanées** (voir exercice suivant).

Exercice 12

La fonction ci-dessous prend en argument les coordonnées de deux points du plan.

```
def truc(p1, p2):
    x1, y1 = p1
    x2, y2 = p2
    mx = (x1+x2) / 2
    my = (y1+y2) / 2
    return (mx, my)
```

1. Que renvoie cette fonction lors de l'appel `truc((1, 5), (3, 9))` ?
2. Que représente cette valeur ?

Exercice 13

1. Écrire une fonction `distance(p1, p2)` qui prend en argument les coordonnées de deux points (représentées sous la forme de tuples) et renvoie la distance les séparant.
2. Concevoir une fonction `plus_proche_voisin(tab, p)` qui :
   - prend en argument un tableau de coordonnées de points, ainsi que celles d'un point `p` ;
   - renvoie le point du tableau qui est le plus proche de `p`.

7 Construction de tableaux

Définition mathématiques

En mathématiques, un ensemble peut être défini
- en extension, c'est-à-dire en nommant tous les éléments ;
- en compréhension, à l'aide d'une expression et/ou d'une propriété qui caractérise ses éléments.

Tableaux en extension

On construit le tableau en y ajoutant des éléments grâce à une boucle bornée.

Voici un pseudo-code (en langage naturel) illustrant ce principe :

```
initialiser un tableau tab
pour chaque valeur val entre 0 et 9
    ajouter val dans tab
fin pour
```

Tableaux en compréhension

Le langage Python offre la possibilité de définir un tableau en compréhension selon la syntaxe :
`[ expression for variable in tableau ]` ou bien
`[ expression for variable in tableau if condition ]`

Exemple

Pour construire le tableau des 10 premiers carrés parfaits, on peut utiliser une boucle bornée comme ci-dessous.

```
tab = []
for n in range(10):
    tab.append(n**2)
```

En utilisant la définition en compréhension, le code peut être réduit à la seule ligne suivante.

```
tab = [n**2 for n in range(10)]
```

Exercice 14

Déterminer le contenu du tableau `tab` pour chacune des affectations effectuées dans le script ci-dessous.

```
from math import pi
tab = [2*n+1 for n in range(5)]
tab = [n for n in range(10) if n%3 == 1]
tab = [n for n in range(1, 12) if 12%n == 0]
tab = [(n, n**2) for n in range(5)]
tab = [c for c in str(pi)]
tab = [chr(n) for n in range(97, 123)]
```

Exercice 15

La fonction `randint(a, b)` du module `random` prend en argument deux entiers `a` et `b` et renvoie un entier au hasard compris entre `a` (inclus) et `b` (inclus). On peut donc tirer une note sur 20 au hasard avec les lignes suivantes.

```
from random import randint

note = randint(0, 20)
```

Écrire une fonction `notes_au_hasard(nb)` qui prend en argument un entier `nb` et renvoie un tableau composé de `nb` notes tirées au hasard.

Exercice 16

Écrire une fonction `nombres_premiers(n)` qui prend en argument un entier `n`, et renvoie le tableau des `n` premiers nombres premiers.

Exercice 17

Écrire une fonction qui prend en argument un tableau de nombres et renvoie le tableau des carrés de ces nombres.

Exercice 18

Écrire une fonction `fibonacci(n)` qui renvoie le tableau des `n` premiers termes de la suite de Fibonacci.

L'appel `fibonacci(10)` doit donc renvoyer le tableau `[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]`.

Exercice 19

Le contenu d'un fichier texte est le suivant :

```
Lycée; Adresse; Code Postal; Ville
Lycée Langevin-Wallon; 126 avenue Roger Salengro; 94500; Champigny-sur-Marne
Lycée Louise Michel; 7 rue Pierre Marie Derrien; 94500; Champigny-sur-Marne
```

Dans ce contexte, le point-virgule est utilisé comme séparateur de champs.

Concevoir une fonction `champs(s)` qui prend en argument une chaîne de caractères et renvoie le tableau des champs.

Par exemple, l'appel

```
champs("Lycée Langevin-Wallon; 126 avenue Roger Salengro; 94500; Champigny-sur-Marne")
```

doit renvoyer le tableau

```
['Lycée Langevin-Wallon', '126 avenue Roger Salengro', '94500', 'Champigny-sur-Marne'].
```

Exercice 20

L'écart type σ d'une série statistique (x_i) est la racine carrée de la variance V. La variance est quant à elle définie par la formule $V = \frac{1}{N}\sum (x_i - \overline{x})^2$, où $\overline{x}$ désigne la moyenne et N l'effectif total de la série.

Écrire une fonction « `ecart_type` » qui prend en argument un tableau de nombres et en renvoie l'écart type.

Exercice 21

Concevoir un programme organisé en plusieurs fonctions qui effectue n simulations d'un lancer de dé (penser à `randint`), et affiche (avec `pygame`) l'histogramme des fréquences associées aux éventualités possibles.