

Scheme Reference Card

(keyed to a purely-functional subset of PLT Pretty Big Scheme)

Compiled by Robert Keller for HMC CS 60

Last update: 15 January 2007

Literals	
Form	Description
#f #t	are truth values false and true (aka “Booleans”). Also, any value that is not #f will be interpreted as true.
1 42 -37	are integers
5/7 -2/4	are rationals
1e12 0.5e-2 -3.14e8	are reals
1.0+1.0i	are complex
‘foo	is a symbol (sort of like a string, but with special properties)
‘(foo 1 2 3.4 #f)	is a list of literals

Defining Functions	
Form	Description
(define s₁ expr)	Define symbol s ₁ to have the value of expression expr.
(define (f₁ a₁ . . .) expr)	Define function named f ₁ with zero or more arguments a ₁ . . . to have the value of expression expr.
(lambda (a₁ . . .) expr)	Evaluates to an anonymous function which, with arguments a ₁ . . . evaluates to the value of expr.

Other Special Forms	
Form	Description
(if c₁ e₂ e₃)	Evaluates c ₁ . If the result is not #f, returns the value of e ₂ , otherwise returns the value of e ₃ .
(cond (c₁ e₁) . . . (else e_n))	Evaluates c ₁ . If the result is not #f, returns the value of e ₁ , otherwise evaluates c ₂ , etc. If no value of a c _i is not #f, returns the value of e _n .
(case e₀ (v₁ e₁) . . .)	Evaluates e ₀ , then compares the result to the values v ₁ . . . until a match v _i is found, in which case e _i is returned. If there is no match, the result is undefined, so better be sure there is a match.
(let ((s₁ e₁) . . .) expr)	Evaluates the expressions e ₁ . . . , binding symbols s ₁ . . . locally to their values, then evaluates expr in the resulting environment.
(let* ((s₁ e₁) . . .) expr)	Similar to let, except the environment accumulates as successive expressions bindings are made. That is, the value of e ₂ is computed with the binding of s ₁ , etc.
(letrec ((s₁ e₁) . . .) expr)	Similar to let, except that the environment in which the expressions are evaluated includes all the bindings created by the bindings themselves.

Logic Functions	
Form	Description
(and $n_1 \dots$)	Return #f (false) if some argument is #f, otherwise return the last argument.
(or $n_1 \dots$)	Return the first value that is not #f, otherwise return #f.
(not n_1)	Return #t if the argument is #f, otherwise return #f.
(boolean? n_1)	Return #t if the argument is #t or #f, otherwise return #f.

Numeric Functions	
Form	Description
(+ $n_1 \dots$)	Return the sum of numbers $n_1 \dots$.
(- $n_1 n_2 \dots$)	Return the first number n_1 minus the sum of the remaining numbers $n_2 \dots$.
(-) n_1)	Return the negative of the number n_1 .
(* $n_1 \dots$)	Return the product of numbers $n_1 \dots$.
(/ $n_1 n_2 \dots$)	Return the first number n_1 divided by the product of the remaining numbers $n_2 \dots$.
(/) n_1)	Return the reciprocal of the number n_1 .
(max $n_1 \dots$)	Returns the maximum of numbers $n_1 \dots$.
(min $n_1 \dots$)	Returns the minimum of numbers $n_1 \dots$.
(modulo $n_1 n_2$)	Return the remainder of dividing n_1 by n_2 .
(= $n_1 n_2$)	Return #t if numbers n_1 and n_2 are equal, otherwise #f.
(< $n_1 n_2$)	Return #t if number n_1 is less than n_2 , otherwise #f.
(> $n_1 n_2$)	Return #t if number n_1 is greater than n_2 , otherwise #f.
(<= $n_1 n_2$)	Return #t if number n_1 is less than or equal n_2 , otherwise #f.
(>= $n_1 n_2$)	Return #t if number n_1 is greater than or equal n_2 , otherwise #f.
(integer? n_1)	Return #t if number n_1 is an integer, otherwise #f.
(rational? n_1)	Return #t if number n_1 is a rational, otherwise #f. (Note: Integers are rational.)
(real? n_1)	Return #t if number n_1 is a real, otherwise #f. (Note: Rationals are real.)
(complex? n_1)	Return #t if number n_1 is a real otherwise #f. (Note: Reals are complex.)
(exact? n_1)	Return #t if number n_1 is exact, otherwise #f.
(inexact? n_1)	Return #t if number n_1 is not exact, otherwise #f.
(zero? n_1)	Return #t if number n_1 is some kind of zero, otherwise #f.
(positive? n_1)	Return #t if <i>real</i> number n_1 is positive, otherwise #f.
(negative? n_1)	Return #t if <i>real</i> number n_1 is negative, otherwise #f.
(odd? n_1)	Return #t if <i>integer</i> n_1 is odd, otherwise #f.
(even? n_1)	Return #t if <i>integer</i> n_1 is even, otherwise #f.
(abs n_1)	Return the absolute value of <i>real</i> number n_1 .
(magnitude n_1)	Return the magnitude of <i>complex</i> number n_1 .
(angle n_1)	Return the angle in radians of the polar coordinates for <i>complex</i> number n_1 .
(make-rectangular $r_1 r_2$)	Make a complex number from reals as rectangular

