## CONTENTS

1 Function Fields ................................................ 3
2 Elements of function fields ................................. 43
3 Orders of function fields ................................... 55
4 Ideals of function fields .................................... 75
5 Places of function fields .................................. 97
6 Divisors of function fields ............................... 103
7 Differentials of function fields ......................... 109
8 Valuation rings of function fields ...................... 113
9 Morphisms of function fields ........................... 117
10 Factories to construct function fields ................. 127
11 Indices and Tables ........................................ 131

Bibliography .................................................. 133
Python Module Index ......................................... 135
Index .......................................................... 137
Sage allows basic computations with elements and ideals in orders of algebraic function fields over arbitrary constant fields. Advanced computations, like computing the genus or a basis of the Riemann-Roch space of a divisor, are available for global function fields.

A reference for the basic theory of algebraic function fields is [Stich2009].
FUNCTION FIELDS

A function field (of one variable) is a finitely generated field extension of transcendence degree one. In Sage, a function field can be a rational function field or a finite extension of a function field.

EXAMPLES:

We create a rational function field:

```sage
sage: K.<x> = FunctionField(GF(5^2,'a')); K
Rational function field in x over Finite Field in a of size 5^2
sage: K.genus()
0
sage: f = (x^2 + x + 1) / (x^3 + 1)
sage: f
(x^2 + x + 1)/(x^3 + 1)
sage: f^3
(x^6 + 3*x^5 + x^4 + 2*x^3 + x^2 + 3*x + 1)/(x^9 + 3*x^6 + 3*x^3 + 1)
```

Then we create an extension of the rational function field, and do some simple arithmetic in it:

```sage
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^3 - (x^3 + 2*x*y + 1/x)); L
Function field in y defined by y^3 + 3*x*y + (4*x^4 + 4)/x
sage: y^2
y^2
sage: y^3
2*x*y + (x^4 + 1)/x
sage: a = 1/y; a
(x/(x^4 + 1))*y^2 + 3*x^2/(x^4 + 1)
sage: a * y
1
```

We next make an extension of the above function field, illustrating that arithmetic with a tower of three fields is fully supported:

```sage
sage: S.<t> = L[]
sage: M.<t> = L.extension(t^2 - x*y)
sage: M
Function field in t defined by t^2 + 4*x*y
sage: t^2
x*y
t
((1/(x^4 + 1))*y^2 + 3*x/(x^4 + 1))*t
sage: M.base_field()
Function field in y defined by y^3 + 3*x*y + (4*x^4 + 4)/x
```

(continues on next page)
It is also possible to construct function fields over an imperfect base field:

```
sage: N.<u> = FunctionField(K)
```

and inseparable extension function fields:

```
sage: J.<x> = FunctionField(GF(5)); J
Rational function field in x over Finite Field of size 5
sage: T.<v> = J[]
sage: O.<v> = J.extension(v^5 - x); O
Function field in v defined by v^5 + 4*x
```

## 1.1 Global function fields

Most of advanced computations are available only for global function fields as yet. A global function field in Sage is an extension field of a rational function field over a finite constant field by an irreducible separable polynomial over the rational function field.

**EXAMPLES:**

A fundamental computation for a global or any function field is to get a basis of its maximal order and maximal infinite order, and then do arithmetic with ideals of those maximal orders:

```
sage: K.<x> = FunctionField(GF(3)); _.<t> = K[]
sage: L.<y> = K.extension(t^4 + t - x^5)
sage: O = L.maximal_order()
sage: O.basis()
(1, y, 1/x*y^2 + 1/x*y, 1/x^3*y^3 + 2/x^3*y^2 + 1/x^3*y)
sage: I = O.ideal(x,y); I
Ideal (x, y + x) of Maximal order of Function field in y defined by y^4 + y + 2*x^5
sage: J = I^-1
sage: J.basis_matrix()
[ 1 0 0 0]
[1/x 1/x 0 0]
[ 0 0 1 0]
[ 0 0 0 1]
sage: L.maximal_order_infinite().basis()
(1, 1/x^2*y, 1/x^3*y^2, 1/x^4*y^3)
```

As an example of the most sophisticated computations that Sage can do with a global function field, we compute all the Weierstrass places of the Klein quartic over \( \mathbb{F}_2 \) and gap numbers for ordinary places:

