Untitled

```
8^2
```

```
64
```

```
type(64)
```

```
<type 'sage.rings.integer.Integer'>
```

```
7/3
```

```
7/3
```

```
type(7/3)
```

```
<type 'sage.rings.rational.Rational'>
```

```
expand((x+1)^2)
```

```
x^2 + 2*x + 1
```

```
expand((y+1)^2)
```

```
Traceback (click to the left for traceback)
...
NameError: name 'y' is not defined
```

```
y = var('y')
expand((y+1)^2)
```

```
y^2 + 2*y + 1
```

```
expr = expand((y+1)^2)
type(expr)
```

```
<class 'sage.calculus.calculus.SymbolicArithmetic'>
```

```
range(9)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8]
```

```
range(1,9)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
2.sqrt().n()
```

```
1.41421356237310
```

```
n(sqrt(2))
```

```
1.41421356237310
```

```
i^2
```

```
-1
```

```
i = 1; i^2
```

```
1
```

```
reset('i'); i^2
```

```
-1
```

```
1+i
```

```
I + 1
```

```
n(pi)
```

```
3.14159265358979
```

```
n=1
```

```
n(pi)
```
```
Traceback (click to the left for traceback)
...
TypeError: 'sage.rings.integer.Integer' object is not callable
```
```
numerical_approx(pi)
```
```
3.14159265358979
```
```
type(add)
```
```
<type 'builtin_function_or_method'>
```
```
type(sin)
```
```
<class 'sage.calculus.calculus.Function_sin'>
```
```
type(exp)
```
```
<class 'sage.calculus.calculus.Function_exp'>
```
```
type(log)
```
```
<type 'function'>
```
```
type(ln)
```
```
<type 'function'>
```
```
type(bessel_J)
```
```
<type 'function'>
```
```
factor?
```
```
parent?
```
```
def f(x): return x^2
```
```
g(x) = x^2
```
```
h = function('h', x)
```
```
f.diff()
```
```
Traceback (click to the left for traceback)
...
AttributeError: 'function' object has no attribute 'diff'
```
```
g.diff()
```
```
x |--> 2*x
```
```
h.diff()
```
```
diff(h(x), x, 1)
```
```
desolve(diff(h,x)+h-1, [h,x])
```
```
'%e^-x*(%e^x+%c)'
```
```
type(f)
```
```
<type 'function'>
```

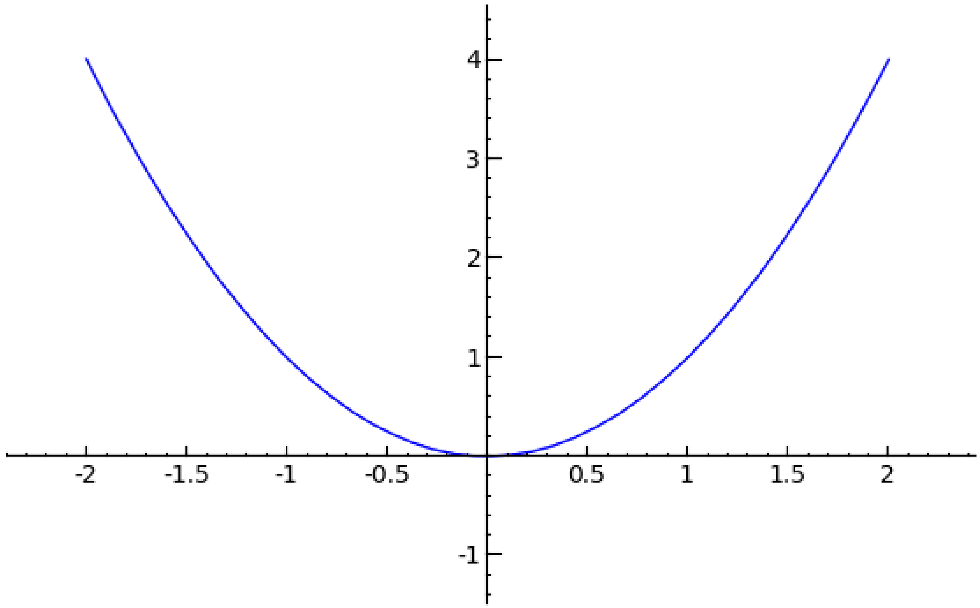
```
type(g)
```

```
<class 'sage.calculus.calculus.CallableSymbolicExpression'>
```

