```
-----------------------------------------------------------------------------
-- Groebner bases at ease
use QQ[x,y,z];
I := ideal(x^3 +x*y^2 -2*z,  x^2*y^3 -y*z^2);
GBasis(I);    // same as  "print GBasis(I);"

-- printing facilities
indent(GBasis(I));
-- [
--   x^3 +x*y^2 -2*z,
--   x^2*y^3 -y*z^2,
--   x*y^5 -2*y^3*z +x*y*z^2,
--   x*y^3*z +(-1/2)*x^2*y*z^2 +(-1/2)*y^3*z^2,
--   y^5*z^2 -4*y^3*z^2 +2*x*y*z^3 +y*z^4
-- ]
latex(GBasis(I));
-- [ \ x^3  +x y^2  -2 z,
-- x^2 y^3  -y z^2 ,
-- x y^5  -2 y^3 z +x y z^2 ,
-- x y^3 z -\frac{1}{2} x^2 y z^2  -\frac{1}{2} y^3 z^2 ,
-- y^5 z^2  -4 y^3 z^2  +2 x y z^3  +y z^4 \ ]

-- not recomputing the GBasis (ideal is "mutable")
HasGBasis(I);
-- true

-----------------------------------------------------------------------------
-- GBasis working on algebraic extensions
use R ::= QQ[i];
QQi := R/ideal(i^2 +1);
--  or directly   QQi := NewQuotientRing(NewPolyRing(QQ, "i"), "i^2+1");

use QQi[x,y,z];
I := ideal(x^3 +x*y^2 -2*i*z,  x^2*y^3 -i*y*z^2);

indent(GBasis(I));
-- [
--   x^3 +x*y^2 +(-2*i)*z,
--   x^2*y^3 +(-i)*y*z^2,
--   x*y^5 +(-2*i)*y^3*z +(i)*x*y*z^2,
--   x*y^3*z +(-1/2)*x^2*y*z^2 +(-1/2)*y^3*z^2,
--   y^5*z^2 +(-4*i)*y^3*z^2 +(2*i)*x*y*z^3 +(i)*y*z^4
-- ]

-----------------------------------------------------------------------------
-- non-commutative GBasis (needs refinement and feedback...)
WA := NewWeylAlgebra(QQ, ["x", "y"]);
use WA;
indets(WA); --> x, y, dx, dy
x*dx;
dx*x; --> x*dx +1
ReducedGBasis(ideal(x, dx)); --> [1]


-----------------------------------------------------------------------------
-- Some of CoCoA's *SPECIALITIES*
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
```

```
-- "TwinFloat numbers and the computation of gin"

RR64 := NewRingTwinFloat(64);
use RR64_X ::= RR64[x,y,z];
F := (1/3)*x - 1/100000;  F;
-- "safe" reconstruction over QQ, or not:
10^20 * F; --> (100000000000000000000/3)*x -1000000000000000
10^40 * F; --> 0.3333333333333333333333333333333333333333*10^40*x -...

I := ideal(x^3 +x*y^2 -2*z,  x^2*y^3 -y*z^2);  -- a small example
GBasis(I); ---> reconstructed in QQ: same result as computation over QQ
-- [x^3 +x*y^2 -2*z,  x^2*y^3 -y*z^2,  x*y^5 -2*y^3*z +x*y*z^2,
-- x*y^3*z -1/2*x^2*y*z^2 -1/2*y^3*z^2,  y^5*z^2 -4*y^3*z^2 +2*x*y*z^3 +y*z^4]

-- TwinFloats are very useful for computing "gin"
-- gin(I) is LT(g(I)) where g is a generic/random change of coordinates
use QQ_X ::= QQ[x,y,z];
I := ideal(y^20 -x^5*z^6,  x^7*z^3 -y*z^2);

L := [ sum([ random(-1000,1000)*indet(QQ_X,j) | j in 1..3]) | i in 1..3];
indent(L);
-- [
--   897*x +453*y -634*z,
--   -421*x -947*y -50*z,
--   583*x -725*y -19*z
-- ]

g := PolyAlgebraHom(QQ_X, QQ_X, L); -- QQ[x,y,z] --> QQ[x,y,z] x -> L[1], ...
gI := ideal(g(gens(I)));   gI;
t0 := CpuTime();   GB := GBasis(gI);   TimeFrom(t0);
-- SAFE INTERRUPT: memory state as before the call

-- map GI into RR32_X:
RR32_X := NewPolyRing(NewRingTwinFloat(32), "x,y,z");
phi := PolyRingHom(QQ_X, RR32_X, CanonicalHom(QQ,RR32_X), indets(RR32_X));
gI32 := ideal(phi(gens(gI)));
GB := GBasis(gI32); --> detects insufficient precision
--> ERROR: RingTwinFloat: insufficient precision (...)