	coordinates.
(make-polar $r_1 r_2$)	Make a complex number from reals as polar coordinates.
(quotient $n_1 n_2$)	Return the quotient of dividing integer n_1 by integer n_2 .
(remainder $n_1 n_2$)	Return the remainder of dividing integer n_1 by integer n_2 .
(floor n_1)	Returns the greatest integer less than or equal to n_1 .
(ceiling n_1)	Returns the least integer greater than or equal to n_1 .
(truncate n_1)	Returns the result of discarding any fractional part of n_1 .
(round n_1)	Returns the result of rounding the fractional part of n_1 to the nearest integer.
(gcd $n_1 \dots$)	Returns the greatest common divisor of numbers $n_1 \dots$.
(lcm $n_1 \dots$)	Returns the least common multiple of numbers $n_1 \dots$.
(numerator n_1)	Returns the numerator of a rational number n_1 .
(denominator n_1)	Returns the denominator of a rational number n_1 .
(expt $n_1 n_2$)	Returns the number n_1 raised to the power n_2 . (Complex powers are allowed).
(exp n_1)	Returns e to the power n_1 .
(log n_1)	Returns the natural logarithm of n_1 .
(sqrt n_1)	Returns the square root of n_1 .
(real-part n_1)	Returns the real part of complex number n_1 .
(imag-part n_1)	Returns the imaginary part of complex number n_1 .
(inexact->exact n_1)	Returns a rational version of inexact number n_1 .
(exact->inexact n_1)	Returns an inexact version of number n_1 .
(sin n_1) (cos n_1) (tan n_1) (asin n_1) (acos n_1) (atan n_1)	Returns the sin, etc. of number n_1 .

List Functions	
Form	Description
(list $a_1 \dots$)	Evaluate the arguments $a_1 \dots$, forming a list of the results.
(cons $a_1 L_2$)	Form a list with the value of a_1 as the first element and list L_2 as the rest of the list. (If L_2 is not a list, will return a “dotted pair” of the values.)
(null? L_1)	Returns $\#t$ if the value of L_1 is the empty list, $\#f$ otherwise.
(first L_1)	Return the first element of non-empty list L_1 .
(rest L_1)	Return a list of all but the first element of non-empty list L_1 .
(list-ref $L_1 n_2$)	Returns the n_2 th element of list L_1 , where n_2 is a non-negative integer. The length of L_1 must be at most $n_2 - 1$.
(second L_1) (third L_1) \dots (eighth L_1)	Return the indicated element of list L_1 , which must have enough elements.
(length L_1)	Return the length of list L_1 .
(append $L_1 \dots$)	Create a list of the elements of L_1 followed by those of the remaining arguments.
(member $a_1 L_2$)	Return the first suffix of list L_2 that begins with a_1 . If there is no such suffix, return the empty list.

(map $f_1 L_2 \dots$)	Apply the function f_1 element-wise to the lists. The arity of f_1 must be equal to the number list arguments, and each list must be of the same length.
(apply $f_1 L_2$)	Apply the function f_1 to the list of arguments L_2 . The length of the list must be the same as the arity of the function. If is a built-in, then the list can be any length compatible with that function.
(foldl $f_1 v_2 L_3$)	Apply the binary function f_1 to the list of elements, grouping from the left. The value v_2 must be a unit element for the function f_1 .
(list? a_1)	Return #t if a_1 is a proper list, otherwise #f.
(pair? a_1)	Return #t if a_1 is a pair (created by cons), otherwise #f.

Equality	
Form	Description
(equal? $a_1 a_2$)	Return #t if data values a_1 and a_2 are equal, otherwise #f.
(eq? $a_1 a_2$)	Return #t if data values a_1 and a_2 are in the same memory locations, otherwise #f.
(= $n_1 n_2$)	Return #t if <i>numbers</i> n_1 and n_2 are equal, otherwise #f.
(string=? $s_1 s_2$)	Return #t if <i>strings</i> s_1 and s_2 are equal, otherwise #f.

String Functions	
Form	Description
(string->list e_1)	Evaluate e_1 , which should yield a string, then return the list of chars in the string.
(symbol->string e_1)	Evaluate e_1 , which should yield a symbol, then return the string comprised of chars in the symbol.
(string->symbol e_1)	Evaluate e_1 , which should yield a string, then return the symbol constructed from chars in the string.
(string->number e_1)	Evaluate e_1 , which should yield a string, then return a number obtained by converting the chars in the string.
(string->append $e_1 \dots$)	Form a new string from the characters of the argument values.
(string<=? $e_1 e_2$) (string<? $e_1 e_2$) (string=? $e_1 e_2$) (string>=? $e_1 e_2$) (string>? $e_1 e_2$)	Compare strings in apparent ways.
(substring $e_1 n_2 n_3$)	Return the sub-string of string e_1 , from chars n_2 through n_3 , the first char having index 0.
(substring $e_1 n_2$)	Return the sub-string of string e_1 , from chars n_2 through the end of the string.
(string-length e_1)	Return the length of string e_1 .
(string-ref $e_1 n_2$)	Return the char of string e_1 at index n_2 , the first char having index 0.