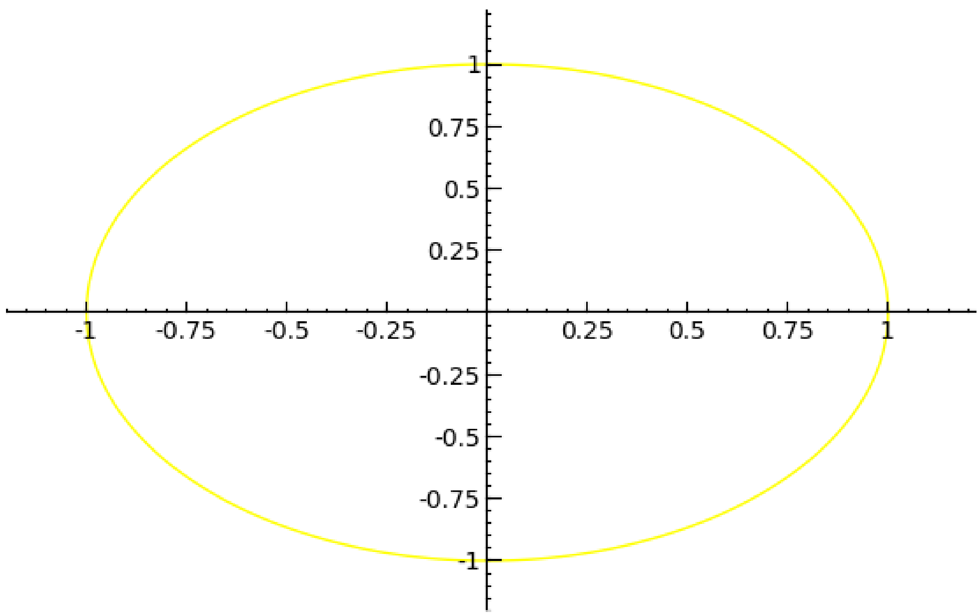
```
type(h)
```

```
<class 'sage.calculus.calculus.SymbolicFunctionEvaluation'>
```

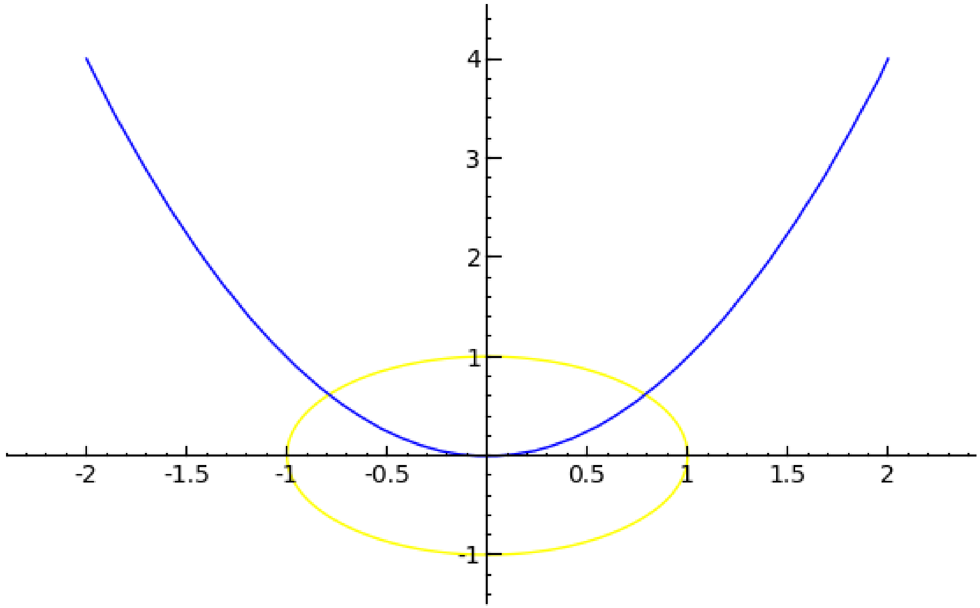
```
plot(f, (-2,2))
```




```
circle((0,0), 1, rgbcolor=(1,1,0))
```




```
p1 = plot(f, (-2,2))
p2 = circle((0,0), 1, rgbcolor=(1,1,0))
show(p1+p2)
```




```
def is_divisible_by(number, divisor=2):
      return number%divisor == 0
```

```
is_divisible_by(6)
```

