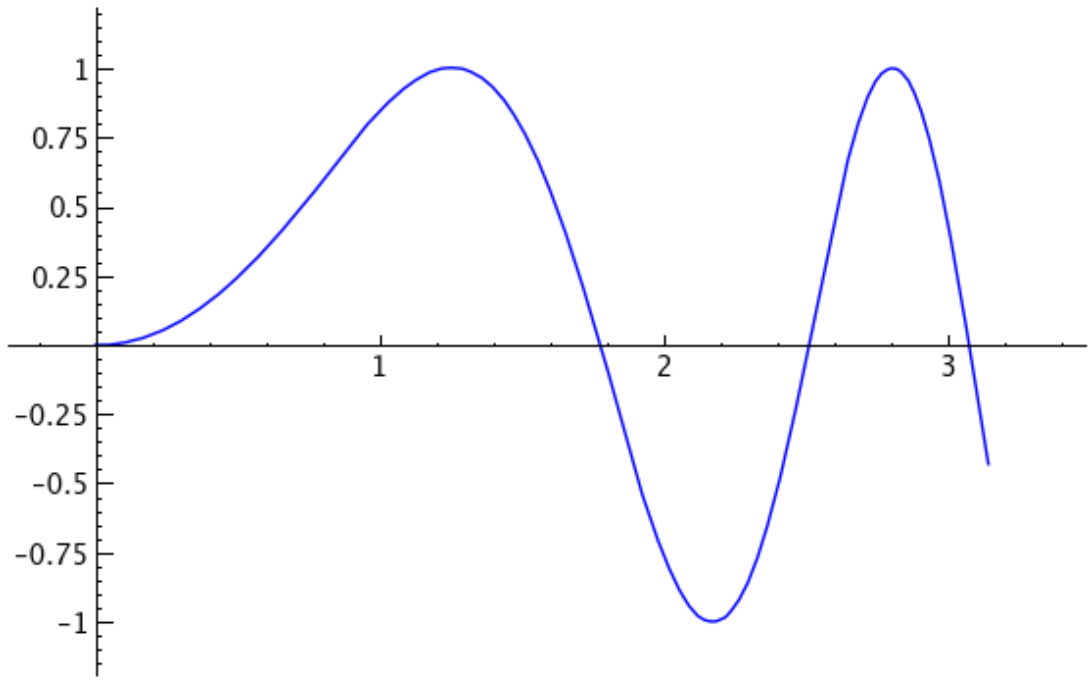
SAGE Demo Egham

Some Calculus

```
2 + 3
```

5

```
x = var('x')
show(plot(sin(x^2), 0, pi))
```




```
a = integrate(sin(x^2),x); a
```

```
sqrt(pi)*((sqrt(2)*I + sqrt(2))*erf((sqrt(2)*I + sqrt(2))*x/2) +
(sqrt(2)*I - sqrt(2))*erf((sqrt(2)*I - sqrt(2))*x/2))/8
```

```
show(a)
```

$$\frac{\sqrt{\pi}\left(\left(\sqrt{2}i+\sqrt{2}\right)\operatorname{erf}\left(\frac{\left(\sqrt{2}i+\sqrt{2}\right)x}{2}\right)+\left(\sqrt{2}i-\sqrt{2}\right)\operatorname{erf}\left(\frac{\left(\sqrt{2}i-\sqrt{2}\right)x}{2}\right)\right)}{8}$$

```
latex(a)
```

```
\frac{{\sqrt{ \pi } \left( {\left( {\sqrt{ 2 } i} + \sqrt{ 2 }
\right) \text{erf} \left( \frac{{\left( {\sqrt{ 2 } i} + \sqrt{ 2 }
\right) x}}{2} \right)} + {\left( {\sqrt{ 2 } i} - \sqrt{ 2 }
\right) \text{erf} \left( \frac{{\left( {\sqrt{ 2 } i} - \sqrt{ 2 }
\right) x}}{2} \right)} \right)}}{8}
```

```
var('a,b,c,X')
s = solve(a*X^2 + b*X + c == 0, X)
show(s[0])
```

$$X = \frac{-\left(\sqrt{b^2 - 4ac}\right) - b}{2a}$$

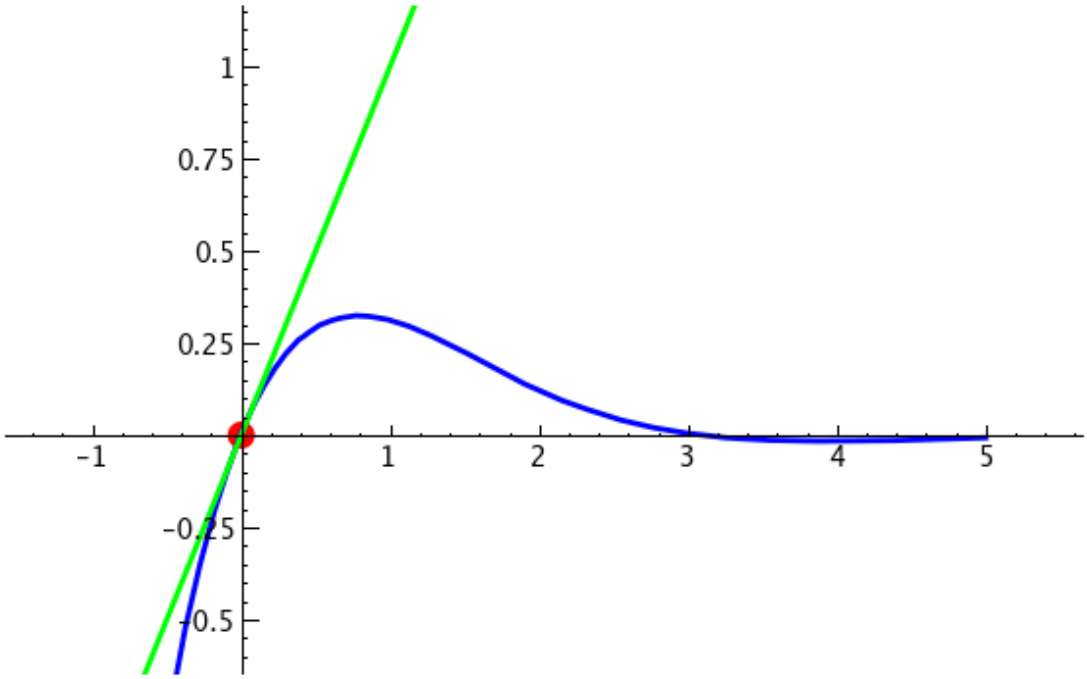
```
reset('e')
x = var('x')
f   = sin(x)*e^(-x)
p   = plot(f,-1,5, thickness=2)
@interact
def _(order=(1..12), x0=input_box("0",label="x_0")):
 x0 = int(x0)
 dot = point((x0,f(x0)),pointsize=80,rgbcolor=(1,0,0))
 ft = f.taylor(x,x0,order)
 pt = plot(ft,-1, 5, color='green', thickness=2)
 html('$f(x)\;=\;%s$'%latex(f))
 html('$\hat{f}(x;%s)\;=\;%s+\mathcal{O}(x^{%s})$'%
(x0,latex(ft),order+1))
 show(dot + p + pt, ymin = -.5, ymax = 1)
```

