Cours 10 - Programmation LISP (V)

Marie-Hélène Abel

Table des matières

Compléments

I

A. Les macro-caractères : ` , @

- Pour faciliter la manipulation d'expressions, il existe une fonctionnalité en Common Lisp appelée backquote.
- Avec cette fonctionnalité, il est possible de créer l'expression finale à partir d'un patron.
- Son utilisation est similaire à celle de quote.

Syntaxe

Pour backquote il y a un caractère, l'apostrophe à l'envers, à utiliser qui signifie le début du patron : `

Attention : Association des macro-caractères : , `

Une expression à l'intérieur qui commence par le caractère virgule sera évaluée et remplacée par sa valeur.

Exemple

```
>(setq a 2)
2
>(setq b 3)
3
>`(,a + ,b = ,(+ a b))
(2 + 3 = 5)
>
```

Remarque

Pour substituer une suite d'expressions dans le patron, il n'est pas possible d'utiliser la virgule, car on obtient alors une liste des expressions plutôt que la suite elle-même.

Association des macro-caractères ,@

Pour cette raison, la fonctionnalité backquote autorise la suite des deux caractères : ,@

Exemple

```
>(setq nom '(Charlie Brown))
(Charlie Brown)
>`(mon nom est ,@nom)
(mon nom est Charlie Brown)
>
```

B. Propriétés des variables

Les variables peuvent être obligatoires, optionnelles, qualifiées de queue, repérées par des clés ou encore auxiliaires.

1. Variables obligatoires

Rappel

Il peut y avoir 0 ou plus variables obligatoires.

Exemple

```
>(defun ff (a b c d)(list a b c d))
FF
>
```

2. Variables optionnelles

&optional

Certaines variables ou paramètres peuvent être déclarés comme étant optionnels, c'est-à-dire que leur présence n'est pas obligatoire lors de l'appel de la fonction.

Exemple

```
>(defun ff (a b &optional c d)(list a b c d))
FF
>(ff 1 2 3)
(1 2 3 NIL)
>
```

Valeur par défaut

Il est possible de donner une valeur d'initialisation à un paramètre optionnel lorsqu'il n'est pas présent au moment de l'appel de la fonction.

Exemple

```
>(defun ff (a b &optional c (d (list 4)))
(list a b c d))
FF
>(ff 1 2 3)
(1 2 3 (4))
>
```

Vérification de la présence de variables

Il est possible de préciser une variable booléenne dont la valeur sera vraie si le paramètre était présent au moment de l'appel de la fonction et faux dans le cas contraire.

Exemple

```
>(defun ff (a b &optional c (d (list 4) was-there?))
(list a b c d was-there?))
FF
>(ff 1 2 3)
(1 2 3 (4) NIL)
>(ff 1 2 3 (list 4))
(1 2 3 (4) T)
>
```

3. Variables de queue

&rest

Il est possible de préciser une variable qui prend le reste des variables d'une liste.

Exemple

```
>(defun ff (a b &rest ll)(list a b ll))
FF
>(ff 1 2 3 4)
(1 2 (3 4))
>
```

Remarque

On peut mélanger les deux options précédentes.

Exemple

```
>(defun ff (a b &optional (c 3) &rest ll)
(list a b c ll))
FF
>(ff 1 2 )
(1 2 3 NIL)
>(ff 1 2 3 4 5)
(1 2 3 (4 5))
>
```

4. Variables repérées par des clés

&key

Il est possible de repérer les arguments par des clés en utilisant l'option &key.

Exemple

```
> (defun ff (a b &key c d)
```

```
(list a b c d))
FF
>(ff 1 2 :d 5)
(1 2 NIL 5)
> (ff 1 2 :d 4 :c 3)
(1 2 3 4)
>
```

5. Variables auxiliaires

&aux

Cette option permet simplement d'alléger l'écriture des fonctions.

Exemple

```
> (defun ff1 (a b &aux (c 3) d)
(list a b c d))
FF1
; est strictement équivalent à
> (defun ff2 (a b)
(let* ((c 3) d)(list a b c d)))
FF2
>(ff1 1 2 )
(1 2 3 NIL)
>(ff2 1 2)
(1 2 3 NIL)
>
```