    True

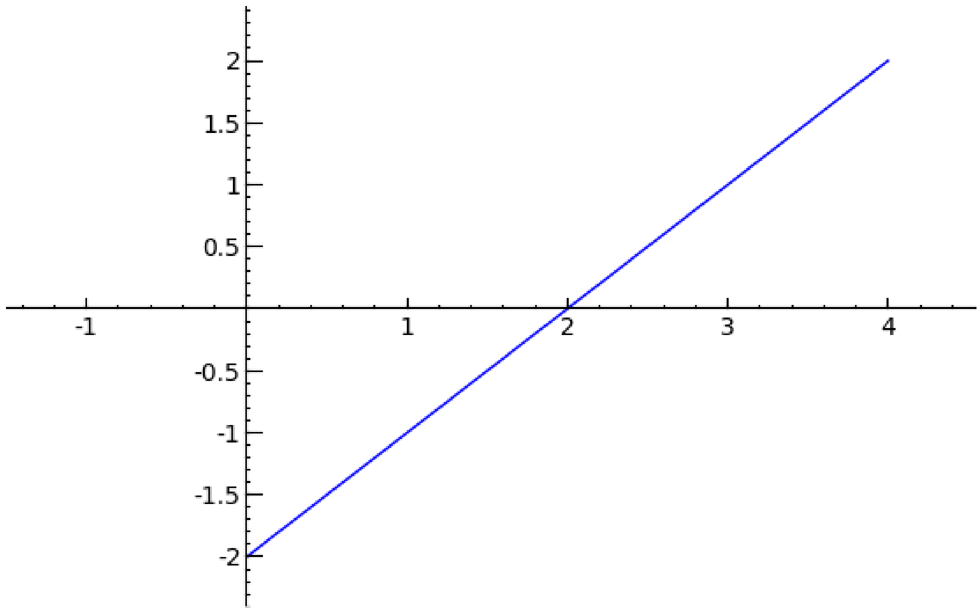
```
is_divisible_by(6,5)
```

    False

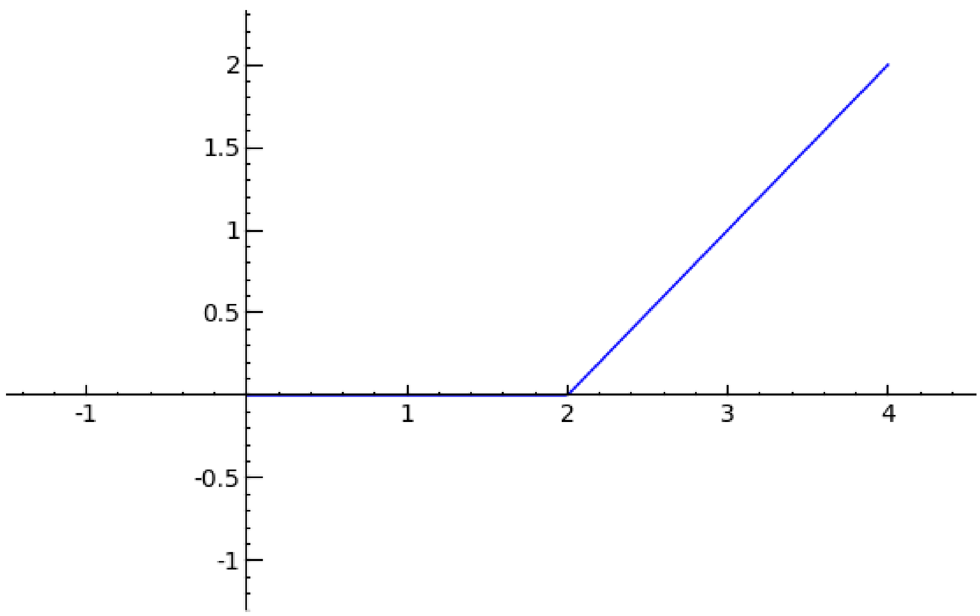
```
is_divisible_by(divisor=7, number=14)
```