order

x_0

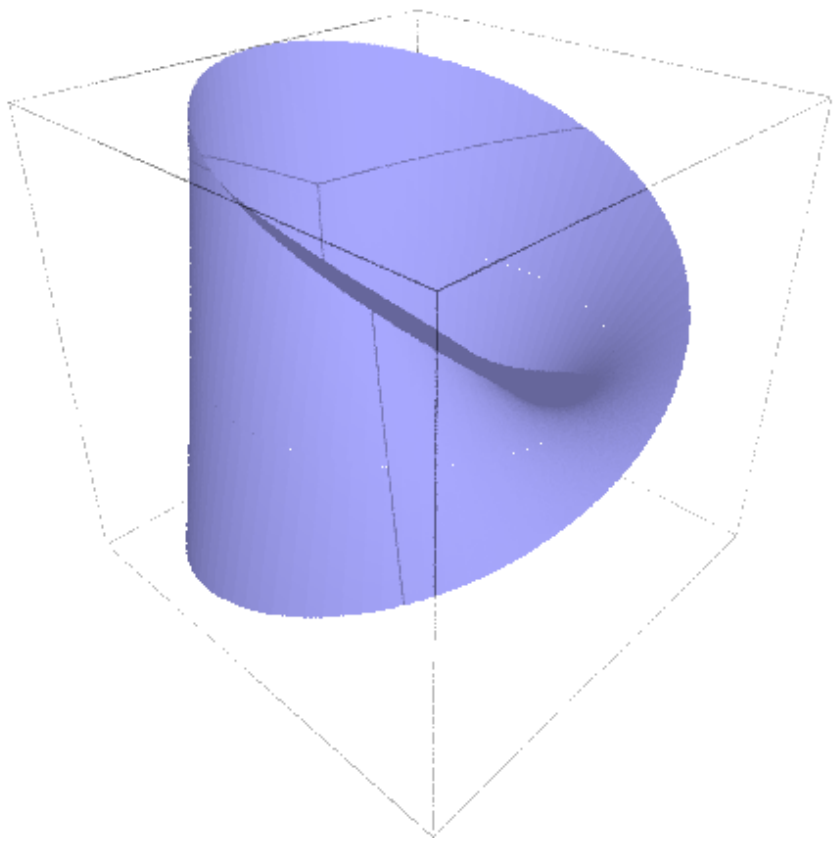
$f(x) \;=\; e^{-x}\sin(x)$

$\hat{f}(x;0) \;=\; x + \mathcal{O}(x^2)$



Interactive 3D Plotting

```
# Mobius strip:
u,v = var("u,v")
p = parametric_plot3d([cos(u)*(1+v*cos(u/2)), sin(u)*(1+v*cos(u/2)),
0.2*v*sin(u/2)], (u,0, 4*pi+0.5), (v,0, 0.3),plot_points=[200,200])
p.show(viewer='tachyon')
```



```
# Möbius strip:
u,v = var("u,v")
parametric_plot3d([cos(u)*(1+v*cos(u/2)), sin(u)*(1+v*cos(u/2)),
0.2*v*sin(u/2)], (u,0, 4*pi+0.5), (v,0, 0.3),plot_points=[140,140])
```

Finite Field Arithmetic

```
k.<a> = GF(2^8)
k
```

```
Finite Field in a of size 2^8
```

```
latex(k)
```

```
\mathbf{F}_{2^{8}}
```

```
show(k)
```

$\mathbf{F}_{2^8}$

```
P.<x> = GF(3)['x']
```

```
while True:
 p = P.random_element(degree=5)
 if p.is_irreducible() and p.degree() == 5:
   break
p
```

```
2*x^5 + x^4 + 2*x^3 + 2*x^2 + 2*x + 1
```

```
p = p/p.leading_coefficient()
p
```

```
x^5 + 2*x^4 + x^3 + x^2 + x + 2
```

```
k.<a> = GF(3^5, modulus=p)
k.modulus()
```

```
x^5 + 2*x^4 + x^3 + x^2 + x + 2
```

Dense Linear Algebra over the Rationals and Integers

The NTRUEncrypt Public Key Cryptosystem is based on the hard mathematical problem of finding very short vectors in lattices of very high dimension. Generate a ntru-like lattice of dimension (400×400), with the coefficients h_i chosen as random 130 bits integers and parameter $q = 35$:

$$\begin{pmatrix} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{d-1} \\ 0 & 1 & \dots & 0 & h_1 & h_2 & \dots & h_0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & h_{d-1} & h_0 & \dots & h_{d-1} \\ 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{pmatrix}$$

```
# this is not guaranteed to be LLL reduced (this is a demo so it has
to be really quick)
from sage.libs.fplll.fplll import gen_ntrulike
A = gen_ntrulike(400,400,35)
time B = A.LLL(algorithm='fpLLL:fast')
```

```
Time: CPU 0.51 s, Wall: 0.54 s
```

Sage ships NTL and fpLLL for LLL (and BKZ) reduction.

```
A = random_matrix(ZZ, 70, 70, x=-2^16,y=2^16, density=0.05)
time B = A.LLL(algorithm='NTL:LLL',fp='rr')
time C = A.LLL(algorithm='fpLLL:wrapper')
time _ = C.BKZ()
```

```
Time: CPU 2.86 s, Wall: 2.94 s
Time: CPU 1.49 s, Wall: 1.51 s
Time: CPU 0.04 s, Wall: 0.04 s
```

For some matrix shapes over **Q** Sage is very fast when computing the row echelon form.

```
n = 100
a = random_matrix(QQ,n, n+1, num_bound=2^64, den_bound=1)
time a.echelonize()
```

```
Time: CPU 0.15 s, Wall: 0.15 s
```

```
%magma
n := 100;
a := RMatrixSpace(RationalField(), n,n+1)![Random(1,2^64): i in
[1..n*(n+1)]];
time e := EchelonForm(a);
```

```
Time: 2.100
```