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
sage: L.genus()
3
sage: L.weierstrass_places()
[Place (1/x, 1/x^3*y^2 + 1/x),
 Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1),
 Place (x, y),
 Place (x + 1, (x^3 + 1)*y + x + 1),
(continues on next page)
```
Place \((x^3 + x + 1, y + 1)\),
Place \((x^3 + x + 1, y + x^2)\),
Place \((x^3 + x + 1, y + x^2 + 1)\),
Place \((x^3 + x^2 + 1, y + x)\),
Place \((x^3 + x^2 + 1, y + x^2 + 1)\),
Place \((x^3 + x^2 + 1, y + x^2 + x + 1)\)

\[
\text{sage: } \text{L.gaps()}
\]
\[\{1, 2, 3\}\]

The gap numbers for Weierstrass places are of course not ordinary:

\[
\text{sage: } \text{p1, p2, p3 = L.weierstrass_places()[:3]}
\text{sage: } \text{p1.gaps()}
\[\{1, 2, 4\}\]
\text{sage: } \text{p2.gaps()}
\[\{1, 2, 4\}\]
\text{sage: } \text{p3.gaps()}
\[\{1, 2, 4\}\]

AUTHORS:

- William Stein (2010): initial version
- Robert Bradshaw (2010-05-30): added is_finite()
- Julian Rüth (2011-06-08, 2011-09-14, 2014-06-23, 2014-06-24, 2016-11-13): fixed hom(), extension(); use @cached_method; added derivation(); added support for relative vector spaces; fixed conversion to base fields
- Maarten Derickx (2011-09-11): added doctests
- Syed Ahmad Lavasani (2011-12-16): added genus(), is_RationalFunctionField()
- Simon King (2014-10-29): Use the same generator names for a function field extension and the underlying polynomial ring.
- Kwankyu Lee (2017-04-30): added global function fields

### class sage.rings.function_field.function_field.FunctionField(base_field, names, category=Category of function fields)

Bases: sage.rings.ring.Field

Abstract base class for all function fields.

INPUT:

- base_field – field; the base of this function field
- names – string that gives the name of the generator

EXAMPLES:

\[
\text{sage: } \text{K.<x> = FunctionField(QQ)}
\text{sage: } \text{K}
\text{Rational function field in x over Rational Field}
\]

\text{characteristic()}

Return the characteristic of the function field.

EXAMPLES:

1.1. Global function fields
```
sage: K.<x> = FunctionField(QQ)
sage: K.characteristic()
0
sage: K.<x> = FunctionField(GF(7))
sage: K.characteristic()
7
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: L.characteristic()
7
```

**divisor_group()**

Return the group of divisors attached to the function field.

**EXAMPLES:**

```
sage: K.<t> = FunctionField(QQ)
sage: K.divisor_group()
Divisor group of Rational function field in t over Rational Field
sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 - (x^3 - 1)/(x^3 - 2))
sage: L.divisor_group()
Divisor group of Function field in y defined by y^3 + (4*x^3 + 1)/(x^3 + 3)
```

**extension**(f, names=None)

Create an extension $K(y)$ of this function field $K$ extended with a root $y$ of the univariate polynomial $f$ over $K$.

**INPUT:**
- $f$ – univariate polynomial over $K$
- names – string or tuple of length 1 that names the variable $y$

**OUTPUT:**
- a function field

**EXAMPLES:**

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: K.extension(y^5 - x^3 - 3*x + x*y)
Function field in y defined by y^5 + x*y - x^3 - 3*x
```

A nonintegral defining polynomial:

```
sage: K.<t> = FunctionField(QQ); R.<y> = K[]
sage: K.extension(y^3 + (1/t)*y + t^3/(t+1), 'z')
Function field in z defined by z^3 + 1/t*z + t^3/(t + 1)
```

The defining polynomial need not be monic or integral:

```
sage: K.extension(t*y^3 + (1/t)*y + t^3/(t+1))
Function field in y defined by t*y^3 + 1/t*y + t^3/(t + 1)
```

**is_finite()**

Return whether the function field is finite, which is false.

**EXAMPLES:**

```
```
is_global()
Return whether the function field is global, that is, whether the constant field is finite.

