cons, append, listの違い

- (cons 'p '(a b c)) ⇒ (p a b c)
- (cons '(a b c) 'p) ⇒ ((a b c) . p)
- (append '(a b) '(c d)) ⇒ (a b c d)
- (append 'a '(b c)) ⇒ エラー (Aはリストじゃない！)
- (append '(a b) 'c) ⇒ (a b . c)
  appendは第一引数のリストをコピーする非破壊的関数
- (list 'a '(b c)) ⇒ (a (b c))

48

- consは新たなconsセルを1つ用意し，引数をくっつける．リストの先頭に要素を足す時に適している．
- appendは第一引数のリストをコピーすることで2つのリストを結合する．
- listは任意個の引数をその数だけの新たなconsセルで結合する．各consセルのcar部は対応する引数を指す．最後のconsセルのcdr部はnilを指す．

49

代入

- (setf vowels '(a i u e o))
  ⇒ (A I U E O)
  最初の引数は変数名でevalしない.
  2番目の引数はevalしてから変数に代入する.
  setfの返値は変数に代入した値.
  大域変数の設定に使おう！

50

局所変数

- (defun average (x y)
  (let ((sum (+ x y)))
  (list x y 'average 'is (/ sum 2.0))))
- (defun price-change (old new)
  (let* ((diff (- new old))
  (proportion (/ diff old))
  (percentage (* proportion 100.0)))
  (list 'changed 'by
  percentage 'percent)))

51

ヘルプ機能

- 組込み関数と組込み変数の説明
  (documentation 'cons 'function)
  ⇒ "(cons x y) returns a list with x as the car and y as the cdr"
- 自分で定義した関数の説明
  (defun average (x y)
  "xとyの平均値を返す"
  (/ (+ x y) 2.0))
  (documentation 'average 'function)
  ⇒ "xとyの平均値を返す"

52

- (apropos "TOTAL" "USER") ⇒
  ARRAY-TOTAL-SIZE (function)
  ARRAY-TOTAL-SIZE-LIMIT, constant, value: 134217727

53

注釈

- ; (セミコロン) の後は行末まで注釈とみなされる

```
;;; function to compute Einstein's E=mc^2
(defun  einstein  (m)
  (let  ((c  300000.0)) ; speed of light km/s
    ;;  E is energy
    ;;  m is mass
    (*  m  c  c))
```

54

リスト操作の関数 (続き)

- reverse　等
- 集合演算
- 表形式のデータ
- 木構造データ

55

プリント4