```
# GAP SCSCP server for the example of calling Singular from Python SCSCP client

LogTo(); # to close the log file in case it was opened earlier
LoadPackage("singular");
LoadPackage("scscp");

# create polynomial from its external representation
AssemblePolynomial := function( extrep )
local fam, rep, coeffs, mons, i, term, j, p;
fam := RationalFunctionsFamily(FamilyObj(1));
rep := [ ];
coeffs := extrep[1];
mons   := extrep[2];
for i in [1..Length(coeffs)] do
  term:=[];
  for j in [1..Length(mons[i])] do
    if mons[i][j]>0 then
      Append(term,[j,mons[i][j]]);
    fi;
  od;
  Append( rep, [ term, coeffs[i] ] );
od;
p:=PolynomialByExtRep(fam,rep);
return p;
end;

# produce external representation of a polynomial
DisassemblePolynomial:=function(f)
local rep, coeffs, mons, deg, t, r, i, term, mon, j;
rep := ExtRepPolynomialRatFun(f);
coeffs := [];
mons   := [];
deg    := Maximum( List( Filtered(rep{[1,3..Length(rep)-1]}, t -> Length(t)>0),
                         r -> Maximum(r{[1,3..Length(r)-1]}) ) );
for i in [1,3..Length(rep)-1] do
  term := rep[i];
  mon := ListWithIdenticalEntries(deg,0);
    for j in [1,3..Length(term)-1] do
      mon[term[j]]:=term[j+1];
    od;
  Add( mons, mon );
  Add( coeffs, rep[i+1]);
od;
return [coeffs,mons];
end;

# This is the main purpose of this server
GroebnerBasisWithSingular:=function( extreps )
# it accepts external representations of polynomials
local R, r, I, B;
# create polynomial ring of appropriate rank
R:=PolynomialRing( Rationals, Maximum(List( extreps, r -> Length(r[2]) ) ) );
# convert arguments to polynomials and get an ideal they generate
I:=Ideal( R, List( extreps, AssemblePolynomial ) );
# call local instance of Singular
B:=GroebnerBasis(I);
# return result in the form of external representations
return List(B,DisassemblePolynomial);
end;

# Procedures that the GAP SCSCP server provides

# Useful for simple tests
InstallSCSCPprocedure( "Identity", x -> x,
  "Identity procedure for tests", 1, 1 );
```

```
# Clearly, f = AssemblePolynomial( DisassemblePolynomial( f ) )
PingPongPoly := x -> DisassemblePolynomial( AssemblePolynomial ( x ) );
InstallSCSCPprocedure( "PingPongPoly", PingPongPoly,
  "Decode/encode polynomial and send it back", 1, 1 );

# Setting up calculation and calling Singular
InstallSCSCPprocedure( "GroebnerBasisWithSingular", GroebnerBasisWithSingular,
  "Groebner Basis with Singular", 1, 1 );

# Start GAP SCSCP server
RunSCSCPserver( SCSCPserverAddress, SCSCPserverPort : OMignoreMatrices);
```