    True

```
def f(x):
      if x<2:
          return 0
      else:
          return x-2
plot(f(x), 0, 4)
```




```
plot(f, 0, 4)
```




```
def cuberoot1(c, prec):
   a = 0;
   b = max(c, 1);
   count = 0;
   while (b-a > prec):
       m = (a+b)/2;
       if (m^3 < c):
           a = m
       else:
           b = m
       count = count + 1
```

```
    print 'did %s loops'%(count)
    return numerical_approx(m);
```

```
cuberoot1(27, 10^(-6))
```

```
did 25 loops
3.00000062584877
```

```
def cuberoot2(c, prec):
    x = 1
    count = 0
    while (abs(x^3-c) > prec):
        x = (2*x^3+c)/(3*x^2)
        count = count + 1
    print 'did %s loops'%(count)
    return numerical_approx(x);
```

```
cuberoot2(27, 10^(-6))
```

```
did 8 loops
3.00000000000010
```

```
time cuberoot1(500, 10^(-6))
```

```
did 29 loops
7.93700572103262
CPU time: 0.00 s,  Wall time: 0.00 s
```

```
time cuberoot2(500, 10^(-6))
```

```
did 13 loops
7.93700525984100
CPU time: 7.30 s,  Wall time: 7.31 s
```

```
class Evens(list):
      def __init__(self, n):
          self.n = n
          list.__init__(self, range(2, n+1, 2))
      def __repr__(self):
          return "Even positive numbers up to n."
```

```
e = Evens(10)
e
```

```
Even positive numbers up to n.
```

```
-4 in ZZ
```

```
True
```

```
pi in QQ
```

```
False
```

```
I+1 in RR
```

```
False
```

```
parent(4/3)
```

```
Rational Field
```

```
R = PolynomialRing(QQ, 't')
R
```

```
Univariate Polynomial Ring in t over Rational Field
```

```
S = QQ['t']
S
```

```
Univariate Polynomial Ring in t over Rational Field
```

```
type(t^2+1)
```

```
<type
'sage.rings.polynomial.multi_polynomial_libsingular.MPolynomial_libs\
ingular'>
```

```
T.<t> = PolynomialRing(QQ)
T
```

```
Univariate Polynomial Ring in t over Rational Field
```

```
type(t^2+1)
```

```
<class
'sage.rings.polynomial.polynomial_element_generic.Polynomial_rationa\
l_dense'>
```

```
R = PolynomialRing(RealField(10), 3, "rst")
R
```

```
Multivariate Polynomial Ring in r, s, t over Real Field with 10 bits
of precision
```

```
R.<r,s,t> = PolynomialRing(QQ)
R
```

```
Multivariate Polynomial Ring in r, s, t over Rational Field
```

```
R.gens()
```

```
(r, s, t)
```

```
R.objgens()
```

```
(Multivariate Polynomial Ring in r, s, t over Rational Field, (r, s,
t))
```

```
ring = PolynomialRing(QQ, 'x')
p1 = ring.random_element(degree=200);
p2 = ring.random_element(degree=200);
time p3 = p1*p2
```

```
Time: CPU 0.01 s, Wall: 0.01 s
```

```
v = var('v')
p1 = (p1 + v) - v
type(p1)
```

```
<class 'sage.calculus.calculus.SymbolicArithmetic'>
```

```
p2 = (p2 + v) - v
time p4 = expand(p1*p2)
```

