Les tuples

→ Un tuple en python est une collection d'objets ordonnés.

→ En python les tuple s'écrivent avec des parenthèses ()

```
>>> T=()
>>> type(T)
<class 'tuple'>
```

```
>>> T=(1,2,3)
>>> type(T)
<class 'tuple'>
```

⚠ un tuple avec un seul élément a s'écrit `(a,)`

→ Le nombre d'éléments dans un tuple `T` est `len(T)`

→ Les éléments d'un tuple sont numérotés de 0 à `len(T)-1`

→ Pour accéder au *i*ème élément d'un tuple `T`, on tape `T[i]`

→ Les tuples ne peuvent pas être modifiés !

Les tuples (suite)

→ affectation multiple à l'aide des tuples

```
>>> (a,b)=(3,'chaine')
>>> print(a)
3
>>> print(b)
chaine
```

Les listes

→ Une liste en python est une collection d'objets ordonnés.

→ En python les listes s'écrivent avec des crochets [ ]

```
>>> L=[]
>>> type(L)
<class 'list'>
```

```
>>> L=[1,2,3]
>>> type(L)
<class 'list'>
```

→ Le nombre d'éléments dans une liste `L` est `len(L)`

→ Les éléments d'une liste sont numérotés de 0 à `len(L)-1`

→ Pour accéder au *i*ème élément d'une liste `L`, on tape `L[i]`

→ Les listes **peuvent être modifiées** :

Pour ajouter un élément à la fin d'une liste : `L.append('element')`

Construire une liste

→ On peut contruire des listes par accumulation :

- On crée une liste vide que l'on stocke dans une variable
- On remplit `L` en ajoutant les éléments un par un

```
>>> L=[]
>>> L.append(1)
>>> L.append(2)
>>> print(L)
[1, 2]
```

→ On peut construire des listes en combinant la boucle `while` ou la boucle `for` avec la construction par accumulation !

`for` avec des `range`

→ Syntaxe pseudo-code et `python`

Tant que

Pseudo-code

pour *i allant de* 1 *à* 5 **faire**
 | affiche(i)
fin

`python`

```
for i in range(1,6):
    print(i)
```

→ **L'indentation est indispensable en `python`**

Indentation = 4 espaces

`for` avec des tuples

Avec `python`, on peut itérer sur les éléments d'un tuple

```
>>> t=(10,20,30,40,50)
>>> for x in t:
...     print(x)
...
10
20
30
40
50
```

`for` avec des listes

Avec `python`, on peut itérer sur les éléments d'une liste

```
>>> L=['Alice','Bob','Eve']
>>> for x in L:
...     print('Prénom :',x)
...
Prénom : Alice
Prénom : Bob
Prénom : Eve
```

`for` avec des chaînes de caractères

Avec `python`, on peut itérer sur des chaînes de caractères

```
>>> ch='Alice'
>>> for x in ch:
...     print(x)
...
A
l
i
c
e
```

Dialogue utilisateur

→ Pour lire des valeurs entrées au clavier, on peut utiliser `input`

Dans l'exemple ci-dessous, l'utilisateur tape Alice au clavier

```
>>> nom=input()
Alice
>>> print(nom)
Alice
```

→ Une version de `input` avec un message :

Dans l'exemple ci-dessous, l'utilisateur tape 6B au clavier

```
>>> groupe=input('Entrez votre groupe : ')
Entrez votre groupe : 6B
>>> print(groupe)
6B
```