LISP SYNTAX CHEAT SHEET

FUNCTION	**EXAMPLE**	**OUTPUT**
+, -, *, /	(+ 2 2)	4
=, >, <	(> 1 2)	nil
abs	(abs -3)	3
and	(and (> 2 1) (< 2 3))	t
append	(append '(a b) '(c d))	(a b c d)
apply	(apply #'+ '(2 2))	4
atom	(atom 'x)	t
car	(car '(a b))	a
cdr	(cdr '(a b))	(b)
cond	(cond ((atom 'x) 2) …)	2
cons	(cons 'a '(b))	(a b)
defmethod	(defmethod oldp ((p PERSON)) (> (person-age p) 25))	
defstruct	(defstruct person age height)	PERSON
defun	(defun funky-fun (x y) (+ x y))	FUNKY-FUN
dolist	(dolist (x '(a b c)) (...))	nil
dotimes	(dotimes (x 5) (…))	nil
eq	(eq 'x 'x)	t
equalp	(equalp '(5 4 (3 2 1)) '(5 4 (3 2 1)))	t
error	(error "quit")	Error: quit
first	(first '(a b c))	a
format	(format t "~s~%" 3)	3, nil
if	(if (> 1 2) 3 4)	4
incf	(incf (car '(1)) 5)	6
lambda	(let ((fun (lambda (x) (+ x 2)))) (apply fun '(4)))	9
let	(let ((x 2) (y 3)) (+ x y))	5
list	(list 1 2 3)	(1 2 3)
listp	(listp '(1 2 3))	t
make-person*	(make-person :height 6 :age 25)	#S(PERSON
mapcar	(mapcar #"listp '((a b) c (d e))	(t nil t)
max	(max 1 2 3)	3
nil	(null nil)	t
not	(not nil)	t
nth	(nth 4 '(5 6 7 8 9 10))	9
null	(null 1)	nil
numberp	(numberp 4)	t
oddp, evenp	(oddp 6)	nil
or	(or t nil)	t
person-age*	(person-age (make-person :age 20))	20
person-p*	(person-p (make-person :height 6))	t
second	(second '(a b c))	b
setf	(setf five 5)	5
t	(or t …)	t
third	(third '(a b c))	c
truncate	(truncate 5 2)	2, 1
zerop	(zerop 3)	nil

* These assume you have already called (defstruct person age height)