Dense Linear Algebra over Finite Fields

```
A = random_matrix(GF(32003),2000,2000)
B = random_matrix(GF(32003),2000,2000)
time C = A._multiply_linbox(B)
```

```
Time: CPU 2.48 s, Wall: 2.54 s
```

```
%magma
n := 2000;
A := RandomMatrix(GF(32003), n,n);
B := RandomMatrix(GF(32003), n,n);
time C := A*B;
```

```
Time: 2.580
```

```
time E = A.echelon_form()
```

```
CPU time: 1.52 s,  Wall time: 1.56 s
```

```
%magma
time E:=EchelonForm(A);
```

```
Time: 1.720
```

```
n = 10000
A = random_matrix(GF(2),n,n)
time E = A.echelon_form()
```

```
Time: CPU 1.53 s, Wall: 1.56 s
```

```
%magma
n := 10000;
A := RandomMatrix(GF(2), n, n);
time E := EchelonForm(A);
```

```
Time: 2.850
```

```
n = 10000
A = random_matrix(GF(2),n,n)
B = random_matrix(GF(2),n,n)
```

```
time C = A*B
```

```
Time: CPU 1.54 s, Wall: 1.55 s
```

```
%magma
n := 10000;
A := RandomMatrix(GF(2), n, n);
B := RandomMatrix(GF(2), n, n);
time C := A*B;
```

```
Time: 2.250
```

Sparse Linear Algebra over Finite Fields

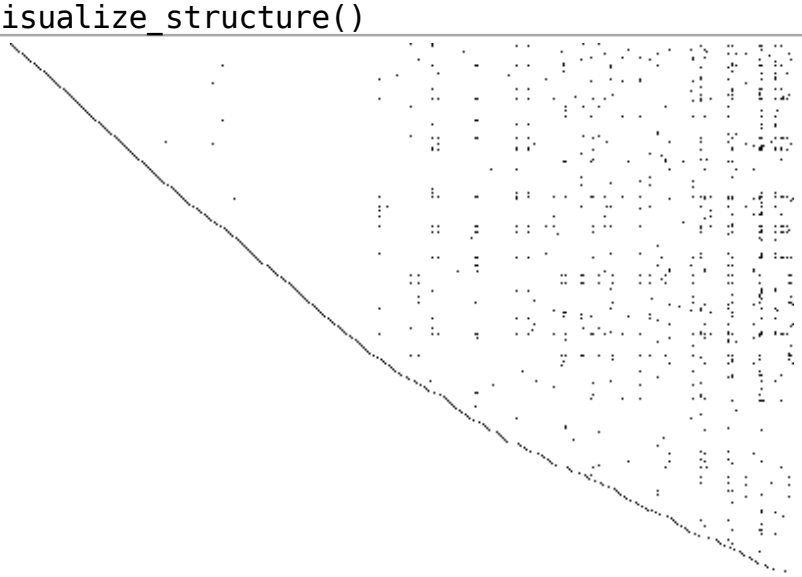
```
A = random_matrix(GF(127),10000,10000,density=2.0/10000,
sparse=True)
time A.rank()
time A.echelonize()
```

```
Time: CPU 1.06 s, Wall: 1.06 s
Time: CPU 1.74 s, Wall: 1.74 s
```

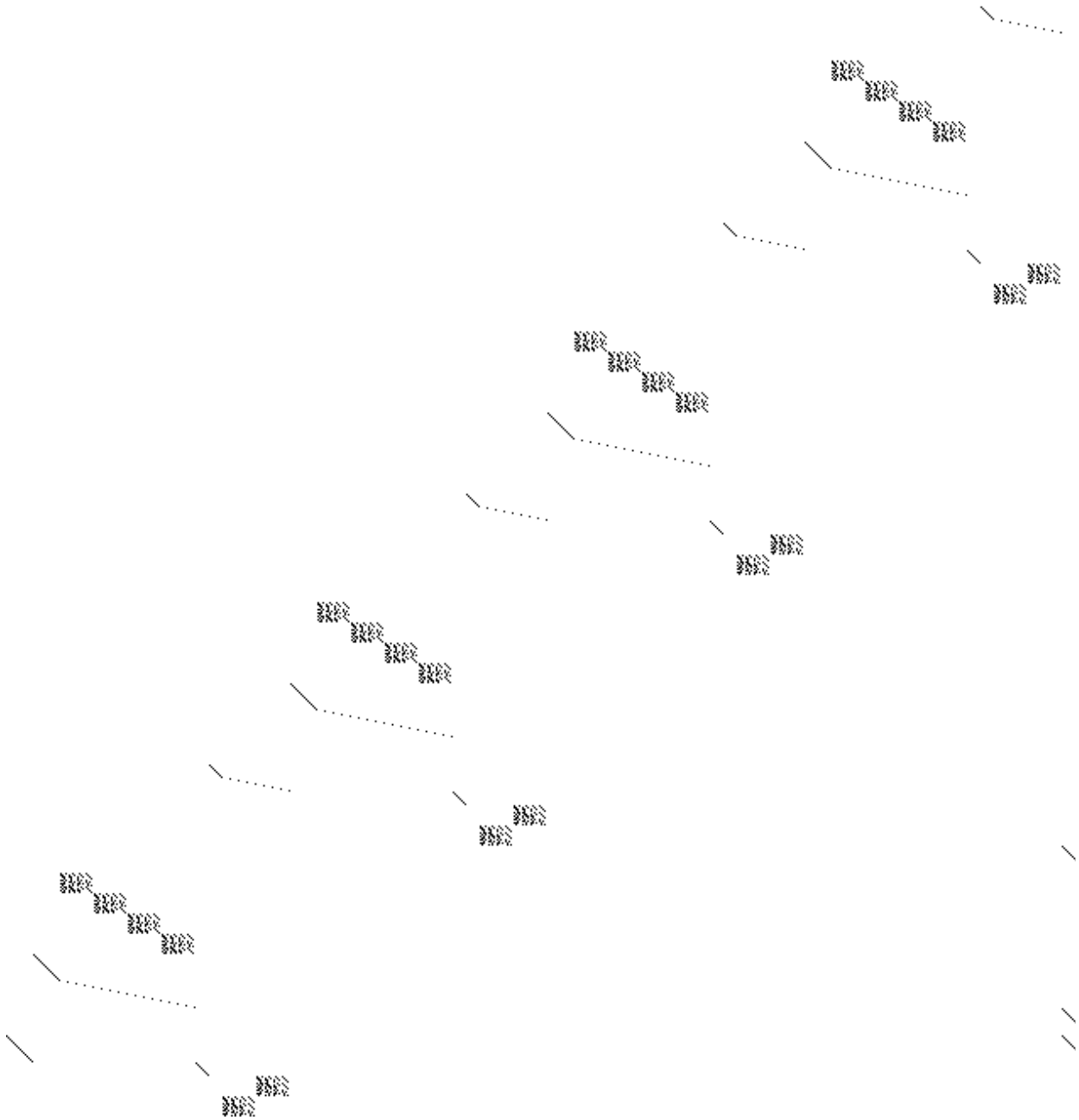
```
A = random_matrix(GF(127),1500,1500,density=10/1500, sparse=True)
b = random_matrix(GF(127),1500,1)
time c = A\b
(A*c) == b
```

```
Time: CPU 3.84 s, Wall: 3.92 s
True
```

```
n = 300
A = random_matrix(GF(127),n,n+100,sparse=True,density=2/n)
A.echelonize()
A.visualize_structure()
```



```
sr = mq.SR(4,2,2,4,gf2=True, allow_zero_inversions=True)
F,s = sr.polynomial_system()
A,v = F.coefficient_matrix()
A.visualize_structure(maxsize=1000)
```



