

# Basic Lisp Overview

## ■ Numeric Functions

**\***, **+**, **-**, **/** - returns product, sum, difference, or quotient

(**\*** 2 3 4)  $\Rightarrow$  24  
(**/** (+ 2 2) (- 3 1))  $\Rightarrow$  2

**sqrt** - square root of number (sqrt 9)  $\Rightarrow$  3

**expt** - (expt *Base Exponent*)  $\Rightarrow$  *Base*<sup>*Exponent*</sup>  
(expt 10 3)  $\Rightarrow$  1000

**min**, **max** - minimum or maximum of numbers  
(min -1 2 -3 4 -5 6)  $\Rightarrow$  -5

**abs**, **mod**, **round** - absolute value, mod, nearest int  
(round (abs -4.2))  $\Rightarrow$  4

**sin**, **cos**, **tan** - trig functions. Arguments in radians, **not** degrees.  
(sin (/ pi 2))  $\Rightarrow$  1.0 ; PI is built-in variable

## ■ List Access Functions

**first** - returns first element of a list. Use instead of CAR.

(first '(A B C D))  $\Rightarrow$  A

**second**, **third**, ..., **tenth** - analogous to "first": (third '(A B C D))  $\Rightarrow$  C

**nth** - (nth *N List*)  $\Rightarrow$  *N*th entry of *List*. Note that *N* starts at 0, not 1.  
(nth 2 '(A B C D))  $\Rightarrow$  C

**rest** - returns all but 1st element of a list. Use instead of CDR.  
(rest '(A B C D))  $\Rightarrow$  (B C D)

**last** - returns **list** of last element of a list  
(last '(A B C D))  $\Rightarrow$  (D)

**length** - returns the number of top-level entries in list  
(length '(A (B C) (D E)))  $\Rightarrow$  3

## ■ List Construction Functions

**cons** - (cons *Entry List*)  $\Rightarrow$  (*Entry List*)

(cons 'A '(B C D))  $\Rightarrow$  (A B C D)  
(cons (first '(A B C)) (rest '(A B C)))  $\Rightarrow$  (A B C)

**append** - (append (*List1*) (*List2*))  $\Rightarrow$  (*List1 List2*)  
(append (L1) (L2) (L3)...(LN))  $\Rightarrow$  (L1 L2 L3 ... LN)  
(append '(A B) '(C D))  $\Rightarrow$  (A B C D)

For CONS and APPEND, if the second arg is not a list, you will get an odd result that looks like a list but has a dot before the last element.

**list** - (list *Entry1 E2 ... EN*)  $\Rightarrow$  (*Entry1 E2 ... EN*)  
(list 'A '(B C) (+ 2 3))  $\Rightarrow$  (A (B C) 5)

## ■ Predicates

**Type-checking Predicates:** **listp**, **numberp**, **integerp**, **stringp**, **atom**  
test if arg is a list, number, integer, string or atom, respectively.

(numberp 5.78)  $\Rightarrow$  t (integerp 5.78)  $\Rightarrow$  NIL

**Numeric Predicates:** **evenp**, **oddp**, **=**, **<**, **>**, **<=**, **>=**  
(oddp 7)  $\Rightarrow$  t (> 7 6)  $\Rightarrow$  t

These will all give errors for non-numbers.

**General Predicates:** **null**, **equal**, **eql** - test if arg is NIL or if two arguments have the same value. EQL does **not** work on lists or strings.

(null (rest '(A)))  $\Rightarrow$  t  
(equal '(A B) (cons 'A '(B)))  $\Rightarrow$  t  
(eql 'A 'A)  $\Rightarrow$  t  
(eql '(A B) (cons 'A '(B)))  $\Rightarrow$  NIL

**Logical Predicates:** **and**, **or**, **not**

(not (and (= 7 (+ 2 5)) (evenp 8)))  $\Rightarrow$  NIL

## ■ Special Forms

Special forms are used for side effects, and don't follow the normal

Lisp rule of evaluating all the args before applying function to the results.

**setq** (or **setf**) - assigns a value to a variable  
(setq Foo 'Bar)  $\Rightarrow$  BAR (list Foo 'Foo)  $\Rightarrow$  (BAR FOO)

**"' "** (or **quote**) - returns argument literally  
'(+ 2 3)  $\Rightarrow$  (+ 2 3) (+ 2 3)  $\Rightarrow$  5

**defun** - defines a function.

(defun *Function-Name* (*Arguments*) *Body*) The value the function returns is the value of the last form in the *Body*.

(defun Square (Num) (\* Num Num))  
(Square 7)  $\Rightarrow$  49

**if** - the most basic conditional operator.

(if *Form1* usually read as (if *Condition*  
*Form2* *Then-Result*  
*Form3* *Else-Result*)

Means to evaluate *Form1*. If its value is "true" (non-NIL), then evaluate and return *Form2*, otherwise evaluate and return *Form3* (or NIL if *Form3* is missing).

(if (= 7 (+2 4)) 'yes 'no)  $\Rightarrow$  NO

**cond** - multiple if-then-else conditional operator.

(cond (*Test1 Result1*)  
(*Test2 Result2*)  
...  
(*TestN ResultN*))

This evaluates each of *Test1* through *TestN* in order. The first one it finds that is "true" (non-NIL), it evaluates and returns the associated *Result*. No further *Tests* or *Results* are evaluated. If you have multiple results associated with a single test, each is evaluated and the value of the last one is returned.

(setq Test 7)  
(cond ((not (numberp Test)) "Not a number!")  
(oddp Test) (+ Test 1))  
(t Test))  
 $\Rightarrow$  8

**progn** - Group multiple commands into a single block, returning the value of the final one. Some constructs do this implicitly.

**loop** - The infamous all-in-one iteration construct. See handout.

## ■ Miscellaneous

**load** - loads the indicated file, evaluating all Lisp forms in file.

**compile-file** - takes the indicated source file (xxx.lisp) and produces a compiled file (xxx.wfasl). Does **not** load this compiled file.

**print**, **format** - prints output. See separate handout on FORMAT.

(print "Hello")  
"Hello" ; prints on screen, is NOT return value  
 $\Rightarrow$  "Hello" ; return value (rarely used)

**On-line help:**

**apropos** - finds functions/variables containing substring  
(apropos 'concat 'user) gives all functions containing "concat" in the default ("user") package, including "concatenate"  
**documentation** - prints the doc-string for a function. E.g.  
(documentation 'concatenate 'function)

**Debugger options:** :A - Abort out of debugger

:B - Backtrace (list previous calls)

:N - Next (earlier) entry on stack

:P - Previous (later) entry on stack

:? - more debugger options

**bye** - quits Harlequin lisp (Harlequin specific).