Programmazione

Assegnazioni

lhs = rhs evaluates rhs and assigns the result to be the value of lhs. From then on, lhs is replaced by rhs whenever it appears.
{l1, l2, ...} = {r1, r2, ...} evaluates the ri, and assigns the results to be the values of the corresponding li.

lhs := rhs assigns rhs to be the delayed value of lhs. rhs is maintained in an unevaluated form. When lhs appears, it is replaced by rhs, evaluated afresh each time.
lhs := rhs /; test is a definition to be used only if test yields True.

lhs =. removes any rules defined for lhs.

Clear[symbol1, symbol2, ...] clears values and definitions for the specified symbols.

ClearAll[symbol1, symbol2, ...] clears all values, definitions, attributes, messages and defaults associated with symbols.

Attributi e opzioni

Attributes[symbol] gives the list of attributes for a symbol.

ClearAttributes[s, attr] removes attr from the list of attributes of the symbol s.

SetAttributes[s, attr] adds attr to the list of attributes of the symbol s.

Flat is an attribute that can be assigned to a symbol f to indicate that all expressions involving nested functions f should be flattened out. This property is accounted for in pattern matching.

HoldAll is an attribute which specifies that all arguments to a function are to be maintained in an unevaluated form.

HoldFirst is an attribute which specifies that the first argument to a function is to be maintained in an unevaluated form.

HoldRest is an attribute which specifies that all but the first argument to a function are to be maintained in an unevaluated form.

Listable is an attribute that can be assigned to a symbol f to indicate that the function f should automatically be threaded over lists that appear as its arguments.

Locked is an attribute which, once assigned, prevents modification of any attributes of a symbol.

OneIdentity is an attribute that can be assigned to a symbol f to indicate that f[x], f[f[x]], etc. are all equivalent to x for the purpose of pattern matching.

Orderless is an attribute that can be assigned to a symbol f to indicate that the elements ei in expressions of the form f[e1, e2, ...] should automatically be sorted into canonical order. This property is accounted for in pattern matching.

Protected is an attribute which prevents any values associated with a symbol from being modified.

Options[symbol] gives the list of default options assigned to a symbol.

SetOptions[s, name1->value1, name2->value2, ...] sets the specified default options for a symbol s.

Funzioni

Function[body] or body& is a pure function. The formal parameters are # (or #1), #2, etc.
Function[x, body] is a pure function with a single formal parameter x.
Function[{x1, x2, ...}, body] is a pure function with a list of formal parameters.
Function[{x1, x2, ...}, body, {attributes}] has the given attributes during evaluation.

Compile[{x1, x2, ...}, expr] creates a compiled function which evaluates expr assuming numerical values of the xi.

Compile[{{x1, t1}, ...}, expr] assumes that xi is of a type which matches ti.
Compile[vars, expr, {{p1, pt1}, ...}] assumes that subexpressions in expr which match pi are of types which match pti.

Identity[expr] gives expr (the identity operation).

Operatori

Apply[f, expr] or f @@ expr replaces the head of expr by f.
Apply[f, expr, levelspec] replaces heads in parts of expr specified by levelspec.

Composition[f1, f2, f3, ...] represents a composition of the functions f1, f2, f3, ...

Inner[f, list1, list2, g] is a generalization of Dot in which f plays the role of multiplication and g of addition.

Nest[f, expr, n] gives an expression with f applied n times to expr.

NestList[f, expr, n] gives a list of the results of applying f to expr 0 through n times.

Map[f, expr] or f /@ expr applies f to each element on the first level in expr.
Map[f, expr, levelspec] applies f to parts of expr specified by levelspec.

MapAll[f, expr] or f //@ expr applies f to every subexpression in expr.

MapAt[f, expr, n] applies f to the element at position n in expr. If n is negative, the position is counted from the end.
MapAt[f, expr, {i, j, ...}] applies f to the part of expr at position {i, j, ...}.
MapAt[f, expr, {{i1, j1, ...}, {i2, j2, ...}, ...}] applies f to parts of expr at several positions.

MapThread[f, {{a1, a2, ...}, {b1, b2, ...}, ...}] gives {f[a1, b1, ...], f[a2, b2, ...], ...}.
MapThread[f, {xa, xb, ...}, n] maps f over the nth level of the n-dimensional tensors xa, xb, ...

Operate[p, f[x, y]] gives p[f][x, y].
Operate[p, expr, n] applies p at level n in the head of expr.

Outer[f, list1, list2, ...] gives the generalized outer product of the listi.

Thread[f[args]] ``threads'' f over any lists that appear in args.
Thread[f[args], h] threads f over any objects with head h that appear in args.
Thread[f[args], h, n] threads f over objects with head h that appear in the first n args.
Thread[f[args], h, -n] threads over the last n args.
Thread[f[args], h, {m, n}] threads over arguments m through n.

Modelli

s:obj represents the pattern object obj, assigned the name s.

p?test is a pattern object that stands for any expression which matches p, and on which the application of test gives True.

p.. is a pattern object which represents a sequence of one or more expressions, each matching p.

p... is a pattern object which represents a sequence of zero or more expressions, each matching p.

p /; test is a pattern which matches only if the evaluation of test yields True.

p1 | p2 | ... is a pattern object which represents any of the patterns pi.