Factoring

```
time factor(next_prime(2^40) * next_prime(2^300),verbose=0)
```

```
1099511627791 *
2037035976334486086268445688409378161051468393665936250636140449354 3
```

```
812997633367061833975 33
CPU time: 3.75 s,  Wall time: 3.81 s
```

```
time ecm.factor(next_prime(2^40) * next_prime(2^300))
```

```
[1099511627791,
2037035976334486086268445688409378161051468393665936250636140449354 3
812997633367061833975 33]
CPU time: 0.16 s,  Wall time: 0.58 s
```

```
v,t = qsieve(next_prime(2^90)*next_prime(2^91),time=True)
print v, t[:4]
```

```
[1237940039285380274899124357, 2475880078570760549798248507] 3.50
```

Elliptic Curves

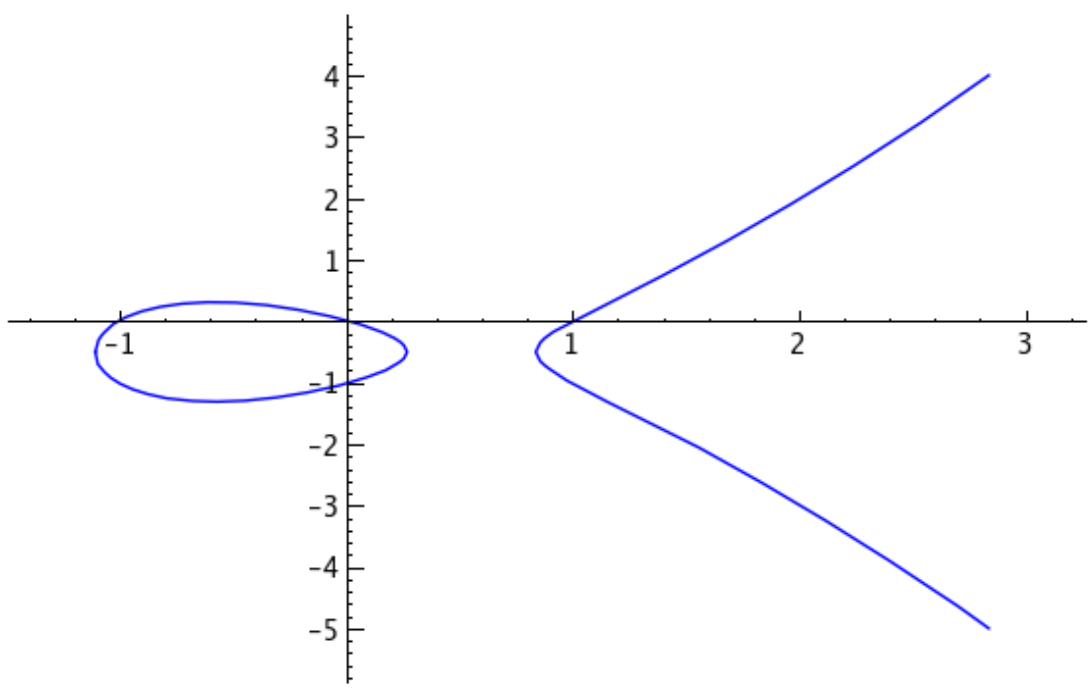
```
e = EllipticCurve("37a") # Cremona Label
show(e)
```

$$y^2 + y = x^3 - x$$

```
e.lift_x(2)
```

```
(2 : 2 : 1)
```

```
show(plot(e))
```




```
e.a_invariants()
```

```
[0, 0, 1, -1, 0]
```

The Fourier coefficients a_p of the modular form attached to this elliptic curve, for all primes $p \leq n$.

```
print e.aplist(20)
time n = e.aplist(10^6)
```

```
[-2, -3, -2, -1, -5, -2, 0, 0]
Time: CPU 4.69 s, Wall: 4.78 s
```

```
%magma
print GetVersion();
E := EllipticCurve([0,0,1,-1,0]);
print TracesOfFrobenius(E, 20);
time n := TracesOfFrobenius(E,10^6);
```

```
2 14 14

[ -2, -3, -2, -1, -5, -2, 0, 0 ]
Time: 5.420
```

