Cours_tuple

April 1, 2021

1 Les tuples

On a vu que les listes étaient mutables, c'est-à-dire modifiables. Cela est très pratique et optimise la gestionde la mémoire mais elles sont potentiellement soumises à une modification par effet de bord. Pour cette raison, il a été créé un nouveau type `tuple` qui est une **liste immuable** donc non modifiable et donc non soumise aux effets de bords. Les **tuple** sont donc utilisés dans le cas où on veut se protéger des effets de bords.

Les tuple sont comme des listes !!... A 2 exceptions près :

- **Les éléments d'un tuple sont entourés de parenthèses `()` plutôt que de crochets `[]`**
- **Les tuples ne sont pas modifiables (ils ont été créés pour cela !)**

[8]:
```
# data est un tuple
data = (1, 2, 3, 5)
print(data)
type(data)
```

```
(1, 2, 3, 5)
```

[8]:
```
tuple
```

[9]:
```
# Donc data n'est pas modifiable

data[2] = 12

print(data)
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-9-96de42eca60b> in <module>
```

```
      1 # Donc data n'est pas modifiable
      2
----> 3 data[2] = 12


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

[10]:
```
# data est une liste
data = [1, 2, 3, 5]
print(data)
type(data)
```

```
[1, 2, 3, 5]
```

[10]:
```
list
```

[ ]:
```
# Donc data est modifiable

data[2] = 12

print(data)
```

2 Remarque sur la syntaxe

- On peut oublier les parenthèses pour un tuple (Attention, cela peut prêter à confusion !)

[11]:
```
# data est un tuple
data = 1, 2, 3, 5
print(data)
type(data)
```

```
(1, 2, 3, 5)
```

[11]:
```
tuple
```

- N'oubliez pas : le `.` est utilisés pour les `float` et la `,` pour les `tuple` !!

[12]:
```
x = 1.2
type(x)
```

[12]:
```
float
```

[13]:
```
x = 1,2
type(x)
```

[13]:
```
tuple
```

- Attention à la syntaxe des tuples possédant un seul élément ou des tuples vides, elles peuvent paraître surprenantes

[19]:
```
# tuple possédant un seul élément
a = 4,
type(a)
```

[19]:
```
tuple
```

[10]:
```
# tuple possédant un seul élément
a = (4,)
type(a)
```

[10]:
```
tuple
```

[12]:
```
# tuple vide
a = ()
type(a)
```

[12]:
```
tuple
```

[13]:
```
# syntaxe incorrecte
a = ,
type(a)
```

```
  File "<ipython-input-13-82a9e4dba4f1>", line 1
    a = ,
        ^
SyntaxError: invalid syntax
```

[14]:
```
# syntaxe incorrecte
a = (,)
type(a)
```

```
  File "<ipython-input-14-857f5f676f8c>", line 1
    a = (,)
         ^
SyntaxError: invalid syntax
```

2.1 Exemples d'utilisation d'un tuple

2.1.1 Fonction renvoyant "plusieurs résultats"

En fait, une fonction renvoie toujours UN SEUL résultat mais celui-ci étant un tuple, il peut contenir plusieurs résultats

[14]:
```
# définition de la fonction
def division_euclidienne(a,b):
    partie_entiere = a // b
    reste = a % b
    return partie_entiere, reste # la fonction renvoie un tuple contenant 2
 ↪éléments

# appel de la fonction
division_euclidienne(33,6)
```

[14]:
```
(5, 3)
```

2.1.2 Permutation de 2 variables

Cet exemple est à connaître. Il est très utilisé !!

[42]:
```
a = 1
b = 3

a, b = b, a # permutation de a et b

print("a =",a)
print("b =",b)
```

```
a = 3
b = 1
```