_ or Blank[ ] is a pattern object that can stand for any Mathematica expression.
_h or Blank[h] can stand for any expression with head h.

___ (three _ characters) or BlankNullSequence[ ] is a pattern object that can stand for any sequence of zero or more Mathematica expressions.
___h or BlankNullSequence[h] can stand for any sequence of expressions, all of which have head h.

__ (two _ characters) or BlankSequence[ ] is a pattern object that can stand for any sequence of one or more Mathematica expressions.
__h or BlankSequence[h] can stand for any sequence of one or more expressions, all of which have head h.

represents the first argument supplied to a pure function.
#n represents the nth argument.

represents the sequence of arguments supplied to a pure function.
##n represents the sequence of arguments supplied to a pure function, starting with the nth argument.

MatchQ[expr, form] returns True if the pattern form matches expr, and returns False otherwise.

Strutture

expr1; expr2; ... evaluates the expri in turn, giving the last one as the result.

Do[expr, {imax}] evaluates expr imax times. Do[expr, {i, imax}] evaluates expr with the variable i successively taking on the values 1 through imax (in steps of 1).
Do[expr, {i, imin, imax}] starts with i = imin.
Do[expr, {i, imin, imax, di}] uses steps di.
Do[expr, {i, imin, imax}, {j, jmin, jmax}, ...] evaluates expr looping over different values of j, etc. for each i.

For[start, test, incr, body] executes start, then repeatedly evaluates body and incr until test fails to give True.

While[test, body] evaluates test, then body, repetitively, until test first fails to give True.

Block[{x, y, ...}, expr] specifies that expr is to be evaluated with local values for the symbols x, y, ...
Block[{x = x0, ...}, expr] defines initial local values for x, ...
Block[{vars}, body /; cond] allows local variables to be shared between conditions and function bodies.

Module[{x, y, ...}, expr] specifies that occurrences of the symbols x, y, ... in expr should be treated as local.
Module[{x = x0, ...}, expr] defines initial values for x, ...

With[{x = x0, y = y0, ...}, expr] specifies that in expr occurrences of the symbols x, y, ... should be replaced by x0, y0, ...

If[condition, t, f] gives t if condition evaluates to True, and f if it evaluates to False.
If[condition, t, f, u] gives u if condition evaluates to neither True nor False.

Switch[expr, form1, value1, form2, value2, ...] evaluates expr, then compares it with each of the formi in turn, evaluating and returning the valuei corresponding to the first match found.

Which[test1, value1, test2, value2, ...] evaluates each of the testi in turn, returning the value of the valuei corresponding to the first one that yields True.

Valutazione

Evaluate[expr] causes expr to be evaluated, even if it appears as the argument of a function whose attributes specify that it should be held unevaluated.

Hold[expr] maintains expr in an unevaluated form.

ReleaseHold[expr] removes Hold and HoldForm in expr.

TimeConstrained[expr, t] evaluates expr, stopping after t seconds.
TimeConstrained[expr, t, failexpr] returns failexpr if the time constraint is not met.

MemoryConstrained[expr, b] evaluates expr, stopping if more than b bytes of memory are requested.
MemoryConstrained[expr, b, failexpr] returns failexpr if the memory constraint is not met.

Check[expr, failexpr] evaluates expr, and returns the result, unless messages were generated, in which case it evaluates and returns failexpr.
Check[expr, failexpr, s1::t1, s2::t2, ...] checks only for the specified messages.

Messaggi

symbol::tag (or MessageName[symbol, "tag"]) is a name for a message.

Messages[symbol] gives all the messages assigned to a particular symbol.

On[symbol::tag] switches on a message, so that it can be printed.
On[s] switches on tracing for the symbol s.
On[m1, m2, ...] switches on several messages.
On[ ] switches on tracing for all symbols.

Off[symbol::tag] switches off a message, so that it is no longer printed.
Off[s] switches off tracing messages associated with the symbol s.
Off[m1, m2, ...] switches off several messages.
Off[ ] switches off all tracing messages.

Message[symbol::tag] prints the message symbol::tag unless it has been switched off.
Message[symbol::tag, e1, e2, ...] prints a message, inserting the values of the ei as needed.

Debugging

Trace[expr] generates a list of all expressions used in the evaluation of expr.
Trace[expr, form] includes only those expressions which match form.
Trace[expr, s] includes all evaluations which use transformation rules associated with the symbol s.

TracePrint[expr] prints all expressions used in the evaluation of expr.
TracePrint[expr, form] includes only those expressions which match form.
TracePrint[expr, s] includes all evaluations which use transformation rules associated with the symbol s.

TraceScan[f, expr] applies f to all expressions used in the evaluation of expr.
TraceScan[f, expr, form] includes only those expressions which match form.
TraceScan[f, expr, s] includes all evaluations which use transformation rules associated with the symbol s.
TraceScan[f, expr, form, fp] applies f before evaluation and fp after evaluation to expressions used in the evaluation of expr.