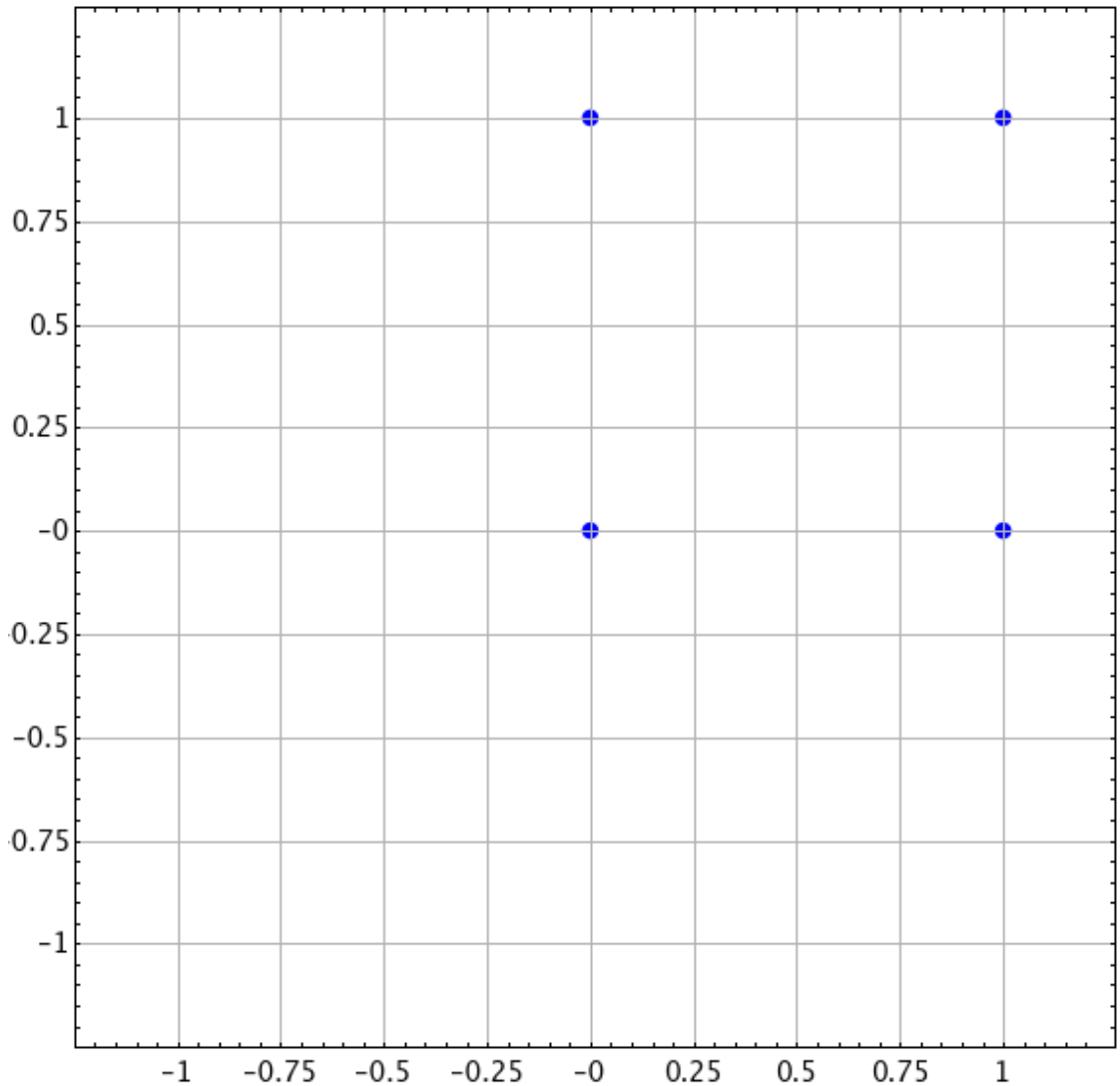
```
sage: E = elliptic_curves.rank(3)[0]
sage: E.integral_points(both_signs=False)
```

```
[(-3 : 0 : 1), (-2 : 3 : 1), (-1 : 3 : 1), (0 : 2 : 1), (1 : 0 : 1),
(2 : 0 : 1), (3 : 3 : 1), (4 : 6 : 1), (8 : 21 : 1), (11 : 35 : 1),
(14 : 51 : 1), (21 : 95 : 1), (37 : 224 : 1), (52 : 374 : 1), (93 :
896 : 1), (342 : 6324 : 1), (406 : 8180 : 1), (816 : 23309 : 1)]
```

```
@interact
def f(p=primes(2,500)):
   E = EllipticCurve('37a')
   show(E)
   show(plot(E.change_ring(GF(p)),pointsize=30), \
        axes=False, frame=True, gridlines="automatic",
aspect_ratio=1, gridlinesstyle={'rgbcolor':(0.7,0.7,0.7)})
```

p

$y^2 + y = x^3 - x$




```
k = GF(next_prime(10^7))
E = EllipticCurve(k, [k.random_element(),k.random_element()])
E
```

```
Elliptic Curve defined by y^2  = x^3 + 3248759*x + 6256184 over
Finite Field of size 10000019
```

```
P = E.random_element()
P.order()
```

```
666730
```

```
E.cardinality()
```

```
10000950
```

```
2*P + P
```

```
(2858307 : 6660064 : 1)
```

Number Fields

```
x = var('x')
K.<a> = NumberField(x^3 - 2)
L.<b> = K.galois_closure()
L
```

```
Number Field in b with defining polynomial x^6 + 40*x^3 + 1372
```

```
K.complex_embeddings()
```

```
[
Ring morphism:
  From: Number Field in a with defining polynomial x^3 - 2
  To:   Complex Double Field
  Defn: a |--> -0.629960524947 - 1.09112363597*I,
Ring morphism:
  From: Number Field in a with defining polynomial x^3 - 2
  To:   Complex Double Field
  Defn: a |--> -0.629960524947 + 1.09112363597*I,
Ring morphism:
  From: Number Field in a with defining polynomial x^3 - 2
  To:   Complex Double Field
  Defn: a |--> 1.25992104989
]
```

p-Adic Numbers

```
R = Zp(5)
e = R(3125346)
f = R(5*34234234)
t = cputime()
for i in xrange(10^6):
 z = e*f
cputime(t)
```

```
0.42393599999999765
```

```
%magma
R := pAdicRing(5);
e := R!3125346;
f := R!5*34234234;
time for i in [1..10^6] do
 z := e*f;
end for
```

```
Time: 0.670
```

```
gp.eval("e = %s"%(e._gp_().name()))
gp.eval("f = %s"%(f._gp_().name()))
```

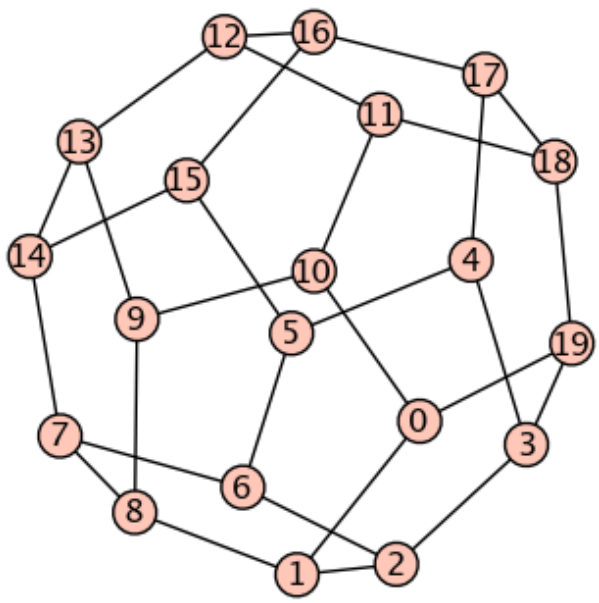
```
'4*5 + 5^2 + 4*5^3 + 3*5^4 + 4*5^5 + 4*5^6 + 3*5^8 + 2*5^9 + 2*5^10
+ 3*5^11 + O(5^21)'
```

```
%gp
gettime;
for(i=1,10^6,z=e*f);
gettime/1000.0
```

