CoCoA 4 Reference Card

All command names and keywords begin with a capital letter. Every command **must end with a semicolon** (;) and may extend over several lines.

Quitting CoCoA

- `Quit;` or `Ciao;` will quit the CoCoA session

See also the section **Help! CoCoA is not responding**.

Online help

CoCoA has on-line help for all commands and functions.

- ?*key-phrase* prints the manual entry associated with *key-phrase*, if several entries are relevant their titles are printed.
- ??*key-phrase* prints out the titles of all manual entries associated with *key-phrase*.
- `?tutorial` will start the online CoCoA tutorial.

Ring Environments

Before computing with polynomials you must specify the ring to use via the command `Use` or `Using`. The special operator `::=` stores a ring in a variable.

- `Q[x,y,z];` — coeffs in $\mathbb{Q}$, indets `x,y,z`
- `Z/(23)[x,y,z];` — coeffs in $\mathbb{Z}/23\mathbb{Z}$
- `Q[x,y,z],Lex;` — lexicographic term ordering
- `Q[x[1..7],y[0..3]];` — subscripted indets
- `S ::= Q[x[1..8]];` — store ring in variable `S`
- `Use R;` — use ring stored in `R`
- `Use R ::= Q[x,y];` — store ring in `R` and use it
- `Use Q[x,y,z];` — use a newly created ring
- `Using R Do` *commands* `EndUsing;`
  — use ring stored in `R` for *commands*
- `CurrentRing()` returns the ring currently in use

Ring Indeterminates

The name of an indeterminate is one lowercase letter possibly indexed by integers: some examples are `x`, `t[1]`, `m[2,3]` and `m[I,J+1]`. An index may be any expression having an integer value; the range of valid indices is given when the ring is created, *e.g.* `Q[x[1..6,1..6]]`.
Juxtaposition of indeterminates represents product: `xy` is simply shorthand for `x*y`, and `xyzzy` is short for `x*y^2*z^2`.

Storage Variables and arithmetic

Variables are used for storing results which will be needed again later. The name of a variable must start with a capital letter, and may be followed by further letters and/or digits: *e.g.* `X`, `X2`, `Gamma`, and `TotalDistance`. To store a value in a variable use the assignment operator `:=`

```
NumPoints:=12;
MinimalPoly:=x^6+2*x^5+3*x^4+4*x^3+5*x^2+6*x+7;
```

The arithmetic operators are: `+ - * / ^` being respectively sum, difference, product, quotient, and power.

WARNING `1/2*x` is the same as `(1/2)*x`, to put `x` in the denominator you must write `1/(2*x)` or `1/2/x`. Also a negative exponent must be in parentheses `10^(-2)` otherwise you get a `parse error`.

Lists

A list is created by writing several values between square brackets. Use indexing to get at the entries in a list.

`Empty:=[];` — store the empty list

`Primes:=[2,3,5,7,11];` — store a list of five numbers

`Primes[3];` — returns 3rd element of the list `Primes`

`1..10` — returns the list of integers from 1 to 10

`Concat([1,2],[3,4]);` — returns the concatenation

`Append(Primes, 13);` — changes value of `Primes`

`Len(Primes);` — returns the number of elements in the list

You can apply a function to some or all the elements of a list:
`[X^2 | X In [1,2,3]]` — produces `[1,4,9]`
`[X^2 | X In [1,2,3] And X>2]` — produces `[9]`

`[` *expr* `|` *var* `In` *list* `]` creates a list of values of *expr* as *var* runs through *list*.

`[` *expr* `|` *var* `In` *list* `And` *cond* `]` creates a list of values of *expr* whenever *cond* is true as *var* runs through *list*.

See also the programming construct `Foreach`. Try typing `??list` in a CoCoA session for more information.

Matrices

A list of lists can be converted into a matrix using `Mat`:
`Mat([[1,2],[0,1]])` — make matrix from list of rows

`Det(M)` — returns determinant of the matrix `M`

`Transposed(M)` — have a guess what this does!

`M^(-1)` — returns inverse of the matrix `M`

`M[I,J]` — returns entry (`I`,`J`) of matrix `M`

To solve a linear system see the online manual for `LinSol` and `LinKer`. Try typing `??mat` in a CoCoA session for more information about operations on matrices.

Ideals, Modules, and Gröbner Bases

To compute a G-basis you must first create an ideal/module:

`Ideal(x,y,z^2)` — ideal generated by x, y, z^2

`Ideal(L)` — ideal generated by elements of list `L`

`Module([x,y],[z,t])` — module gen'd by given vectors

`Module(M)` — module gen'd by the rows of matrix `M`

`GBasis(A)` — G-basis of ideal/module `A`

`ReducedGBasis(A)` — reduced G-basis of ideal/module `A`

The ordering used for the Gröbner basis is that associated with the ring containing the generators — it was specified when the ring was created.

