言語まとめ Lisp

[言語まとめ][Lisp]

以下から参照、引用

- Emacs Lisp Programming <http://www.bookshelf.jp/texti/elisp-intro/jp/emacs-lisp-intro.html>
- Common Lisp 入門 <http://wisdom.sakura.ne.jp/programming/lisp/index.html>

リスト処理

リスト

- リストを以下のように書く

```
'(rose violet daisy buttercup)
```

- データとプログラムのどちらも同じ方法で表現する。どちらも、単語や数やリストを空白で区切って括弧で囲んだリストである。

リストの直前に付けた1つのアポストロフィ'をクォートと呼び、リストの直前にこれを付けると、そのリストに関しては何もせずに字面どおりに扱うことをLispに指示する。リストの直前にクォートがない場合には、リストの先頭要素を特別扱いし、コンピュータが従うべきコマンドとなる

アトム

- これまで単語と呼んできたものをアトム (atoms) と呼ぶ。
- リストでは、アトムを空白で区切る。アトムは、括弧のすぐ隣にあってもよい。
- リストは、たった1個のアトムを含むだけでも、まったく含まなくてもよい。
- 何も含まないリストは () のように書き、空リスト (empty list) と呼ぶ。
- 空リストは、それ以外のものとは異なり、アトムであると同時にリストでもある。

プログラムの実行

- リストの直前にクォートがない場合には、リストの先頭要素を特別扱いし、コンピュータが従うべきコマンドとなる。

```
>(+ 1 2)
```

```
3
```

シンボル名と関数定義

- + のようなシンボルは、それ自身はコンピュータが実行すべき命令列ではない
- 定義、すなわち、命令列を探すためにシンボルを (一時的に) 使う

インタープリタ

基本動作

1. リストの直前にクォートがあるかどうかを調べる。
2. クォートがあれば、リストを返すだけ。
3. クォートがなければ、インタープリタはリストの先頭要素を調べ、それに関数定義がある

かどうかを調べる。

4. 関数定義があれば、インタプリタはその関数定義内の命令列を実行する。
5. さもないければ、インタプリタはエラーメッセージを出力する。

特別な動作

変数

- ・ リストに加えて、クオートもせず括弧で囲まれてもいないシンボルを評価できる
- ・ 変数 (variable) としてのシンボルの値を決定しようとする

スペシャルフォーム

- ・ 特殊な関数があり、これらは普通的方式のように動作しない。これらをスペシャルフォーム (special forms) と呼ぶ。関数の定義などの特殊なことを行うものであり、それらの個数は多くはない。

ネストしたリスト

- ・ インタプリタが探しあてた関数がスペシャルフォームでなく、しかも、それがリストの一部である場合には、Lisp インタプリタはリストの内側にリストがあるかどうかを調べる。内側にリストがあれば、Lisp インタプリタは内側のリストを処理してから、外側のリストを処理する。

引数

データ型

- ・ 関数に渡すデータの型は、関数を使用する情報の種類に応じて決まる。
- ・ + は数を加算するので、+ のような関数への引数は数値である必要がある。

引数としての変数やリストの値

- ・ 引数は、評価したときに値を返すシンボルでもよい。

可変個数の引数

- ・ +、* などのある種の関数は、任意個数の引数を取る

変数

値の設定

- ・ set か setq を使う
- ・ let を使う

set の使い方

- ・ シンボル flowers の値としてリスト '(rose violet daisy buttercup) を設定する

```
>(set 'flowers '(rose violet daisy buttercup))
(ROSE VIOLET DAISY BUTTERCUP)
>'flowers
FLOWERS
>flowers
(ROSE VIOLET DAISY BUTTERCUP)
```

set を使う場合、いずれの引数も評価してほしくない場合には、両方の引数をクオートする必要がある

setq の使い方

- ・ 実用上、set の第 1 引数をほとんどつねにクオートするはず
- ・ set で第 1 引数をクオートする組み合わせは多用されるので、スペシャルフォーム setq が用意してある
- ・ set とほとんど同じであるが、第 1 引数を自動的にクオートするので、引用符をタイプする必要はない

```
>(setq flowers '(rose violet daisy buttercup))
(ROSE VIOLET DAISY BUTTERCUP)
>flowers
(ROSE VIOLET DAISY BUTTERCUP)
```

- ・ さらに、setq は、複数の異なる変数に異なる値を代入するためにも使える。

```
>(setq alpha '(a b c)
      num '(1 2 3))
(1 2 3)
>alpha
(A B C)
>num
(1 2 3)
```

関数定義

defun

- ・ シンボルには、関数として呼ばれたときにコンピュータが実行するコードが結び付けられている。
- ・ このコードを関数定義 (function definition) と呼び、シンボル defun (define function (関数を定義する) の略) で始まる Lisp の式を評価することで作成する
- ・ defun は、その引数を通常のように評価しないので、スペシャルフォーム (special form) と呼ばれる。

書式

defun 関数名 (引数) " 関数の説明 (省略可)" 関数本体

例

```
>(defun pow (x) "power" (* x x))
POW
>(pow 2)
4
```

条件判定

if

書式

- ・第 1 要素が if であるリストの第 2 要素と第 3 要素のそれぞれに、判定条件と真の場合の動作を指定する。

```
(if 判定条件
  真の場合の動作)
```

例

```
>(if (> 5 4)
    '("5 is greater than 4"))
("5 is greater than 4")
>(if (> 5 6)
    '("5 is greater than 4"))
NIL
```

If--then--else 式

- ・ if 式には第 3 引数を指定することもでき、判定条件が偽の場合の動作である。

```
>(setq x 2)
2
>(if (= x 1)
    '("x is 1")
    '("x is not 1"))
("x is not 1")
```

偽について