The "Unknown:"s below indicate that an entry is incomplete.

- either the entry exist in the language, and please tell.
- either the entry doesn't exist in the language, and please tell so. The entry will be marked as such and won't appear as missing anymore.

- Category: Dynamically typed
- Various

`%`	commenting (until end of line)
`@< / @=< / @> / @>=`	comparison
`min / max`	comparison (min / max (binary or more))
`compare`	comparison (returns 3 values (i.e. inferior, equal or superior))
`== \==`	equality / inequality (deep)
`=@= \=@= / = \= / =:= =\=`(1)	equality / inequality (deep)
`garbage_collect`	force garbage collection
`( ... )`	grouping expressions
`=..`(2)	runtime evaluation
`case-sensitive`	tokens (case-sensitivity (keywords, variable identifiers...))
`[_a-z][_a-zA-Z0-9]*`	tokens (function identifier regexp (if different from variable identifier regexp))
`[_A-Z][_a-zA-Z0-9]*`	tokens (variable identifier regexp)
`CamelCase for variables, underscores for predicates`	tokens (what is the standard way for scrunching together multiple words)
`is`	variable assignment or declaration (assignment)
`=`	variable assignment or declaration (declaration)
`:-`	variable assignment or declaration (declaration)

- Functions

`f(a,b,...)`	function call
`.. [ f, A, B, ...]`	function call
`f`	function call (with no parameter)
`the predicate fail`	function called when a function is not defined (in dynamic languages)

`f(Para1, Para2, ....) :- ... .`	function definition (predicates)
`current_predicate`	runtime inspecting the caller information

- Control Flow

`catch`	exception (catching)
`throw`	exception (throwing)
`c -> b1 ; c2 -> b2 ; b3`	if_then_else
`repeat, ..., c`	loop (do something until condition)

`X = val,`
`(X = v1, ... ; X = v2, ... ; ...)`
	multiple selection (switch)

`,`	sequence

- Package, Module

`:- module(p)`	declare
`:- use_module(name1, name2, ...)`	import (selectively)

Unknown:

package scope

- Strings

`char_code`	ascii to character
`char_code`	character to ascii
`sub_string / sub_atom`	extract a substring
`sub_string / sub_atom`	locate a substring
`write`	simple print (on any objects)
`format`(3)	simple print (printf-like)
`append`	string concatenation
`= \=`	string equality & inequality
`length`	string size
`atom_length`	string size
`'...'`	strings (with no interpolation of variables)
`"..."`	strings (with no interpolation of variables)

`char_type(C_, to_upper(C)), char_type(C_, to_lower(C))`	upper / lower case character
`upcase_atom/downcase_atom`	uppercase / lowercase / capitalized string
`upcase_atom / downcase_atom`	uppercase / lowercase / capitalized string

Unknown:

strings (end-of-line (without writing the real CR or LF character))
sprintf-like
accessing n-th character

- Booleans

`No`	false value
`fail`	false value
`not`(4)	logical not
`; / ,`	logical or / and (short circuit)
`true`	true value
`Yes`	true value

- Bags and Lists

`[ e | l ]`	adding an element at the beginning (list cons) (return the new list (no side-effect))
`L = [_|ButFirst]`	all but the first element
`forall`	for each element do something
`member`	is an element in the list
`concat_atom`	join a list of strings in a string using a glue string
`last`	last element
`append`	list concatenation
`[ a, b, c ]`(5)	list constructor
`flatten`	list flattening (one level depth)
`length`	list size
`nth0 / nth1`	list/array indexing
`get_assoc`	lookup an element in a association list
`reverse`	reverse
`min / max`	smallest / biggest element

`sort`(6)	sort
`predsort / keysort / mergesort`	sort
`maplist`	transform a list (or bag) in another one
`sublist`	transform a list (or bag) in another one
`maplist2`	transform two lists in parallel

Unknown:

first element
get the first element and remove it
get the last element and remove it
remove duplicates

- Various Data Types

`Nothing`	computable tuple (these are a kind of immutable lists playing a special role in parameter passing) (empty tuple)
`L =.. [ F | Args ], call(L)`	computable tuple (these are a kind of immutable lists playing a special role in parameter passing) (using a tuple for a function call)
`findall(Key, item(Key, _), Keys)`	dictionary (list of keys)
`numlist / between`	range (inclusive .. inclusive)
`( a, b, c )`	tuple constructor

Unknown:

computable tuple (these are a kind of immutable lists playing a special role in parameter passing) (1-uple)
reference (pointer) (creation)
reference (pointer) (dereference)

- Mathematics

`+ / - / * / /`	addition / subtraction / multiplication / division
`/\ / \/ / xor`	bitwise operators (and / or / xor)
`\`	bitwise operators (bitwise inversion)
`<< / >>`	bitwise operators (left shift / right shift / unsigned right shift)
`**`	exponentiation (power)
`log10`	logarithm (base 10)
`log`	logarithm (base e)

`mod`	modulo (modulo of -3 / 2 is -1)
`mod`	modulo (modulo of -3 / 2 is 1)
`rem`	modulo (modulo of -3 / 2 is 1)
`-`	negation
`1000.0, 1E3`	numbers syntax (floating point)
`1000`	numbers syntax (integers)
`random`	random (random number)
`sqrt / exp / abs`	square root / e-exponential / absolute value
`sin / cos / tan`	trigonometry (basic)
`asin / acos / atan`(7)	trigonometry (inverse)
`truncate / round / floor / ceiling`	truncate / round / floor / ceil

Unknown:

operator priorities and associativities

Remarks

- (1) normal / structural / unification / arithmetic
- (2) Univ operator
- (3) but not using the C-like %-syntax
- (4) Smalltalk: postfix operator
- (5) new in PHP 5.4
- (6) in Scheme, not standard, but nearly standard
- (7) Ruby >= 1.7

Pixel