Les messages d'erreur

1. Les erreurs de crochets

Pour accéder à un élément d'une liste, nous utilisons une paire de crochets contenant un indice entier. Cette syntaxe est source de multiples erreurs :

```
T = [3,4,5]
T(4)= 5
```

Comme vous l'avez peut-être remarqué, les crochets ont été malencontreusement remplacés par une paire de parenthèses. L'erreur associée est peu compréhensible :

```
T(4) = 5
SyntaxError: can't assign to function call
```

En fait, dès lors que l'interpréteur voit une paire de parenthèses suivant un identificateur, ici `T`, il comprend que l'on cherche à appeler une fonction nommée `T` (voir le chapitre Les fonctions). Un appel de fonction (*function call*) ne peut pas figurer à gauche d'un symbole = d'affectation, car on attend une variable. Le terme "affectation" se traduisant par "assignment" en anglais, on obtient le message d'erreur : "can't assign to function call".

2. Les erreurs d'indice

Pour accéder à l'élément d'une liste, il faut utiliser une valeur entière. Voyons ce qu'il se passe dans le cas contraire :

```
L = [4,5,6]
k = 1.3
print(L[k])
```

Le message est clair et vous explique le problème :

```
print(L[k])
TypeError: list indices must be integers or slices, not float
```

En anglais, le terme "slice" désigne une plage de valeurs. Ce message signifie donc que l'indice d'une liste doit être un nombre entier ou une plage d'indices, comme `[0:4]`.

3. On mélange les listes et les nombres

Parfois, nous avons beaucoup de variables : des entiers, des listes… Et manque de chance, on utilise une variable pour désigner une liste alors qu'elle correspond à un nombre :

```
a = 1
```

```
a[1] = 5
```

Le message d'erreur obtenu est le suivant :

```
a[1] = 5
TypeError: 'int' object does not support item assignment
```

La variable correspond à un nombre entier (type int). À la ligne suivante, nous accolons une paire de crochets à un indice [1] comme s'il s'agissait d'une liste, ce qui n'est pas le cas. Le message d'erreur se décode de la manière suivante : une variable de type entier suivie d'un indice ne peut servir pour une affectation.

Les messages d'erreur dépendent du contexte. Ainsi, on peut obtenir des messages différents pour le même problème. Dans l'exemple précédent, l'interpréteur Python pensait traiter une affectation. Voici un autre exemple :

```
a = 4
print(a[3])
```

Le message d'erreur est le suivant :

```
print(a[3])
TypeError: 'int' object is not subscriptable
```

Le message est correct, mais peu compréhensible. En anglais, le mot "subscript" désigne un indice. Ainsi, "int is not subscriptable" signifie : un entier n'est pas apte à recevoir un indice. Ce qui est vrai !