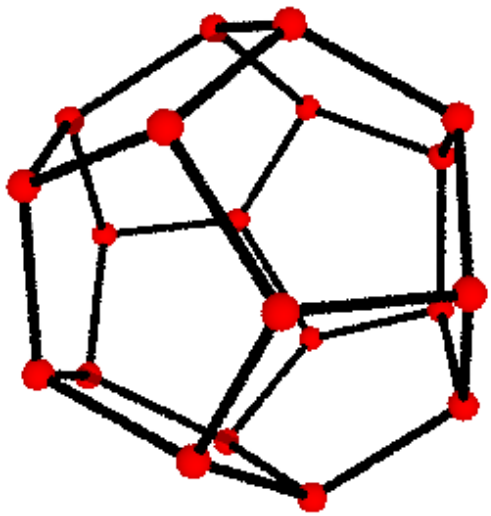
```
0.698000000000000000000000000000000000000
```

Graph Theory

```
D = graphs.DodecahedralGraph()
D.show()
```




```
D.show3d(viewer='tachyon')
```



```
D.show3d()
```

```
gamma = SymmetricGroup(20).random_element()
E = D.copy()
E.relabel(gamma)
D.is_isomorphic(E)
```

```
True
```

```
D.radius()
```

```
5
```

Group Theory

```
G = PermutationGroup([[(1,2,3),(4,5)], (3,4)]);
G
```

```
Permutation Group with generators [(1,2,3)(4,5), (3,4)]
```

```
G.order()
```

```
120
```

```
G.is_abelian()
```

```
False
```

```
G.derived_series()
```

```
[Permutation Group with generators [(1,2,3)(4,5), (3,4)],
Permutation Group with generators [(1,5)(3,4), (1,5)(2,3), (1,5,4)]]
```

```
G.random_element()
```

```
(1,5)(2,4)
```

```
G.center()
```

```
Permutation Group with generators [()]
```

```
SL4 = SL(4,GF(127)); SL4
```

```
Special Linear Group of degree 4 over Finite Field of size 127
```

```
SL4.order()
```

```
3606024503582215846241658863616 0
```

```
M = SL4.as_matrix_group(); M
```

```
Matrix group over Finite Field of size 127 with 2 generators:
 [[[3, 0, 0, 0], [0, 85, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]], [[126,
0, 0, 1], [126, 0, 0, 0], [0, 126, 0, 0], [0, 0, 126, 0]]]
```

```
M.is_finite()
```

```
True
```

Univariate Polynomials over Z

```
P.<x> = ZZ[]
deg = 32; coeff=64
f = P.random_element(degree=deg,x=0,y=2^coeff)
g = P.random_element(degree=deg,x=0,y=2^coeff)
```

```
type(f)
```

```
<type
'sage.rings.polynomial.polynomial_integer_dense_flint.Polynomial_int
eger_dense_flint'>
```

```
%time
for _ in xrange(10^4):
  w = f*g
```

```
CPU time: 0.14 s,  Wall time: 0.14 s
```

```
gp.eval('f = Pol(%s)'%f.list())
_ = gp.eval('g = Pol(%s)'%g.list())
```

```
%gp
gettime; for(i=1,10^4,w=f*g); gettime/1000.0
```

```
0.901000000000000000000000000000000000000
```

```
%magma
R<x> := PolynomialRing(IntegerRing());
```

```
magma.eval('f := R!%s'%f.list())
_ = magma.eval('g := R!%s'%g.list())
```

```
%magma
time for i in [1..10^4] do w := f*g; end for;
```

```
Time: 1.430
```

```
# NTL
ff = ntl.ZZX(f.list())
gg = ntl.ZZX(g.list())
```

```
time for i in xrange(10^4): w = ff*gg
```

```
CPU time: 0.78 s,  Wall time: 0.78 s
```

Multivariate Polynomial Rings over Finite Fields

```
#In SAGE
P.<x,y,z> = PolynomialRing(GF(32003),3)
p = (x + y + z + 1)^20 # the Fateman fastmult benchmark
q = p + 1
time r = p*q
```

```
Time: CPU 0.11 s, Wall: 0.11 s
```

```
%magma
P<x,y,z> := PolynomialRing(GF(32003),3);
p:= (x + y + z + 1)^20;
q:= p + 1;
time r:= p*q;
```

```
Time: 0.270
```

```
#In SAGE
P.<x,y,z> = PolynomialRing(QQ,3)
p = (x + y + z + 1)^20 # the Fateman fastmult benchmark
q = p + 1
time r = p*q
```

```
Time: CPU 1.03 s, Wall: 1.05 s
```

```
%magma
```

```
P<x,y,z> := PolynomialRing(RationalField(),3);
p := (x + y + z + 1)^20;
q := p + 1;
time r := p*q;
```

Time: 0.430

```
P = PolynomialRing(GF(32003),7,'x')
I = sage.rings.ideal.Cyclic(P,7)
t = cputime()
gb = I.groebner_basis('libsingular:std')
print 'Sage/Singular', cputime(t)

I = sage.rings.ideal.Cyclic(P,7)
t = magma.cputime()
gb = I.groebner_basis('magma:GroebnerBasis')
print 'MAGMA', magma.cputime(t)
```

Sage/Singular 2.188667
MAGMA 0.39

```
P.<x, y, z> = PolynomialRing(ZZ, 3)
I = ideal( x^2 - 3*y, y^3 - x*y, z^3 - x, x^4 - y*z + 1 )
R = P.change_ring( QQ )
I.change_ring( R ).groebner_basis()
```

[1]

```
@interact
def _(p=primes(1000)):
 gb0 = sorted(I.groebner_basis(),reverse=True)
 show(gb0)
 n = ZZ(gb0[-1])
 html("<div align='center'>$%s = %s</div>"%
(latex(n),latex(factor(n))))
 J = I.change_ring(I.ring().change_ring(GF(p)))
 for f in J.groebner_basis():
   show(f)
```

p

$$[x - 112418054324, y + 2149906854, z - 112352790903, 282687803443]$$

$$282687803443 = 101 \cdot 103 \cdot 27173681$$

1

Algebraic Cryptanalysis

```
sr = mq.SR(2,1,1,4,gf2=True)
sr
```

```
SR(2,1,1,4)
```

```
F,s = sr.polynomial_system()
F
```

```
Polynomial System with 104 Polynomials in 36 Variables
```

```
gb = F.groebner_basis()
Ideal(gb).variety()
```

```
[{s001: 1, s103: 1, s101: 0, x103: 0, s000: 1, x101: 0, k003: 0,
k100: 1, k001: 0, k200: 0, x200: 1, k202: 1, x202: 1, w102: 0, w100:
1, w201: 0, s002: 0, w203: 1, k101: 1, s102: 1, s100: 0, x102: 1,
x100: 0, k002: 1, k000: 1, x201: 1, k201: 0, x203: 0, k203: 0, k103:
0, w103: 1, k102: 1, w101: 0, w200: 0, s003: 0, w202: 1}]
```

