TP4 - Liste

I Les opérations sur les listes

I.1 Définition

Une liste est un ensemble ordonné d'éléments.
L'ensemble des éléments est encadré par des crochets et les éléments de la liste sont séparés par des virgules.
La longueur d'une liste est le nombre d'éléments de la liste.

Exemple :

```
liste_1 = [2, 5, 7]
liste_2 = [ ] # liste vide
liste_3 = ["Pierre", "Paul"]
print(len(liste_3)) # liste_3 a pour longueur 2
```

I.2 Accéder à un élément de la liste

On accède à un élément de la liste grâce à son indice.
Le 1er élément a pour indice 0, le 2e a pour indice 1.

```
liste = [2, 4, 6]
print(liste[0]) # 2
print(liste[1]) # 4
print(liste[2]) # 6
```

Remarque : Pour accéder au dernier élément d'une liste, on tape `liste[len(liste) - 1]` qui s'écrit plus simplement `liste[-1]` ; pour accéder à l'avant-dernier élément d'une liste, on tape `liste[len(liste) - 2]` qui s'écrit plus simplement `liste[-2]` ; etc.

I.3 Modifier une liste

Une liste est un objet mutable, c'est-à-dire qu'une fois définie, la liste peut être modifiée.

```
liste = [2, 4, 6]

liste[1] = 10 # on modifie l'élément d'indice 1
print(liste) # [2, 10, 6]

liste.append(100) # on rajoute 100 à la fin de la liste
print(liste) # [2, 10, 6, 100]

liste.insert(1, 0) # insère 0 à la position 1
print(liste) # [2, 0, 10, 6, 100]

element = liste.pop(1) # on supprime l'élément de position 1 et on le récupère
print(liste, element) # [2, 10, 6, 100] 0
```

```
14
15 liste.remove(6) # on supprive l'élément 6
16 print(liste) # [2, 10, 100]
```

I.4 Appartenir à une liste

On peut tester si un élément appartient à une liste.

```
1 liste = [2, 4, 6]
2 print( 6 in liste) # True
3 print(5 not in liste) # True
```

I.5 Concaténation de deux listes

Lorsqu'on concatène deux listes, on crée une nouvelle liste.

```
1 liste1 = [3, 4]
2 liste2 = [10, 20]
3
4 liste3 = liste1 + liste2
5 print(liste3) # [3, 4, 10, 20]
6
7 liste4 = 3 * liste1
8 print(liste4) # [3, 4, 3, 4, 3, 4]
```

Q1. Sans ordinateur, indiquer ce qu'il s'affiche.

```
1 liste1 = [5, 2, 3, 0]
2 liste2 = [2, 4, 1]
3 liste1.append(liste2[-2])
4 print(liste1)
5 liste2.remove(4)
6 print(liste2)
7 liste2 = 2 * liste2
8 print(liste2)
9 print(liste2[3])
```

I.6 Exercices : création de listes

Q2. On souhaite créer la liste des entiers multiples de 14 et compris entre 100 et 200.
On peut d'abord créer une liste vide, puis à l'aide d'une boucle rajouter les éléments.
Compléter le code suivant :

```
1 liste = [ ]
2 for n in range(100, 201):
3   if
4
5
6
7 print(liste)
```

Sortie
[112, 126, 140, 154, 168, 182, 196]

On aurait pu aussi définir la liste par compréhension :

```
liste = [n for n in range(100, 201) if n % 14 == 0]
```

Q3. Suite de Fibonacci
La suite de Fibonacci est définie par $u_0 = u_1 = 1$ et pour tout $n \in \mathbb{N}$ $u_{n+2} = u_{n+1} + u_n$.
Ecrire une fonction `fibonacci(n : int) -> List[int]` qui renvoie la liste des n premiers termes de la suite.
Afficher la liste des 10 premiers termes.

Sortie
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

Q4. La suite de Syracuse
La suite de Syracuse est définie de la façon suivante :

- le premier terme est un entier strictement positif u_0 ;
- pour tout $n \in \mathbb{N}$ $u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$

1. Ecrire une fonction `syracuse_20(u_0 : int ) -> List[int]` qui renvoie la liste des 20 premiers termes de la suite de Syracuse de premier terme u_0.
2. Afficher la liste des 20 premiers termes pour $u_0 = 1$, $u_0 = 2$, ... $u_0 = 10$.

Sortie
[1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4]
[2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1]
[3, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1]
[4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2]
[5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2]
[6, 3, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2]
[7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1]
[8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4]
[9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
[10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4]

On remarque que la suite finit toujours par arriver à 1

3. Vérifier (mathématiquement) que si l'un des termes de la suite est égal à 1, alors la suite ne prend plus que trois valeurs.
4. Ecrire une fonction `syracuse(u_0 : int) -> List[int]` qui renvoie la liste des termes de la suite jusqu'au terme égal à 1.

