Punctuation

Extended documentation for mathematical symbols & functions is here.

symbol	meaning
`@m`	the at-symbol invokes macro `m`; followed by space-separated expressions or a function-call-like argument list
`!`	an exclamation mark is a prefix operator for logical negation ("not")
`a!`	function names that end with an exclamation mark modify one or more of their arguments by convention
`#`	the number sign (or hash or pound) character begins single line comments
`#=`	when followed by an equals sign, it begins a multi-line comment (these are nestable)
`=#`	end a multi-line comment by immediately preceding the number sign with an equals sign
`$`	the dollar sign is used for string and expression interpolation
`%`	the percent symbol is the remainder operator
`^`	the caret is the exponentiation operator.
`&`	single ampersand is bitwise and
`&&`	double ampersands is short-circuiting boolean and
`|`	single pipe character is bitwise or
`||`	double pipe characters is short-circuiting boolean or
`⊻`	the unicode xor character is bitwise exclusive or
`~`	the tilde is an operator for bitwise not
`'`	a trailing apostrophe is the `adjoint` (that is, the complex transpose) operator A^H

`*`	the asterisk is used for multiplication, including matrix multiplication and string concatenation
`/`	forward slash divides the argument on its left by the one on its right
`\`	backslash operator divides the argument on its right by the one on its left, commonly used to solve matrix equations
`()`	parentheses with no arguments constructs an empty `Tuple`
`(a,...)`	parentheses with comma-separated arguments constructs a tuple containing its arguments
`(a=1,...)`	parentheses with comma-separated assignments constructs a `NamedTuple`
`(x;y)`	parentheses can also be used to group one or more semicolon separated expressions
`a[]`	array indexing (calling `getindex` or `setindex!`)
`[,]`	vector literal constructor (calling `vect`)
`[;]`	vertical concatenation (calling `vcat` or `hvcat`)
`[ ]`	with space-separated expressions, horizontal concatenation (calling `hcat` or `hvcat`)
`T{ }`	curly braces following a type list that type's parameters
`{}`	curly braces can also be used to group multiple `where` expressions in function declarations
`;`	semicolons separate statements, begin a list of keyword arguments in function declarations or calls, or are used to separate array literals for vertical concatenation
`,`	commas separate function arguments or tuple or array components
`?`	the question mark delimits the ternary conditional operator (used like: `conditional ? if_true : if_false`)
`" "`	the single double-quote character delimits `String` literals
`""" """`	three double-quote characters delimits string literals that may contain `"` and ignore leading indentation
`' '`	the single-quote character delimits `Char` (that is, character) literals

`` ` ` ``	the backtick character delimits external process (`Cmd`) literals
`A...`	triple periods are a postfix operator that "splat" their arguments' contents into many arguments of a function call or declare a varargs function that "slurps" up many arguments into a single tuple
`a.b`	single periods access named fields in objects/modules (calling `getproperty` or `setproperty!`)
`f.()`	periods may also prefix parentheses (like `f.(...)`) or infix operators (like `.+`) to perform the function element-wise (calling `broadcast`)
`a:b`	colons (`:`) used as a binary infix operator construct a range from `a` to `b` (inclusive) with fixed step size `1`
`a:s:b`	colons (`:`) used as a ternary infix operator construct a range from `a` to `b` (inclusive) with step size `s`
`:`	when used by themselves, `Colon`s represent all indices within a dimension, frequently combined with indexing
`::`	double-colons represent a type annotation or `typeassert`, depending on context, frequently used when declaring function arguments
`:( )`	quoted expression
`:a`	`Symbol` a
`<:`	subtype operator
`>:`	supertype operator (reverse of subtype operator)
`=`	single equals sign is assignment
`==`	double equals sign is value equality comparison
`===`	triple equals sign is programmatically identical equality comparison.

« I/O and Network Sorting and Related Functions »