Numerical Optimisation

For a simple quadratic programming example, if we want to minimize

$$2x_1^2 + x_2^2 + x_1x_2 + x_1 + x_2$$

subject to

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$x_1 + x_2 = 1$$

```
from cvxopt.base import matrix as m
from cvxopt import solvers

Q = 2r*m([ [2r, .5r], [.5r, 1r] ])
p = m([1.0r, 1.0r])
G = m([[-1.0r,0.0r],[0.0r,-1.0r]])
h = m([0.0r,0.0r])
A = m([1.0r, 1.0r], (1r,2r))
b = m(1.0r)
sol=solvers.qp(Q, p, G, h, A, b)
print sol['x']
```

```
     pcost       dcost       gap    pres   dres
 0:  0.0000e+00  0.0000e+00  3e+00  1e+00  0e+00
 1:  1.0776e+00  1.3668e+00  6e-01  4e-01  2e-16
 2:  1.8460e+00  1.8291e+00  6e-02  2e-02  4e-16
```

```
3:  1.8741e+00  1.8681e+00  8e-03  1e-03  3e-16
4:  1.8750e+00  1.8748e+00  2e-04  3e-05  5e-16
5:  1.8750e+00  1.8750e+00  3e-06  2e-07  1e-16
6:  1.8750e+00  1.8750e+00  3e-08  1e-09  6e-16
  2.5000e-01
  7.5000e-01
```

Statistics

```
S = AlphabeticStrings()
k = Permutations(range(26)).random_element()
K = S(k)
E = SubstitutionCryptosystem(S)
e = E(K)
```

```
m = "Sage can be used to study general and advanced, pure and
applied mathematics. This includes a huge range of mathematics,
including algebra, calculus, elementary to very advanced number
theory, cryptography, numerical computation, commutative algebra,
group theory, combinatorics, graph theory, exact linear algebra and
much more. It combines various software packages and seamlessly
integrates their functionality into a common experience. It is well
suited  for education, studying and research. The interface is a
notebook in a web-browser or the command-line. Using the notebook,
Sage connects either locally to your own Sage installation or to a
Sage server on the network. Inside the Sage notebook you can create
embedded graphics, beautifully typeset mathematical expressions, add
and delete input, and share your work across the network. The
following showcase presents some of Sage's capabilities, screenshots
and gives you an overall impression of what Sage is. The examples
show the lines of code in Sage on the left side, accompanied by an
explanation on the right. They only show the very basic concepts of
how Sage works. Please refer to the documentation material for more
detailed explanations or visit the library to see Sage in action."
m = S("".join([c.upper() for c in m if c.upper() in
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"]))
m
```

```
SAGECANBEUSEDTOSTUDYGENERALANDADVANCEDPUREANDAPPLIEDMATHEMATICSTHISI
NCLUDESAHUGERANGEOFMATHEMATICSINCLUDINGALGEBRACALCULUSELEMENTARYTOVE
RYADVANCEDNUMBERTHEORYCRYPTOGRAPHYNUMERICALCOMPUTATIONCOMMUTATIVEALG
EBRAGROUPTHEORYCOMBINATORICSGRAPHTHEORYEXACTLINEARALGEBRAANDMUCHMORE
ITCOMBINESVARIOUSSOFTWAREPACKAGESANDSEAMLESSLYINTEGRATESTHEIRFUNCTIO
NALITYINTOACOMMONEXPERIENCEITISWELLSUITEDFOREDUCATIONSTUDYINGANDRESE
ARCHTHEINTERFACEISANOTEBOOKINAWEBBROWSERORTHECOMMANDLINEUSINGTHENOTE
```

```
BOOKSAGECONNECTSEITHERLOCALLYTOYOUROWNSAGEINSTALLATIONORTOASAGESERVE
RONTHENETWORKINSIDETHESAGENOTEBOOKYOUCANCREATEEMBEDDEDGRAPHICSBEAUTI
FULLYTYPESETMATHEMATICALEXPRESSIONSADDANDDELETEINPUTANDSHAREYOURWORK
ACROSSTHENETWORKTHEFOLLOWINGSHOWCASEPRESENTSSOMEOFSAGESCAPABILITIESS
CREENSHOTSANDGIVESYOUANOVERALLIMPRESSIONOFWHATSAGEISTHEEXAMPLESSHOWT
HELINESOFCODEINSAGEONTHELEFTSIDEACCOMPANIEDBYANEXPLANATIONONTHERIGHT
THEYONLYSHOWTHEVERYBASICCONCEPTSOFHOWSAGEWORKSPLEASEREFERTOTHEDOCUME
NTATIONMATERIALFORMOREDETAILEDEXPLANATIONSORVISITTHELIBRARYTOSEESAGE
INACTION
```

```
c = e(m)
c
```