II Parcourir une liste

On peut parcourir une liste de deux façons :

- en utilisant les indices

```
for k in range(len(liste)):
    print(liste[k])
```

- en accédant directement aux éléments

```
for element in liste :
    print(element)
```

Q5. Compléter le tableau :

```
liste = [2, 4, 5]
for k in range(len(liste)):
  print(k * liste[k])
print("fin")
```

ligne	1	2	3	2				
k		0		1				
Affichage			0					

Q6. Compléter le tableau :

```
liste = [2, 4, 5]
for element in liste:
  print(element * 5)
print("fin")
```

ligne	1	2	3	2				
element		2		4				
Affichage			10					

Q7. Sans ordinateur, indiquer ce qu'il s'affiche.

```
liste1 = ['a', 'e', 'j']
for i in range(len(liste1)):
  print(i, liste1[i])
```

```
liste2 = [2, 4, 1, 5]
for elt in liste2:
  print(elt * 3)
```

Q8. La somme des termes d'une liste
Ecrire une fonction `somme(liste : List[int]) -> int` qui renvoie la somme des termes d'une liste.

Q9. Ecrire une fonction `occurrence(liste : List[int], x : int) -> int` qui renvoie le nombre d'éléments de `liste` égaux à `x`.

Q10. Ecrire une fonction `maxi(liste : List[int]) -> int` qui renvoie l'indice du maximum des éléments d'une liste.

Q11. Ecrire une fonction `positif(liste : List[int]) -> List[int]` qui renvoie la liste des entiers positifs contenus dans `liste`. On créera une nouvelle liste.

Q12. Ecrire une fonction `est_triee(liste : List[int]) -> bool` qui renvoie `True` si `liste` est triée et `False` sinon.

Q13. Ecrire une fonction `conversion_binaire_vers_decimal(liste : List[int])-> int` qui prend en paramètre une liste $[a_n, \cdots, a_0]$ et qui renvoie le nombre décimal dont $(a_n \cdots a_0)$ est l'écriture binaire.

III Listes de listes

`liste = [[10, 20, 30, 40], [5, 15, 25]]` est une liste qui a deux éléments.
Le 1er élément est la liste `[10, 20, 30, 40]` et le 2e élément est la liste `[5, 15, 25]`.

```
liste = [[10, 20, 30, 40],
         [5, 15, 25]]
print(len(liste)) # 2
print(liste[0]) # le 1er élément : [10, 20, 30, 40]
print(liste[1]) # le 2e élément : [5, 15, 25]
print(len(liste[0]) # 4
print(liste[1][2]) # 25
```

Q14. Sans ordinateur, indiquer ce qu'il s'affiche :

```
liste = [[10, 20, 30, 40],
         [5, 15, 25]]
for element in liste :
  print(element)
  print(element[1])
```

```
liste = [[10, 20, 30, 40],
         [5, 15, 25]]
for k in range(len(liste)) :
  print(liste[k])
  print(liste[k][k])
```

Q15. Dans un tableau, on a stocké les notes d'un élève ainsi que les coefficients associés aux notes.

Exemple : `[(15, 1), (10, 3), (12, 2)]`

On considère le type `NotePond = Tuple[int, int]` dont le 1er élément du tuple est la note et le 2e élément le coefficient.

Exemple : `(15, 1)` est de type `NotePond`.

Ecrire une fonction `moy_pond(tab : List[NotePond]) -> float` qui a pour paramètre un tableau de notes `tab` et qui renvoie la moyenne pondérée des notes du tableau.

Q16. On considère un tableau de notes contenant les notes de colles d'une classe.

12	15	17	13
13	10	14	
8	9	12	8
...			

On désigne par `Notes` le type `List[int]`.
Ce tableau de notes peut être représenté par une liste de `Notes` :
`[[12,15,17,13], [13,10,14], [8,9,12,8], ... ]`

1. Ecrire une fonction `get_notes_eleve(tab : List[Notes], id : int) -> Notes` qui a pour paramètre un tableau de notes `tab` et un identifiant d'élèves `id` et qui renvoie la liste des notes obtenues par l'élève `id`.

2. Ecrire une fonction `moyenne_eleve(tab : List[Notes], id : int) -> float` qui a pour paramètre un tableau de notes `tab` et un identifiant d'élèves `id` et qui renvoie la moyenne des notes de l'élève `id`.

3. Ecrire une fonction `moyenne_note1(tab : List[Notes]) -> float` qui a pour paramètre un tableau de notes `tab` et qui renvoie la moyenne de la 1ère note des élèves de la classe.

4. Ecrire une fonction `eleves_inf_10(tab : List[Notes]) -> List[int]` qui a pour paramètre un tableau de notes `tab` et qui renvoie la liste des identifiants des élèves ayant eu au moins une note inférieure à 10.