-- we try with higher precision:
phi := PolyRingHom(QQ_X, RR64_X, CanonicalHom(QQ,RR64_X), indets(RR64_X));
gI64 := ideal(phi(gens(gI)));
GB := GBasis(gI64);  -- it did it!
LT(gI64);

GB[1]; -->  is reconstructed to QQ
substring(sprint(GB[3]), 1,70);  -->  is not reconstructed to QQ
--> cannot be safely reconstructed to QQ, but we just need its LT!

-- CoCoA's "gin" does this all TwinFloats creation/mapping for you, and more:
-- random coeffs between -10^6 and 10^6;  computed twice, just to make sure!
SetVerbosityLevel(90);
gin(I); --> ideal(x^5, x^4*y^16, x^3*y^18, x^2*y^20, x*y^22, y^24)

SetVerbosityLevel(0);

---------------------------------------------------------------------------
-- GROEBNER BASES of ideals of points
-- (Abbott-Bigatti-Kreuzer-Robbiano 1999)
```

```
P ::= QQ[x,y];
points := mat([[10, 0], [-10, 0], [0, 10], [0, -10],
               [7, 7], [-7, -7], [7, -7], [-7, 7]]);
IdealOfPoints(P, points);
-- ideal(x^2*y +(49/51)*y^3 +(-4900/51)*y,  x^3 +(51/49)*x*y^2 -100*x,
--       y^4 +(-2499/2)*x^2 +(-2699/2)*y^2 +124950,  x*y^3 -49*x*y)


-- BORDER BASES of ideals of (approximate) points
-- (by Abbott-Fassino-Torrente)
epsilon := mat([[0.1, 0.1]]);
ApproxPointsNBM(P, points, epsilon);  -- Numerical Buchberger Algorithm
indent(It, 2);
--   AlmostVanishing := [
--     x^2 +(4999/5001)*y^2 -165000/1667, ...
--> indeed the points are "almost" on the circle centered in (0,0) radius 10

epsilon := mat([[0.01, 0.01]]);
indent(ApproxPointsNBM(P, points, epsilon).AlmostVanishing); --> degree > 3
--> not on a conic if the desired approximation is ~0.01



-----------------------------------------------------------------------------
-- Another kind of approximation:  (Abbott 2017)
-- modular computations and fault-tolerant rational reconstruction

RingElem(NewPolyRing(NewZZmod(32003),"x,y"), "(10/3)*x^2␣+(-1/4)*y");
-- 10671*x^2 -8001*y
RingElem(NewPolyRing(NewZZmod(31991),"x,y"), "(10/3)*x^2␣+(-1/4)*y");
-- 10667*x^2 -7998*y
RingElem(NewPolyRing(NewZZmod(32009),"x,y"), "(10/3)*x^2␣+(-1/4)*y");
-- 10673*x^2 +8002*y

CRTPoly(10671*x^2 -8001*y, 32003,     10667*x^2 -7998*y, 31991);
CRTPoly(It.residue, It.modulus,     10673*x^2 +8002*y, 32009);
-- record[modulus := 32771069407757, residue := 10923689802589*x^2 +8192767351939*y]

RatReconstructPoly(It.residue, It.modulus);  -- fault-tolerant
-- (10/3)*x^2 +(-1/4)*y

-----------------------------------------------------------------------------
-- 0-dimensional ideals:  (Abbott-Bigatti-Palezzato-Robbiano 2020)
-- Lots of hidden features

use P ::= QQ[x,y,z];
I := ideal((z^7 -z -1)^2,  (y*z -x^2)^3,  x^9-x-1);
multiplicity(P/I);

SetVerbosityLevel(90);
radI := radical(I);
-- modular arithmetic
-- GBasis with time limit
-- probabilistic algorithm with actual verification
```