Chap. 2 Types structurés Thème 2

p-uplets (ou tuples)

Sommaire du chapitre 2

Un **p-uplet** (ou *tuple* en anglais) est une collection ordonnée d'éléments, appelés *composantes* ou *termes*. Il correspond à un p-uplet en mathématiques. Un tuple avec deux éléments est une *paire*, avec trois éléments un *triplet*, etc.

I Création d'un p-uplet

En langage Python, les p-uplets sont notés entre **parenthèses** en listant les éléments séparés par des virgules. Chaque élément peut être de n'importe quel type. Les parenthèses ne sont pas obligatoires mais sont fortement conseillées pour la lisibilité du code.

Exemples

— Si `x` et `y` sont deux nombres, le tuple `(x, y)` peut représenter un point
— le tuple `(14, 'juillet', 2020)` peut représenter une date
— le tuple `(7, 'pique')` peut représenter une carte à jouer
— si `p` et `q` sont des entiers `(p, q)` peut représenter le nombre rationnel `p/q`

Remarques

- Les instructions `(14, 'juillet', 2020)` ou encore `14, 'juillet', 2020` créent le même triplet.
- Un tuple vide est noté `( )`. Un p-uplet ne contenant qu'un seul élément s'écrit avec une virgule après l'élément pour le distinguer d'une simple expression mathématique entre parenthèses :
  `(10, )` et `(10)` sont deux types différents.

Un tuple peut être affecté à une variable :

```
position = (100, 200)
date = (14, 'juillet', 2020)
carte = (7, 'pique')
fraction = (10, 7)
```

II Indexation des éléments du p-uplet

Pour un p-uplet de `n` éléments, noté `t`, les termes sont **indexés** de `0` à `n-1`.
L'accès aux éléments du p-uplet se fait par la **notation indexée** `t[i]`, où `i` est appelé l'**indice** de l'élément `t[i]`. Le premier élément du tuple est à l'indice 0, le nombre d'éléments du tuple est donné par `len(t)`.

Exemples

`date[1]` désigne le deuxième élément du triplet `date` qui vaut `'juillet'`.
`date[-1]` ou `date[len(date)-1]` ou `date[2]` désigne le dernier élément du triplet `date` qui vaut `2020`, et qui se trouve au rang `n-1`.

Remarques

Si l'on tente d'accéder à un élément avec un indice en dehors du tuple, on obtient une erreur :

```
>>> carte[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: tuple index out of range
```

III Modification d'un p-uplet

Il n'est pas possible de modifier par affectation les termes d'un p-uplet après sa création. Un p-uplet est dit **non muable** ou **immuable** (« *non mutable* » ou « *immutable* » en anglais).

En particulier, si on construit un tuple avec des variables, ce sont les *valeurs* de ces variables qui sont utilisées, et le tuple ne sera pas modifié si on change les valeurs de ces variables :

```
a = 10
b = 20
t = (a, b)  # t vaut (10, 20)
a = 50      # t vaut toujours (10, 20)
```

Remarques

Si les valeurs des termes d'un tuple doivent être changées au fil du programme, alors il faudra choisir un autre type de variable, comme le tableau.

IV Fonction renvoyant un p-uplet de valeurs

Une fonction peut retourner un tuple.

```
def position_tortue() :
    """ Retourne un tuple contenant la position (x, y) de la tortue """
    return (xcor(), ycor())
```

Afin de pouvoir réutiliser les coordonnées de la tortue , il est possible de les affecter à une unique variable `coordonnees` avec l'instruction Python suivante : `coordonnees = position_tortue()`.

La variable `coordonnees` sera alors de type tuple et les coordonnées de la tortue Python seront accessibles avec les instructions `coordonnees[0]` pour l'abscisse de la tortue et `coordonnees[1]` pour son ordonnée.

Remarques

Une fonction peut prendre un tuple en paramètre :

```
def norme(p) :
    """ Calcule la norme du point p """
    x, y = p
    return math.sqrt(x*x + y*y)
```

V Affectation multiple

- Pour extraire les éléments d'un tuple, on utilise l'**affectation multiple**.

```
x, y = position              # x vaut 100 et y vaut 200
jour, mois, annee = date     # jour = 14, mois = 'juillet', annee = 2020
valeur, couleur = carte      # valeur = 7, couleur = 'pique'
p, q = fraction              # p = 10, q = 7
```

Remarques

Le nombre de variables à gauche de l'affectation doit correspondre au nombre d'éléments du tuple, sinon une erreur se produit :

```
>>> x, y = (1, 2, 3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: too many values to unpack (expected 2)
```

- Un p-uplet peut contenir un p-uplet comme élément. L'affectation multiple peut alors récupérer ce tuple, ou bien ses élément.

```
personne = ('Lovelace', 'Ada', (10, 'décembre', 1815), 'Londres')
nom, prenom, date, lieu = personne # date vaut (10, 'décembre', 1815)
# ou bien :
nom, prenom, (jour, mois, an), lieu = personne
```

- L'affectation multiple permet d'échanger les valeurs de deux variables : `a, b = (b, a)`. Les parenthèses sont optionnelles et on peut écrire `a, b = b, a`