I P-UPLETS, P-UPLETS nommés

Nous avons présenté dans un premier temps, les types de base (nombre entier ou flottant, caractères, booléens).Il est possible de travailler avec des éléments plus élaborés et notamment ce que l'on appelle des **séquences** .

Une séquence est un ensemble fini et ordonnée d'éléments. Nous allons aborder deux types de séquences : **Les listes et les tuples** .

Nous allons travailler avec la syntaxe du langage python.

1) Les listes

- Création d'une liste :

L=[]	Création d'une liste vide
L=[0]*10	Création de la liste [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
L=[i for i in range(10)]	Création de la liste [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

  La troisième méthode de création est appelée création par compréhension, on y reviendra.
- Une liste est une séquence : Elle est donc ordonnée. **Son premier indice a pour rang 0** .
  On peut ainsi accéder à un élément d'une liste par son rang.

l=[1,5,8,4,10,54], l[2]=8
On peut aussi accéder à des parties de la liste :
l[:2] renvoie [1, 5]
l[2 :] renvoie[8,4,10,54]
l[2 :4] renvoie [8,4]

- **Les listes sont mutables** : On peut remplacer un élément.
  l[2]= -87 (on affecte -87 au terme de rang 2 de la suite).

Nouvel état de la liste :

- On peut ajouter un élément à une liste avec la méthode **append.**

Cet élément vient se positionner au dernier rang de la liste .

l.append(17)

Nouvel état de la liste :

- On peut supprimer un élément d'une liste avec la méthode del

  del(l[2]) .

Nouvel état de la liste :

Si l'on veut enlever une certaine valeur de la liste , par exemple, la valeur 4, on peut aussi utiliser la méthode remove .

l.remove(4) enlève tous les 4 de la liste. Bon, ici il n'y en a qu'un .

Nouvel état de la liste :

- On peut déterminer la longueur d'une liste avec la méthode len
  len(l) renvoie le nombre d'éléments de la liste.

2) Parcours d'une liste

On peut parcourir une liste à l'aide d'une boucle for .

```
In [23]: l=[1,2,3,4,5]

In [24]: for valeur in l:
    ...:     print(valeur)
    ...:
1
2
3
4
5
```

On peut également effectuer des changements sur les valeurs de la suite

```
In [25]: for valeur in l:
    ...:     valeur= valeur**2
    ...:     print(valeur)
    ...:
1
4
9
16
25
```

On peut également tester l'appartenance d'un élément à une liste :

```
In [15]: def ok(v):
    ...:     if v in l :
    ...:         return "OK"
    ...:     else :
    ...:         return "N'appartient pas"
    ...:

In [16]: ok(7)
Out[16]: "N'appartient pas"

In [17]: ok(1)
Out[17]: 'OK'
```

3) Copie d'une liste

On peut être amené à vouloir modifier une liste mais aussi à conserver l'originale .

Instinctivement, on va effectuer la démarche suivante avec la liste l=[1,2,3,4,5]

```
In [26]: m=l  # on affecte à la liste m le contenu de la liste l

In [27]: print(m)
[1, 2, 3, 4, 5]

In [28]: l[2]=8

In [29]: print(m)
[1, 2, 8, 4, 5]
```

En modifiant la liste l , on modifie aussi la liste m. Ce qui n'est pas le but.

```
In [30]: m=list(l)

In [31]: l[2]='arbre'

In [32]: print(l)
[1, 2, 'arbre', 4, 5]

In [33]: print(m)
[1, 2, 8, 4, 5]
```

Ici on utilise la fonction list, créatrice de liste avec la liste l en paramètre. Les deux objets sont désormais distincts.

4) Création de liste par compréhension

La compréhension de liste est une expression qui permet de construire une liste à partir de tout autre type itérable (liste, chaîne de caractères…). Le résultat obtenu est toujours une liste.

Exemples :

```
In [34]: prenom='Benedict'

In [35]: l=[lettre for lettre in prenom]

In [36]: print(l)
['B', 'e', 'n', 'e', 'd', 'i', 'c', 't']
```

```
In [37]: l=[p**2 for p in range(1,5)]

In [38]: print(l)
[1, 4, 9, 16]
```

On peut utiliser des instructions conditionnelles dans la création de liste par compréhension :

Exemples : 1/

```
In [39]: l=[p**2 for p in range(0,10) if p%2==1]

In [40]: print(l)
[1, 9, 25, 49, 81]
```

2/

```
In [32]: l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
    ...: l1 = [n*2 if n % 2 == 0 else n*3 for n in l]
    ...: print(l1)
    ...:
[0, 3, 4, 9, 8, 15, 12, 21, 16, 27, 20, 33, 24, 39, 28, 45]
```

5) Exercice sur les listes

1/ Soit l= [1,2,5,7,9]. Créer, à partir de l une liste qui contient les carrées des éléments de l.

