TP : Les listes

Dès que l'on commence à manipuler un certain nombre de données, la notion de variable numérique s'avère insuffisante. Imaginons que l'on souhaite placer en mémoire les notes des élèves d'une classe en vue de déterminer la moyenne, l'écart-type et la médiane de cette série de notes. Une solution naïve consisterait à définir pour chaque élève une variable contenant sa note. Même pour une classe d'une trentaine d'élèves, il serait fastidieux de définir une trentaine de variables. On voit combien il serait avantageux d'attribuer à une seule variable l'ensemble des notes, chaque note étant repérée par un indice.
En **Python**, la structure de données s'appelle une liste.

```
Liste=['stage',12,12.23,17,7,15, « Toto»]
```

Comme on le voit dans l'exemple précédent, une liste est une séquence d'éléments, rangés dans un certain ordre ; de plus, en **Python**, une liste n'est pas nécessairement homogène : elle peut contenir des objets de types différents les uns des autres.
La première manipulation que l'on a besoin d'effectuer sur une liste, c'est d'en extraire un élément : la syntaxe est alors **Liste[indice].**

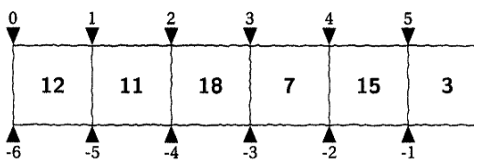
Exemple : On considère la liste suivante :

```
Liste=[12,11,18,7,15,3]
```

- Quelle est la réponse si on tape l'instruction : Liste[2] ? ……..
  Que remarquez-vous ?

En **Python**, les éléments d'une liste sont indexés à **partir de 0 et non de 1** !

- On peut extraire une sous-liste d'une liste. Essayer d'extraire la sous-liste comprenant le troisième, le quatrième et le cinquième élément de la liste précédente. Que remarquez-vous ?
  Encore une fois, le résultat n'est pas à la hauteur de celui que l'on attendait. Il faut comprendre que les indices repèrent les tranchent et non pas les cases :



Remarque : on peut même se servir d'indices négatifs.

ADOPTONS PYTHON

Exercice 1 : Extraire une sous liste d'une liste

Taper les instructions suivantes avec Liste=[12,11,18,7,15,3]et compléter le tableau.

Instruction à saisir	Résultat obtenu + commentaire ou explication
len(Liste)	
Liste[2:]	
Liste[:2]	
Liste[0:len(Liste)]	
Liste[:]	
Liste[2:5]	
Liste[0 :5 : 2]	
Liste[-2:-4]	
Liste[-4:-2]	
Liste[len(Liste)]	
Liste[len(Liste)-1]	
Liste[-1]	

Extraction de sous-listes

Soit **lst** une liste quelconque :

lst[p] renvoie l'élément d'indice **p** de **lst.**

lst[p :n] renvoie un nouvelle liste constituée des éléments de **lst** d'indice **p** inclus à **n** exclu.

lst[p :n :pas] renvoie un nouvelle liste constituée des éléments de **lst** d'indice **p** inclus à **n** exclu, tous les pas.

lst[:] renvoie une nouvelle liste constituée de tous les éléments de **lst.**

lst[p:] renvoie une nouvelle liste constituée des éléments de **lst** à partir de l'élément d'indice **p** inclus**.**

lst[:n] renvoie une nouvelle liste constituée de tous les éléments de **lst** depuis le premier jusqu'à l'élément d'indice **n** exclu.

lst[: : pas] renvoie une nouvelle liste constituée des éléments de **lst**, tous les pas.

Conclusion : On peut extraire une sous-liste d'une liste en indiquant entre crochets les indices des éléments que l'on veut extraire.

NS PYTHON

Exercice 2 : *Manipuler les listes*

Taper les instructions suivantes avec maListe= [22, « coucou »,33, « z »,'a','b',111,99] et compléter le tableau.

