

Oz Programming: Basic syntax cheatsheets

This document is a non-exhaustive reminder of the syntax of the Oz programming language. It is always possible to improve it and your help is therefore welcome – just submit an issue on the link below and we will modify the document. Source code and the latest version of the pdf can be found at the following address:

<https://github.com/some-github/a-wonderful-link>

Keywords	Meaning
Basic statements	
<code>Var = ...</code>	variable assignment
<code>declare Var</code>	global declaration of Var
<code>local Var in</code> ... <code>end</code>	local declaration
<code>fun {FunName Arg1 ... ArgN}</code> ... <code>end</code>	function definition
<code>proc {ProcName Arg1 ... ArgN}</code> ... <code>end</code>	procedure definition
<code>if Condition1 then ...</code> <code>elseif Condition2 then ...</code> <code>else ...</code> <code>end</code>	if ...else if ...else ...
<code>case Var of Pattern_1 then ...</code> <code>[] Pattern_2 then ...</code> <code>else ...</code> <code>end</code>	pattern matching
Booleans expressions and operators	
<code>false</code>	false value
<code>true</code>	true value
<code>andthen</code>	logical and

<code>orelse</code>	logical or
<code>==</code>	logical equality
<code>\=</code>	logical inequality
<code>{Not [Your Expression]}</code>	logical not

Comparison operators

<code><</code>	less than
<code>=<</code>	less than or equal to
<code>></code>	greater than
<code>>=</code>	greater than or equal to

Arithmetic operators

<code>+</code>	addition
<code>-</code>	subtraction
<code>*</code>	multiplication
<code>/</code>	division (for floating point numbers)
<code>div</code>	division (for integers)
<code>mod</code>	modulo
<code>{Pow A B}</code>	A^B
<code>{Abs A}</code>	absolute value of A
<code>E = ~1</code>	unary negation

Data structures

<code>S = "A string"</code>	string declaration
<code>A = hELLO</code>	atom declaration (with lowercase first letter)
<code>A = 'An atom'</code>	same (with uppercase first letter and space)
<code>X = label(feature1:Field1 ... featureN:FieldN)</code>	record structure
<code>R.feature</code>	access to the record's fields
<code>T = 1#2#3</code>	common operator (<code>T = '#'(1:1 2:2 3:3)</code>)
<code>L = ' '(1:1 2:' '(1:2 2:nil))</code>	list structure

<code>L = 1 2 nil</code>	a syntactic sugar to declare a list
<code>L = [1 2]</code>	another syntactic sugar for list declaration
<code>X = {NewCell Y}</code>	cell creation (multiple assignment variable)
<code>@X</code>	access to the cell's current content
<code>X := Z</code>	changes the content of the cell

Object-oriented programming

<pre>class AClass attr a1 ... an meth init(Arg) ... end meth m1 ... end ... meth mn(Arg) ... end end</pre>	class definition
<pre>X = {New AClass init('arg')} {X m1}</pre>	object creation and use

Exceptions handling

<code>raise E end</code>	throws an exception E
<code>try ... catch X then ... end</code>	catches a raised exception

Concurrent programming

<code>thread ... end</code>	thread creation
-----------------------------	-----------------