2/ Soit l=[A,b,C,d]. Ecrire un programme qui remplace chaque lettre de la liste par son code ASCII.

3/ Créer une liste qui contient les 100 premiers nombres entiers non multiples de 2,3 ou 5.

4/ Créer une liste par compréhension qui contient les cubes des entiers non multiples de 5 plus petits que 101.

6) Les tuples

Si les tuples ont des points communs avec les suites , ce sont des séquences (donc un ensemble fini d'éléments ordonnés), ils différent dans la mesure où sont des objets non mutables .

La syntaxe n'est pas non plus tout à fait la même :

```
In [2]: t=(1,2,3,4,5)

In [3]: type(t)
Out[3]: tuple

In [4]: len(t)
Out[4]: 5

In [5]: t[1]
Out[5]: 2

In [6]: t[1]=6
Traceback (most recent call last):

  File "<ipython-input-6-0350a950cbdb>", line 1, in <module>
    t[1]=6

TypeError: 'tuple' object does not support item assignment
```

Les propriétés des tuples sont assez identiques à celles des listes. La principale différence vient du fait qu'un tuple n'est pas modifiable.

Cependant , on peut réaliser l'opération suivante

```
In [7]: t=(7,)+t[1:]

In [8]: print(t)
(7, 2, 3, 4, 5)
```

On remarquera que pour définir un tuple, il faut au moins une virgule .

```
In [11]: t=(7)+t[1:]
Traceback (most recent call last):

  File "<ipython-input-11-54b7a7e72838>", line 1, in <module>
    t=(7)+t[1:]

TypeError: 'int' object is not subscriptable
```

II TABLEAUX A DEUX DIMENSIONS

Nous avons déjà travaillé avec les tableaux à une dimension, nous allons maintenant nous pencher sur les tableaux à deux dimensions.

Nous allons définir ces tableaux à l'aide de liste de listes.

1) Création de tableaux

Supposons que nous voulions créer un tableau de 10 lignes et de 10 colonnes contenant des 0.

Nous pouvons utiliser la syntaxe mat=[[0,0,0,0,0,0,0,0,0,0],[0…..]……] mais qui est pénible.

Il existe une méthode plus simple :

```
In [18]: l=[[0]*10]*10

In [19]: print(l)
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0,
0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0,
0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

[0]*10 : On crée une liste de dix 0. […]*10 : On le fait dix fois et le résultat est stocké dans une liste.

Exercices

On va créer un tableau dont la première ligne contient les nombres entiers de 0 à 9, la deuxième ligne ceux de 10 à 19, etc …, et la dernière (10 ème ligne) les entiers de 90 à 99.

1) Créer ce tableau à l'aide de boucles for

2) Créer ce tableau par compréhension.

2) Recherche d'un élément dans un tableau

Soit T un tableau à deux dimensions :

```
T = [[0, 1, 2, 3, 4],
     [5, 6, 7, 8, 9],
     [10, 11, 12, 13]]
```

Il s'agit de trois listes d'entiers. L'élément T[i][j] correspond au $j^{ème}$ élément de la $i^{ème}$ liste. Attention, on compte à partir de 0.

Ainsi T[2][3] = 13

3) Tri d'un tableau

Pour trier une liste, on peut utiliser la méthode sort.

```
[15]: l=[4,5,7,1,8,10]

[16]: l.sort()

[17]: l
[17]: [1, 4, 5, 7, 8, 10]
```

On remarquera que la méthode sort modifie la liste.

Si l'on veut la liste en ordre décroissant, il suffit d'indiquer *reverse = True* dans les parenthèses de la méthode sort.

Pour classer dans un tableau à deux dimensions, il faut choisir un critère de classement .

Exemple

```
T = [['Anna', 17],
     ['Benjamin', 12],
     ['Sarah', 15],
     ['Kevin', 9]]
```

Ici, l'on peut classer par ordre alphabétique ou par notes croissantes ou décroissantes.

Il va falloir préciser si l'on trie suivant l'élément 0 de la liste ou l'élément 1.

La méthode sorted est ici adaptée, avec la syntaxe suivante

```
T = [['Anna', 17],
     ['Benjamin', 12],
     ['Sarah', 15],
     ['Kevin', 9]]
tri = sorted(T, key=lambda note: note[1])
```

```
Out[19]: [['Kevin', 9], ['Benjamin', 12], ['Sarah', 15], ['Anna', 17]]
```

Attention, la méthode *sorted* créé une autre liste. T n'a pas été modifiée. C'est tri qui a été créée.

Exercices

Ex1) Afficher le tableau T classé par ordre alphabétique inversé.

Ex2)

1) Créer une liste de cinq listes contenant chacune trois éléments :
   un mot français , sa traduction dans deux autres langues.
2) Afficher le tableau avec un classement alphabétique en français , puis dans une autre langue.
3) Afficher le tableau tel que le premier élément de chaque liste soit un mot étranger.