Les types `list` et `tuple` de Python

Définition

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 (fonction `len`) 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 (hybride entre tableau, liste et pile), appelée liste (type `list`) :

- Une liste est déclarée à l'aide de crochets, et les éléments sont séparés par des virgules :
  `lst = [] # une liste vide`
  `lst = [18, 13, 8, 20, 16]` ou `lst = ["pomme", "poire", "kiwi"]`
- Il est possible d'ajouter ou de retirer des éléments à une liste ;
- On peut accéder à un élément par son indice ;
- Une liste 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 liste

Deux façons de parcourir une liste Python

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

Les syntaxes ci-dessous sont donc équivalentes.

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

```
for element in lst:
    # traitement
    #
```

On a ainsi deux manières de parcourir une liste Python.

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

Exercice 2

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

Exercice 3

1. Écrire une fonction `contient(lst, el)` qui prend en argument une liste de nombres entiers et un nombre `el`, et renvoie `True` si la liste contient `el` (et `False` sinon).
2. Écrire une fonction `position_de_(lst, el)` qui prend en argument une liste de nombres entiers et un nombre `el`, et renvoie l'indice du premier élément de la liste égal à `el` si la liste contient `el` (et -1 sinon).

2 Les méthodes

Définition

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

- `append(el)` : ajoute un élément `el` à la fin de la liste ;

Pour appliquer une méthode à une liste, la syntaxe est :
`nom_de_la_liste.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(lst):
    lst2 = []
    for el in lst :
        if el%2==0 :
            lst2.append(el)
    return lst2
```

Exercice 5

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

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

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

3 Modification en place

Définition
Lorsqu'une fonction **modifie une liste passée en argument (paramètre)**, on dit que la modification s'effectue **« en place »**. C'est-à-dire que la liste passée en paramètre sera modifiée à l'extérieur de la fonction si elle est modifiée dans la fonction.
Si on ne veut pas modifier la liste passée en paramètre, il faut en créer une copie et la renvoyer. (Voir exercices suivants et annexe sur les types mutables/immutables.)

Les méthodes `append`, `remove` et `insert`, ainsi que la fonction `bar` de l'exercice 5 sont des exemples de fonctions qui modifient en place la liste passée en argument.

Exercice 6

1. Écrire une fonction `bonnes_notes(notes)` qui prend en argument une liste de notes, et renvoie une **nouvelle** liste constituée des notes supérieures ou égales à 10.
2. Écrire une fonction `supprime_mauvaises_notes(notes)` qui prend en argument une liste de notes, et la modifie **en place** pour en supprimer les notes inférieures à 10.

4 Modifier un élément

Pour modifier la valeur de l'élément d'indice `idx` d'une liste `lst`, on utilise la syntaxe :
```
lst[idx] = nouvelle_valeur
```

Exercice 7

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

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

Exercice 8

Les coordonnées d'un point du plan peuvent être représentées par une liste 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 listes

Concaténation

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

- `lst1+lst2` est la liste formée des éléments de `lst1`, suivis des éléments de `lst2` ;
- `lst*n` est la liste $\underbrace{\texttt{lst} + \cdots + \texttt{lst}}_{\texttt{n fois}}$.

Exercice 9

À la fin de l'exécution du script ci-dessous, quelles sont les valeurs contenues dans la liste `lst` ?

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

Exercice 10

1. Que contient la liste `lst` à l'issue de l'exécution des scripts suivants ?

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

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

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

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

Tranchage *(slicing)*

Le langage Python offre une syntaxe pour extraire des éléments contigus d'une liste :

- `lst[i:]` est composée des éléments de `lst` d'indice supérieur ou égal à `i` ;
- `lst[:j]` est composée des éléments de `lst` d'indice inférieur à `j` ;
- `lst[i:j]` est composée des éléments de `lst` d'indice compris entre `i` (inclus) et `j` (exclus) ;

Note : le *slicing* fonctionne également avec les chaînes de caractères.

Exercice 11

La fonction ci-dessous prend en argument une liste. Décrire la valeur renvoyée.

```
def machin(lst):
    n = len(lst)
    return [lst[n-1]] + lst[:n-1]
```

Exercice 12

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

- prend en argument une liste `lst`, une valeur `el` et un entier `idx` ;
- renvoie une copie de la liste `lst`, où l'élément `el` a été inséré à l'indice `idx`.

6 Les *tuples* ou *n-uplets*

Définition
En Python, un *tuple*, appelé aussi *n-uplet* est une liste qu'il n'est pas possible de modifier.
(C'est un type de variable *immutable*, tout comme les chaînes de caractères.)
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 13

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 14

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(liste, p)` qui :
   - prend en argument une liste de coordonnées de points, ainsi que celles d'un point `p` ;
   - renvoie le point de la liste qui est le plus proche de `p`.

7 Construction de listes

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.

Listes en extension
On construit la liste 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
```

Listes en compréhension

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

Exemple

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

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

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

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

Exercice 15

Déterminer le contenu de la liste `lst` pour chacune des affectations effectuées dans le script ci-dessous.

```
lst = [2*n+1 for n in range(5)]
lst = [n for n in range(10) if n%3==1]
lst = [n for n in range(1, 12) if 12%n==0]
lst = [(n, n**2) for n in range(5)]

from math import pi
lst = [c for c in str(pi)]

lst = [chr(n) for n in range(97, 123)]
```

Exercice 16

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 une liste composée de `nb` notes tirées au hasard.

Exercice 17

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

Exercice 18

Écrire une fonction qui prend en argument une liste de nombres et renvoie la liste des carrés de ces nombres.

Exercice 19

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

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

Exercice 20

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 la liste des champs.

Par exemple, l'appel

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

doit renvoyer la liste

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

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 une liste 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.

Exercice 22

Les coefficients binomiaux sont présentés ci-dessous pour former le « triangle de Pascal ».

1						
1	1					
1	2	1				
1	3	3	1			
1	4	6	4	1		
1	$\boxed{5}$	$\boxed{10}$	10	5	1	
1	6	$\boxed{15}$	20	15	6	1

Chaque ligne du triangle de Pascal se déduit de la précédente en additionnant deux termes consécutifs.

On a par exemple $5 + 10 = 15$ pour les nombres encadrés dans le tableau.

Écrire une fonction `pascal(n)` qui prend en argument un entier `n` et renvoie la `n`–ième ligne du triangle de Pascal.