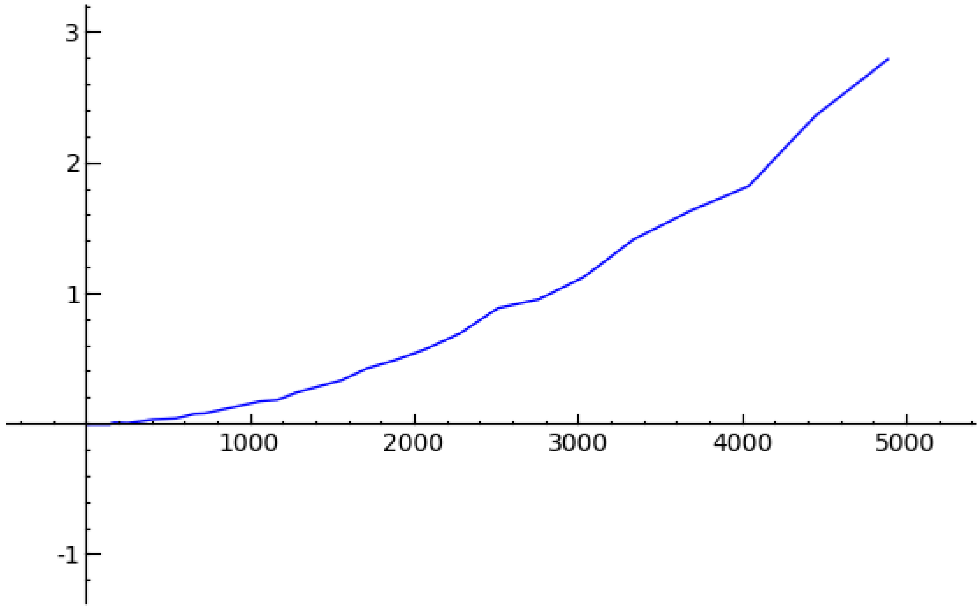
```
Time: CPU 0.06 s, Wall: 2.85 s
```

```
expand(p3 - p4)
```

```
0
```

```
def time_polymult(deg):
   ring = PolynomialRing(QQ, 'x')
   res = []
   d = 1
   while (d < deg):
       p1 = ring.random_element(degree=d)
       p2 = ring.random_element(degree=d)
       t = cputime()
       p1 = p1*p2
       res.append((d,cputime(t)))
       d = ceil(11/10*d)
   return res
```

```
timing = time_polymult(5000)
list_plot(timing, plotjoined=True)
```




```
x = QQ['x'].0
f = x^3 + 1
g = x^2 - 17
h = f/g; h
```

    (x^3 + 1)/(x^2 - 17)

```
h.parent()
```

    Fraction Field of Univariate Polynomial Ring in x over Rational
    Field

```
R.<x> = LaurentSeriesRing(QQ)
R
```

    Laurent Series Ring in x over Rational Field

```
GF(3)
```

    Finite Field of size 3

```
GF(27, 'a')
```

```
Finite Field in a of size 3^3
```

```
Zp(5)
```

```
5-adic Ring with capped relative precision 20
```

```
sqrt(3) in QQbar
```

```
True
```

```
R = PolynomialRing(QQ, 2, "xy")
x,y = R.gens()
f = x^2+y^2+1; g = x^3+2*x*y-y^3
```

```
(f,g)
```

```
(x^2 + y^2 + 1, x^3 - y^3 + 2*x*y)
```

```
id1 = (f,g)*R
```

```
id1
```

```
Ideal (x^2 + y^2 + 1, x^3 - y^3 + 2*x*y) of Multivariate Polynomial
Ring in x, y over Rational Field
```

```
id1.groebner_basis()
```

```
[x^2 + y^2 + 1, x*y^2 + y^3 - 2*x*y + x, y^4 - 3/2*x*y + y^2 + x - y
+ 1/2]
```

```
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.random_element()
```

```
(1,2)(3,5,4)
```

```
time factor(10914964370654197649021196782549355505887530307 1)
```

```
11052888000437433453781 9 * 98752148490260850514330 9
CPU time: 1.31 s,  Wall time: 1.32 s
```

```
time pari('factor(109149643706541976490211967825493555058875303071)')
```

```
[1105288800043743345378l9, 1; 9875214849026085051433O9, 1]
CPU time: 1.31 s,  Wall time: 1.32 s
```

```
time
mathematica('FactorInteger[109149643706541976490211967825493555058875303071]')
```

```
{{1105288800043743345378l9, 1}, {98752148490260850514330 9, 1}}
CPU time: 0.01 s,  Wall time: 7.85 s
```

```
print 'Now we switch to Singular (box at the top of the notebook)'
```

```
Now we switch to Singular (box at the top of the notebook)
```

```
ring r = 0,(t,x,y,z),ls;
```

```
r
```

```
//   characteristic : 0
//   number of vars : 4
//        block   1 : ordering ls
//                  : names    t x y z
//        block   2 : ordering C
```

```
ideal i1 = x-t2,y-t3,z;
```

```
i1
```

```
i1[1]=x-t2
i1[2]=y-t3
i1[3]=z
```

```
std(i1)
```

```
_[1]=z
_[2]=y-t3
_[3]=x-t2
```