Moving values between rings

- `BringIn`
- `Image`

Printing

CoCoA automatically prints out the result of an arithmetic expression whose value is not assigned to a variable. Finer control of printing may be achieved using the commands `PrintLn` and `Print` — for details consult the online manual via `?print`
To output results to a file, there is a variant of the printing commands: consult the online manual with `?print ON`
The command `Set Indentation;` tells CoCoA to print lists out "vertically" rather than "horizontally".

Programming Constructs

Here are the syntaxes of the more common constructs: keywords are in `typewriter`, *italics* describe the class of expression to appear in that position.

- `If` *cond* `Then` *commands* `EndIf;`
- `If` *cond* `Then` *commands* `Else` *commands* `EndIf;`
- `While` *cond* `Do` *commands* `EndWhile;`
- `For` *var* `:=` *lower* `To` *upper* `Do` *commands* `EndFor;`
  lower and *upper* must evaluate to integers, and if *lower* > *upper* then *commands* are not executed
- `Foreach` *var* `In` *list* `Do` *commands* `EndForeach;`
- `Define` *fn-name* `(` *args* `)` *commands* `EndDefine;`
  Give a definition of a new function/procedure. *fn-name* comprises letters and digits, and must begin with a capital
- `Return;` — return from a procedure
- `Return` *expr*`;` — return a value from a function

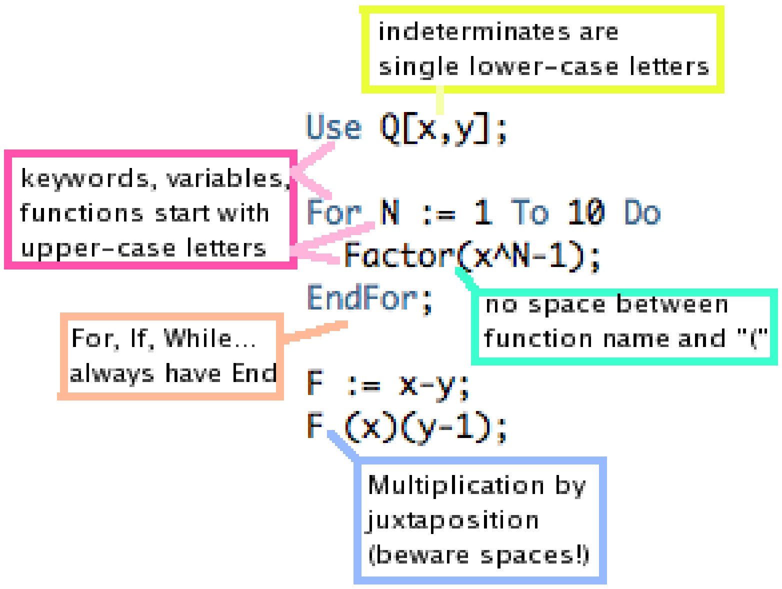
Understanding errors

If you get lots of error messages in succession, concentrate just on the first one or two, and ignore the rest. The most common mistakes are omitting the semicolon at the end of a command, or not making the first letter of a keyword of variable capital. Sometimes the real error is a line or two before where CoCoA said it had trouble.
An error message saying `Undefined indeterminate` probably means you forgot a capital letter at the start of a keyword, variable or function name.
An error message saying `Undefined variable` *XYZ*, where *XYZ* is a function probably means you put a space between *XYZ* and the following bracket. There should be no space. If you do leave a space (*e.g.* `F (N-1)`) then the expression is interpreted as the product of the variable `F` by the value `(N-1)`. If you meant the product, write explicitly `F*(N-1)`.

Help! CoCoA is not responding

There are several reasons why CoCoA might seem to ignore commands. If this happens, try to evaluate a simple command (*e.g.* `1+1;`), to see if you get an answer. If not, read on.

- You may have omitted to type a semicolon (`;`) at the end of the previous command. CoCoA will obey the command only when it sees the semicolon. So just enter a semicolon as the next line.
- CoCoA might still be working on some previous lengthy computation. New commands will be handled only when the previous computations have been completed. To interrupt (*i.e.* stop and discard) a lengthy computation: on Unix/Linux/MacOS X type `control-C`, on Microsoft try clicking randomly on icons.
- You might have forgotten an `EndIf;` or other `End`-word, so try typing a hash (`#`). This will probably produce a `parse error` which you should ignore, and will cancel the last command. If you do not get a `parse error`, try the following suggestion.
- You might have forgotten to end a string. A string in CoCoA may contain "newline" characters, so if you forget to close it then subsequent lines will be swallowed up into the string giving the impression that CoCoA is no longer responding. To get out of this, terminate the string by typing a single or double quote (depending on which you used to start it) and then type a hash (`#`).


indeterminates are single lower-case letters
keywords, variables, functions start with upper-case letters
no space between function name and "("
For, If, While... always have End
Multiplication by juxtaposition (beware spaces!)

```
Use Q[x,y];
For N := 1 To 10 Do
  Factor(x^N-1);
EndFor;
F := x-y;
F (x)(y-1);
```