Les types de données composés.

I Les n-uplets

Un n-uplet, tuple en anglais est une généralisation du concept de couple ou de triplet. Comme en mathématiques, pour construire une expression n-uplet, il suffit de placer des expressions entre parenthèses séparées par des virgules. Ces expressions peuvent être de types différents.

Il est possible de stocker un n-uplet dans une variable :

```
>>> x=(2,3.2,5)
>>>
```

```
>>> y=(3,'cat',2==3,2.3)
>>> type(y)
<class 'tuple'>
```

Pour accéder aux composantes d'un n-uplet, c'est-à-dire aux sous-valeurs qu'il contient, on utilise l'expression $t[i]$ où i est le numéro de la composante, on parle de son *indice*.

!!! En Python :

- On commence à numéroter à partir de 0 et non de 1.

```
>>> x=(2,3.2,5)
>>> x[0]
2
>>> x[2]
5
```

- les n-uplets sont immuables, il n'est pas possible d'affecter de nouvelles valeurs aux composantes. C'est-à-dire qu'elles ne peuvent plus être modifiées une fois créées.

```
>>> x[2]=3
Traceback (most recent call last):
  File "<interactive input>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

Concaténation : il est possible de coller un n-uplet et un p-uplet pour obtenir un (n+p) u-plet. On parle de concaténation :

```
>>> y=(3,'cat',2==3,2.3)
>>> x=(2,3.2,5)
>>> x+y
(2, 3.2, 5, 3, 'cat', False, 2.3)
```

Si x et y sont des n-uplets : $x+y \neq y+x$

```
>>> y+x
(3, 'cat', False, 2.3, 2, 3.2, 5)
```

Test d'appartenance : il possible de tester si une valeur appartient à un n-uplet grâce à l'opérateur **in** :

```
>>> t=(5,'toto',3,2==3)
>>> 5 in t
True
>>> 1 in t
False
>>> True in t
False
```

Longueur d'un n-uplet : on l'obtient grâce à la fonction **len** :

```
>>> len(t)
4
>>> len((5,'gyugv','g',5,3))
5
```

II Listes

Une liste est un n-uplet dont on peut changer la valeur des composantes.

Pour construire une liste, on remplace les parenthèses par des

```
>>> l=[1,2,3,'t',5.2]
>>> l[0]
1
>>> l+['chat',6]
[1, 2, 3, 't', 5.2, 'chat', 6]
```

On définit ainsi la **liste vid**

```
>>> l=[]
>>> len(l)
0
```

Pour une liste, contrairement au n-uplet, aucune erreur n'apparaît lorsque l'on change la valeur d'un élément :

```
>>> l=['chien','chat',8]
>>> l[1]='poisson rouge'
>>> l
['chien', 'poisson rouge', 8]
```

Il est possible d'ajouter un élément à la fin d'une lis

```
>>> l=['chien','chat',8]
>>> l.append('canard')
>>> l
['chien', 'chat', 8, 'canard']
```

Il est possible de retrouver l'indice d'un élément :

```
>>> l.index('chat')
1
```

Il est possible d'enlever un élément de la liste :

```
>>> l.remove(8)
>>> l
['chien', 'chat', 'canard']
```

Remarque : pour afficher les termes d'une li

```
>>> l=[2,3,5,6]
>>> for i in range(1,3):
...     print(l[i])
...
3
5
```

Exercices :

1°) Ecrire un script stockant dans une liste contenant tous les multiples de 7 jusqu'à 200 puis qui les affiche.

2°) Ecrire un script stockant dans une liste les nombres impairs, non multiples de 3 et 5 compris entre 1 et N (saisi par l'utilisateur).

3°) Soit la suite définie par récurrence par : $\begin{cases} u_0 = 1 \\ u_{n+1} = \sqrt{3u_n + 1} \end{cases}$

Stocker dans une liste les coordonnées des 20 premiers points du nuage de points $U_n(n;u_n)$.