Instruction à saisir	Résultat obtenu + commentaire ou explication
maListe	
len(maListe)	
'z' in maListe	
maListe[3]=1024	
maListe	
'z' in maListe	
maListe	
maListe.append(33)	
maListe	
maListe.append('hello')	
maListe	
maListe.remove(33)	
33 in maListe	
maListe	
maListe.pop( )	
maListe.pop(1)	
maListe	
maListe.index('a')	
maListe.index(111)	
maListe.sort( )	
maListe.remove('a')	
maListe.remove('b')	
maListe	

ADOPTONS PYTHON

Exercice 3 :

1. Ecrire une fonction Python qui prend en argument un nombre x et une liste, et qui renvoie un booléen qui dit si x est présent dans la liste ou pas.

2. Ecrire une fonction qui prend en entrée une liste et un élément, et qui renvoie la position de cet élément dans la liste.

3. Ecrire une fonction Python qui renvoie la somme des éléments d'une liste d'entiers. On n'utilisera pas celle prédéfinie en Python.

4. Ecrire une fonction Python qui renvoie le maximum d'une liste d'entiers. On n'utilisera pas celle prédéfinie en Python.

5. Ecrire une fonction qui prend en entrée une liste de nombre et qui fournit en sortie le maximum et ses éventuelles positions.

Exercice 4 :

Soit *impairs* la liste de nombres [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21].
Écrire une fonction qui, à partir de la liste *impairs*, construit une liste pairs dans laquelle tous les éléments de *impairs* sont incrémentés de 1.

Exercice 5 :

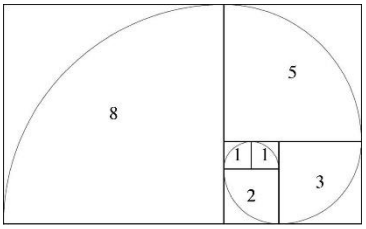
Ecrire une fonction Python qui prend en argument les entiers u, a, b et n et qui renvoie la liste des n premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$ définie par $\begin{cases} u_0 = u \\ u_{n+1} = a * u_n + b \end{cases}$.
Si $n = 0$ alors la fonction doit renvoyer une liste vide, et si $n > 0$ alors elle doit renvoyer la liste $[u_0, u_1, \ldots, u_{n-1}]$.

Exercice 7 : Suite de Fibonacci

La suite de Fibonacci est une suite mathématique qui porte le nom de Leonardo Fibonacci, un mathématicien italien du XIII e siècle. Initialement, cette suite a été utilisée pour décrire la croissance d'une population de lapins mais elle est peut également être utilisée pour décrire certains motifs géométriques retrouvés dans la nature (coquillages, fleurs de tournesol...).

Par définition, les deux premiers termes de la suite de Fibonacci sont 0 et 1.
Ensuite :

le terme au rang n est la somme des termes de rangs n - 1 et n – 2.



1. Déterminer les 10 premiers termes de la suite de Fibonacci.
2. Ecrire une fonction Python qui construit et renvoie la liste des n premiers termes de la suite de Fibonacci.
   Si $n = 0$ alors la fonction doit renvoyer une liste vide, et si $n > 0$ alors elle doit renvoyer la liste $[u_0, u_1, \ldots, u_{n-1}]$.

ADOPTONS PYTHON

Exercice 8 : Conjecture de Syracuse

La conjecture de Syracuse est une conjecture mathématique qui reste improuvée à ce jour et qui est définie de la manière suivante :
Soit un entier positif u_n.

- Si u_n est pair, alors le diviser par 2.
- S'il est impair, alors le multiplier par 3 et lui ajouter 1.

En répétant cette procédure, la suite de nombres atteint la valeur 1, puis se prolonge indéfiniment par une suite de trois valeurs triviales appelée cycle trivial.

Jusqu'à présent, la conjecture de Syracuse, selon laquelle depuis n'importe quel entier positif la suite de Syracuse atteint 1, n'a pas été mise en défaut.

1. Déterminer les six premiers éléments de la suite de Syracuse ayant pour entier de départ 10.
2. Ecrire une fonction Python qui, partant d'un entier positif *n*, crée la liste des n premiers nombres de la suite de Syracuse.
3. Avec différents points de départ, la conjecture de Syracuse est-elle toujours vérifiée ? Quels sont les nombres qui constituent le cycle trivial ?

❖