# 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. Source code and the latest version of the pdf can be found at the following address:

`https://github.com/el-nounou/Oz-syntax-cheatsheet`

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

| | |
|---|---|
| `==` | logical equality |
| `\=` | logical inequality |
| `{Not [Your Expression]}` | logical not |

## Comparison operators

| | |
|---|---|
| `<` | less than |
| `=<` | less than or equal to |
| `>` | greater than |
| `>=` | greater than or equal to |

## Arithmetic operators

| | |
|---|---|
| `+` | addition |
| `-` | subtraction |
| `*` | multiplication |
| `/` | division (for floating point numbers) |
| `div` | division (for integers) |
| `mod` | modulo |
| `{Pow A B}` | $A^B$ |
| `{Abs A}` | absolute value of A |
| `~` | unary negation |

## Data structures

| | |
|---|---|
| `S = "A string"` | string declaration |
| `A = hELLO` | atom declaration |
| `A = 'An atom'` | same (with uppercase first letter and space) |
| `X = label(feature1:Field1 ... featureN:FieldN)` | record structure |
| `R.feature` | access to the record's fields |
| `T = 1#2#3` | common operator (T = '#'(1:1 2:2 3:3)) |
| `L = '\|'(1 '\|'(2 nil))` | list structure |
| `L = 1\|2\|nil` | another way to declare a list |
| `L = [1 2]` | syntactic sugar for list declaration |
| `X = {NewCell Y}` | cell creation (multiple assignment variable) |
| `@X` | access to the cell's current content |
| `X := Z` | changes the content of the cell |

## Object-oriented programming

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

**Exceptions handling**

| | |
|---|---|
| `raise E end` | throws an exception E |
| `try ... catch X then ... end` | catches a raised exception |

**Concurrent programming**

| | |
|---|---|
| `thread ... end` | thread creation |