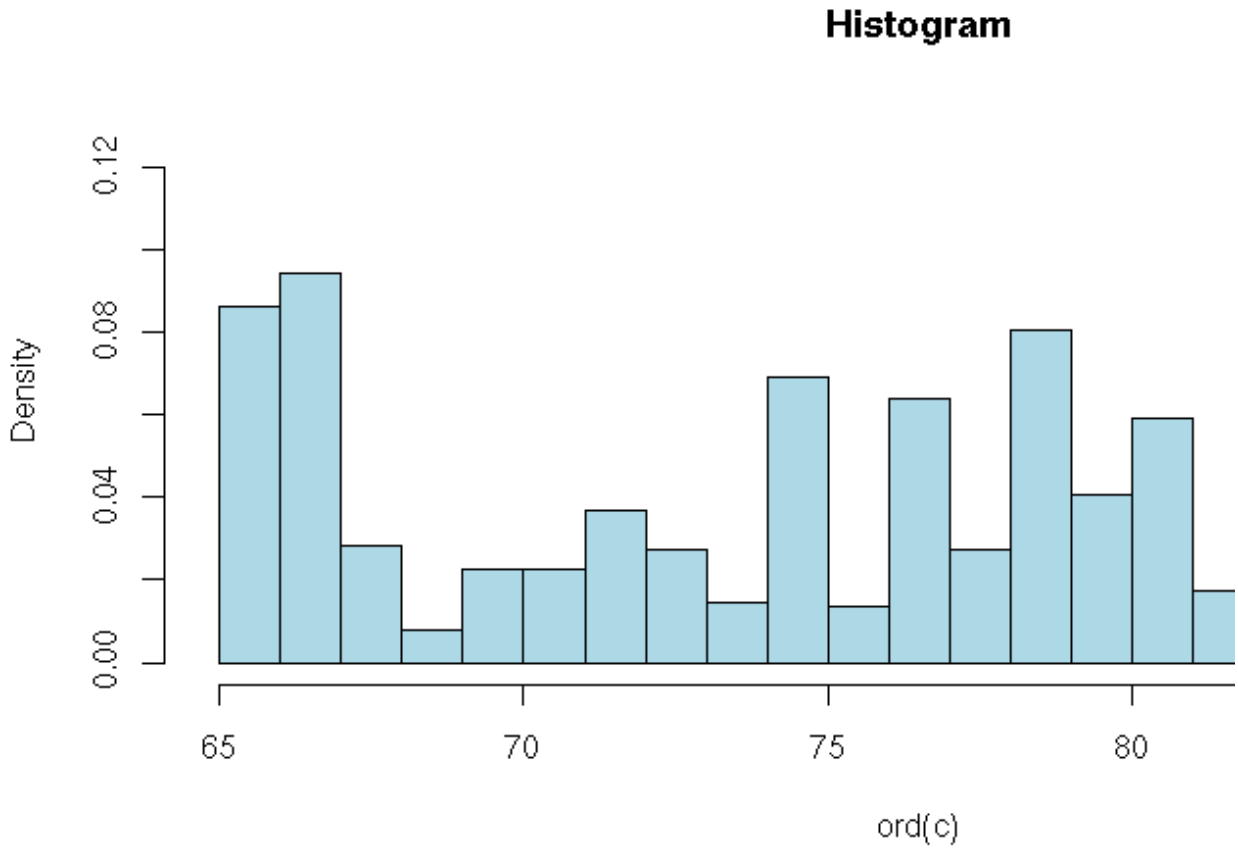
```
KCNUVCXRUIKUWOAKOIWGNUXUQCPCXWCWSCXVUWFIQUCXWCFFPMUWDCOHUDCOMVKOHMKM
XVPIWUKCHINUQCXNUALDCOHUDCOMVKMXVPIWMXNCPNURQCVCPVIPIKUPUDUXOCQGOASU
QGCWSCXVUWXIDRUQOHUAQGVQGFOANQCFHGXIDUQMVCPVADFIOCOMAXVADDIOCOMSUCPN
URQCNQAIFOHUAQGVADRMXCOAQMVKNQCFHOHUAQGUBCVOPMXUCQCPNURQCCXWDIVHDAQU
MOVADRMXUKSCQMAIKKALOJCQUFCVECNUKCXWKUCDPUKKPGMXOUNQCOUKOHUMQLIXVOMA
XCPMOGMXOACVADDAXUBFUQMUXVUMOMKJUPPKIMOUWLAQUWIVCOMAXKOIWGMXNCXWQUKU
CQVHOHUMXOUQLCVUMKCXAOURAAEMXCJURRQAJKUQAQOHUVADDCXWPMXUIKMXNOHUXAOU
RAAEKCNUVAXXUVOKUMOHUQPAVCPPGOAGAIQAJXKCNUMXKOCPPCOMAXAQOACKCNUKUQSU
QAXOHUXUOJAQEMXKMWUOHUKCNUXAOURAAEGAIVCXVQUCOUUDRUWWUWNQCFHMVKRUCIOM
LIPPGOGFUKUODCOHUDCOMVCPUBFQUKKMAXKCWWCXWWUPUOUMXFIOCXWKHCQUGAIQJAQE
CVQAKKOHUXUOJAQEOHULAPPAJMXNKHAJVCKUFQUKUXOKKADUALKCNUKVCFCRMPMOMUKK
VQUUXKHAOKCXWNMSUKGAICXASUQCPPMDFQUKKMAXALJHCOKCNUMKOHUUBCDFPUKKHAJC
HUPMXUKALVAWUMXKCNUAXOHUPULOKMWUCVVADFCXMUWRGCXUBFPCXCOMAXAXOHUQMNHC
OHUGAXPGKHAJOHUSUQGRCKMVVAXVUFOKALHAJKCNUJAQEKFPUCKUQULUQOAOHUWAVIDU
XOCOMAXDCOUQMCPLAQDAQUWUOCMPUWUBFPCXCOMAXKAQSMKMOOHUPMRQCQGOAKUUKCNU
MXCVOMAX
```

```
c = e(m)
c = map(ord,str(c))
c[:10]
```

```
[75, 67, 78, 85, 86, 67, 88, 82, 85, 73]
```

```
from rpy import r
r.png('histogram.png',width=900,height=480)
r.hist(c,r.seq(min(c),max(c),1),main="Histogram",col="lightblue",
prob=True, xlab="ord(c)")
r.dev_off()
```

```
{'null device': 1}
```




```
m[c.index(85)]
```

```
E
```

Combinatorics

```
C = Combinations(range(5)); C
```

```
Combinations of [0, 1, 2, 3, 4]
```

```
C.list()
```

```
[[], [0], [1], [2], [3], [4], [0, 1], [0, 2], [0, 3], [0, 4], [1,
2], [1, 3], [1, 4], [2, 3], [2, 4], [3, 4], [0, 1, 2], [0, 1, 3],
[0, 1, 4], [0, 2, 3], [0, 2, 4], [0, 3, 4], [1, 2, 3], [1, 2, 4],
[1, 3, 4], [2, 3, 4], [0, 1, 2, 3], [0, 1, 2, 4], [0, 1, 3, 4], [0,
2, 3, 4], [1, 2, 3, 4], [0, 1, 2, 3, 4]]
```

```
C.unrank(10)
```

```
[1, 2]
```

Interfaces

```
%gap
a := 1;
for i in [1..100] do if IsPrime(i) then a:=a+1; else a:=a+2; fi;
 od; a;
```

```
1
176
```

```
%singular
int a = 1; int i = 1;
for(i=1; i<=100; i=i+1) { if(prime(i) == i){  a=a+1; } else {a=a+2;}
};
a;
```

```
176
```

```
%gp
a=1; for(X=1,100,if(isprime(X),a+=1,a+=2)); a
```

```
176
```

```
a= gap(1) # or gp(1), magma(1), singular(1)
for i in range(100):
   if gap(i+1).IsPrime():
       a+=1
   else:
       a+=2
a
```

```
176
```

```
pari(a).factor()
```

```
[2, 4; 11, 1]
```

Sequences of Integers

```
for sq in sloane_find([2,3,5,7], 2):
 print sq[0], sq[1]
```

```
Searching Sloane's online database...
40 The prime numbers.
41 a(n) = number of partitions of n (the partition numbers).
```