FUNCIONES LÓGICAS	OBSERVACIONES
(= cadnúm [cadnúm] ...) *Devuelve T si todos los argumentos son numéricamente iguales, de lo contrario, devuelve nil*	• Todos los argumentos cadnúm pueden ser números o cadenas.
(/= cadnúm [cadnúm[...) *Devuelve T si los argumentos no son numéricamente iguales y nil en caso contrario*	• Todos los argumentos cadnúm pueden ser números o cadenas.
(< cadnúm [cadnúm] ...) *Devuelve T si cada argumento es numéricamente menor que el situado detrás de él y nil en caso contrario*	• Todos los argumentos cadnúm pueden ser números o cadenas.
(<= cadnúm [cadnúm] ...) *Devuelve T si cada argumento es numéricamente menor o igual que el situado detrás de él y nil en caso contrario*	• Todos los argumentos cadnúm pueden ser números o cadenas.
(> cadnúm [cadnúm] ...) *Devuelve T si cada argumento es numéricamente mayor que el situado detrás de él y nil en caso contrario*	• Todos los argumentos cadnúm pueden ser números o cadenas.
(>= cadnúm [cadnúm] ...) *Devuelve T si cada argumento es numéricamente mayor o igual que el situado detrás de él y nil en caso contrario*	• Todos los argumentos cadnúm pueden ser números o cadenas.
(and expr ...) *Devuelve el operador lógico AND de una lista de expresiones*	• Si alguna de las expresiones da como resultado nil, la función devuelve nil; en caso contrario, devuelve T.
(Boole func ent1 ent2 ...) *Se utiliza como función booleana basada en bits de tipo general*	• Ver manual de personalización
(eq expr1 expr2) *Determina si dos expresiones son idénticas*	• La función eq determina si expr1 y expr2 están asociadas al mismo objeto (mediante setq, por ejemplo). Devuelve T si las dos expresiones son iguales y nil en caso contrario
(equal expr1 expr2 [aproximación]) *Determina si dos expresiones son iguales*	• La función equal determina si expr1 y expr2 se evalúan igual. Cuando se comparan dos números reales (o dos listas de números reales, como en puntos), los dos números idénticos pueden presentar ligeras diferencias si no se han utilizado los mismos métodos para su cálculo. Por lo tanto, puede utilizar un argumento numérico optativo, aproximación, para especificar la diferencia máxima admitida entre expr1 y expr2, para que sigan considerándose iguales.
(not elemento) *Devuelve la negación lógica de la expresión*	• Devuelve T si elemento se evalúa como nil y devuelve nil en caso contrario.
(or expr ...) *Devuelve el OR lógico de una lista de expresiones*	• La función o calcula las expresiones de izquierda a derecha en busca de una expresión distinta de nil. Si la encuentra, or deja de realizar cálculos y devuelve T. Si el valor de todas las expresiones es nil, or devuelve nil.

EJEMPLOS DE FUNCIONES LÓGICAS

(setq a 23 b 23.001 c 25.0 d 25 e (- 13.44))
-13.44
(= c d)
T
(equal c d)
T
(equal a b)
nil

(equal a b 0.01)
T

(< e a b c)
T

(< e a b c d)
nil

(<= e a b c d)
T

(and (= c d) (<= e a b c d))
T

Las funciones lógicas LISP evalúan como suceso cierto (T) cualquier expresión que no valga explícitamente NIL. Por esa razón pueden llegar a tener sentido expresiones como las siguientes, donde una de las premisas de la operación NOT, AND u OR es un valor numérico:
(and (= c d) (setq m (+ a b)))
T

(or (< c 10) (max c d))
T

(not (> a b))
T
(not (setq m 35))
nil
El nivel de anidamiento es ilimitado:
(or (> a b) (and (= b c) (= c d)) (not (equal a b 0.01)))
nil