EXAMPLES:

```python
sage: R.<t> = FunctionField(QQ)
sage: R.is_global()
False
sage: R.<t> = FunctionField(GF(7))
sage: R.is_global()
True
```

is_perfect()
Return whether the field is perfect, i.e., its characteristic $p$ is zero or every element has a $p$-th root.

EXAMPLES:

```python
sage: FunctionField(QQ, 'x').is_perfect()
True
sage: FunctionField(GF(2), 'x').is_perfect()
False
```

order(x, check=True)
Return the order generated by $x$ over the base maximal order.

INPUT:

- $x$ – element or list of elements of the function field
- check – boolean; if True, check that $x$ really generates an order

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^3 + x^3 + 4*x + 1)
sage: O = L.order(y); O
Order in Function field in y defined by y^3 + x^3 + 4*x + 1
sage: O.basis()
(1, y, y^2)
sage: Z = K.order(x); Z
Order in Rational function field in x over Rational Field
sage: Z.basis()
(1,)
```

Orders with multiple generators are not yet supported:

```python
sage: Z = K.order([x,x^2]); Z
Traceback (most recent call last):
  ...                                                
NotImplementedError
```
order_infinite \( (x, \text{check}=\text{True}) \)

Return the order generated by \( x \) over the maximal infinite order.

INPUT:

- \( x \) – element or a list of elements of the function field
- \( \text{check} \) – boolean; if True, check that \( x \) really generates an order

EXAMPLES:

```sage
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^3 + x^3 + 4*x + 1)
sage: L.order_infinite(y)  # todo: not implemented
```

Orders with multiple generators, not yet supported:

```sage
sage: Z = K.order(x); Z
Order in Rational function field in x over Rational Field
sage: Z.basis()
(1,)
```

order_infinite_with_basis \( (basis, \text{check}=\text{True}) \)

Return the order with given basis over the maximal infinite order of the base field.

INPUT:

- \( basis \) – list of elements of the function field
- \( \text{check} \) – boolean (default: True); if True, check that the basis is really linearly independent and that the module it spans is closed under multiplication, and contains the identity element.

EXAMPLES:

```sage
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^3 + x^3 + 4*x + 1)
sage: O = L.order_infinite_with_basis([1, 1/x*y, 1/x^2*y^2]); O
Infinite order in Function field in y defined by y^3 + x^3 + 4*x + 1
sage: O.basis()
(1, 1/x*y, 1/x^2*y^2)
```

Note that 1 does not need to be an element of the basis, as long it is in the module spanned by it:

```sage
sage: O = L.order_infinite_with_basis([1+1/x*y, 1/x*y, 1/x^2*y^2]); O
Infinite order in Function field in y defined by y^3 + x^3 + 4*x + 1
sage: O.basis()
(1/x*y + 1, 1/x*y, 1/x^2*y^2)
```

The following error is raised when the module spanned by the basis is not closed under multiplication:

```sage
sage: O = L.order_infinite_with_basis([1, y, 1/x^2*y^2]); O
Traceback (most recent call last):
... ValueError: the module generated by basis (1, y, 1/x^2*y^2) must be closed → under multiplication
```
and this happens when the identity is not in the module spanned by the basis:

```sage
sage: O = L.order_infinite_with_basis([1/x, 1/x*y, 1/x^2*y^2])
Traceback (most recent call last):
... ValueError: the identity element must be in the module spanned by basis (1/x, 1/x*y, 1/x^2*y^2)
```

**order_with_basis(basis, check=True)**

Return the order with given basis over the maximal order of the base field.

**INPUT:**

- `basis` – list of elements of this function field
- `check` – boolean (default: True); if True, check that the basis is really linearly independent and that the module it spans is closed under multiplication, and contains the identity element.

**OUTPUT:**

- an order in the function field

**EXAMPLES:**

```sage
sage: K.<x> = FunctionField(QQ); R.<y> = K[]; L.<y> = K.extension(y^3 + x^3 + 4*x + 1)
sage: O = L.order_with_basis([1, y, y^2]); O
Order in Function field in y defined by y^3 + x^3 + 4*x + 1
sage: O.basis()
(1, y, y^2)
```

Note that 1 does not need to be an element of the basis, as long it is in the module spanned by it:

```sage
sage: O = L.order_with_basis([1+y, y, y^2]); O
Order in Function field in y defined by y^3 + x^3 + 4*x + 1
sage: O.basis()
(y + 1, y, y^2)
```

The following error is raised when the module spanned by the basis is not closed under multiplication:

```sage
sage: O = L.order_with_basis([1, x^2 + x*y, (2/3)*y^2]); O
Traceback (most recent call last):
... ValueError: the module generated by basis (1, x*y + x^2, 2/3*y^2) must be closed under multiplication
```

and this happens when the identity is not in the module spanned by the basis:

```sage
sage: O = L.order_with_basics([x, x^2 + x*y, (2/3)*y^2])
Traceback (most recent call last):
... ValueError: the identity element must be in the module spanned by basis (x, x*y + x^2, 2/3*y^2)
```

**rational_function_field()**

Return the rational function field from which this field has been created as an extension.

**EXAMPLES:**

1.1. Global function fields
sage: K.<x> = FunctionField(QQ)
sage: K.rational_function_field()
Rational function field in x over Rational Field

sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2-x)
sage: L.rational_function_field()
Rational function field in x over Rational Field

sage: R.<z> = L[]
sage: M.<z> = L.extension(z^2-y)
sage: M.rational_function_field()
Rational function field in x over Rational Field

some_elements()
Return some elements in this function field.

EXAMPLES:

sage: K.<x> = FunctionField(QQ)
sage: K.some_elements()
[1,
x,
2*x,
x/(x^2 + 2*x + 1),
1/x^2,
x/(x^2 - 1),
x/(x^2 + 1),
1/2*x/(x^2 + 1),
0,
1/x,
...]

sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: L.some_elements()
[1,
y,
1/x*y,
(x + 1)/(x^2 - 2*x + 1)*y - 2*x/(x^2 - 2*x + 1),
1/x,
(1/(x - 1))*y,
(1/(x + 1))*y,
(1/2/(x + 1))*y,
0,
...]

space_of_differentials()
Return the space of differentials attached to the function field.

EXAMPLES:

sage: K.<t> = FunctionField(QQ)
sage: K.space_of_differentials()
Space of differentials of Rational function field in t over Rational Field

sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 - (x^3 - 1)/(x^3 - 2))
sage: L.space_of_differentials()
Space of differentials of Function field in y defined by y^3 + (4*x^3 + 1)/(x^3 - 3 + 3)

valuation (prime)
Return the discrete valuation on this function field defined by prime.

INPUT:

• prime – a place of the function field, a valuation on a subring, or a valuation on another function field together with information for isomorphisms to and from that function field

EXAMPLES:

We create valuations that correspond to finite rational places of a function field:

```
sage: K.<x> = FunctionField(QQ)
sage: v = K.valuation(1); v
(x - 1)-adic valuation
sage: v(x)
0
sage: v(x - 1)
1
```

A place can also be specified with an irreducible polynomial:

```
sage: v = K.valuation(x - 1); v
(x - 1)-adic valuation
```

Similarly, for a finite non-rational place:

```
sage: v = K.valuation(x^2 + 1); v
(x^2 + 1)-adic valuation
sage: v(x^2 + 1)
1
sage: v(x)
0
```

Or for the infinite place:

```
sage: v = K.valuation(1/x); v
Valuation at the infinite place
sage: v(x)
-1
```

Instead of specifying a generator of a place, we can define a valuation on a rational function field by giving a discrete valuation on the underlying polynomial ring:

```
sage: R.<x> = QQ[]
sage: w = valuations.GaussValuation(R, valuations.TrivialValuation(QQ)).
  augmentation(x - 1, 1)
sage: v = K.valuation(w); v
(x - 1)-adic valuation
```

Note that this allows us to specify valuations which do not correspond to a place of the function field:

1.1. Global function fields
sage: w = valuations.GaussValuation(R, QQ.valuation(2))
sage: v = K.valuation(w); v
2-adic valuation

The same is possible for valuations with \( v(1/x) > 0 \) by passing in an extra pair of parameters, an isomorphism between this function field and an isomorphic function field. That way you can, for example, indicate that the valuation is to be understood as a valuation on \( K[1/x] \), i.e., after applying the substitution \( x \mapsto 1/x \) (here, the inverse map is also \( x \mapsto 1/x \)):

sage: w = valuations.GaussValuation(R, QQ.valuation(2)).augmentation(x, 1)
sage: w = K.valuation(w)
sage: v = K.valuation((w, K.hom([-K.gen()]), K.hom([-K.gen()])))

Valuation on rational function field induced by \[ \text{Gauss valuation induced by } 2\text{-adic valuation, } v(x) = 1 \] (in Rational function field in x over Rational Field after x |--> 1/x)

Note that classical valuations at finite places or the infinite place are always normalized such that the uniformizing element has valuation 1:

sage: K.<t> = FunctionField(GF(3))
sage: M.<x> = FunctionField(K)
sage: v = M.valuation(x^3 - t)
sage: v(x^3 - t)
1

However, if such a valuation comes out of a base change of the ground field, this is not the case anymore. In the example below, the unique extension of \( v \) to \( L \) still has valuation 1 on \( x^3 - t \) but it has valuation \( 1/3 \) on its uniformizing element \( x - w \):

sage: R.<w> = K[]
sage: L.<w> = K.extension(w^3 - t)
sage: N.<x> = FunctionField(L)
sage: w = v.extension(N)
# missing factorization, :trac:`16572`
Traceback (most recent call last):
...
NotImplementedError
sage: w(x^3 - t) # not tested
1
sage: w(x - w) # not tested
1/3

There are several ways to create valuations on extensions of rational function fields:

sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x); L
Function field in y defined by y^2 - x

A place that has a unique extension can just be defined downstairs:

sage: v = L.valuation(x); v
(x)-adic valuation

class sage.rings.function_field.function_field.FunctionField_global (polynomial, names)

Bases: sage.rings.function_field.function_field.FunctionField_polymod

Global function fields.
INPUT:

- **polynomial** – monic irreducible and separable polynomial
- **names** – name of the generator of the function field

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[
```
```
sage: L.<y> = K.extension(Y^3 - (x^3 - 1)/(x^3 - 2))
sage: L
Function field in y defined by y^3 + (4*x^3 + 1)/(x^3 + 3)
```

The defining equation needs not be monic:

```
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[
```
```
sage: L.<y> = K.extension((1 - x)*Y^7 - x^3)
sage: L.gaps()
[1, 2, 3]
```

```
Element
```

alias of `sage.rings.function_field.element.FunctionFieldElement_global`

**completion** (*place*, **name=**`None`, **prec=**`None`, **gen_name=**`None`)

Return the completion of the function field at the place.

INPUT:

- **place** – place
- **name** – string; name of the series variable
- **prec** – positive integer; default precision
- **gen_name** – string; name of the generator of the residue field; used only when the place is non-rational

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[
```
```
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: m = L.completion(p); m
Completion map:
From: Function field in y defined by y^2 + y + (x^2 + 1)/x
To: Laurent Series Ring in s over Finite Field of size 2
s^2 + s^3 + s^4 + s^5 + s^7 + s^8 + s^9 + s^10 + O(s^12)
sage: m(x,10)
s^2 + s^3 + s^4 + s^5 + s^7 + s^8 + s^9 + s^10 + O(s^12)
sage: m(y,10)
s^-1 + 1 + s^3 + s^5 + s^7 + O(s^9)
```

```
constant_field()
```

Return the algebraic closure of the base constant field in the function field.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(3)); _.<y> = K[
```
```
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.constant_field()
Finite Field of size 3
```
**different ()**

Return the different of the function field.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: F.different()
2*Place (x, (1/(x^3 + x^2 + x))*y^2)
+ 2*Place (x^2 + x + 1, (1/(x^3 + x^2 + x))*y^2)
```

**exact_constant_field (name='t')**

Return the exact constant field and its embedding into the function field.

**INPUT:**

- **name** – name (default: t) of the generator of the exact constant field

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(3)); _.<Y> = K[]
sage: f = Y^2 - x*Y + x^2 + 1 # irreducible but not absolutely irreducible
sage: L.<y> = K.extension(f)
sage: L.genus()
0
sage: L.exact_constant_field()
(Finite Field in t of size 3^2, Ring morphism:
  From: Finite Field in t of size 3^2
  To:   Function field in y defined by y^2 + 2*x*y + x^2 + 1
  Defn: t |--> y + x)
sage: (y+x).divisor()
0
```

**gaps ()**

Return the gaps of the function field.

These are the gaps at the ordinary places, that is, places which are not Weierstrass places.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3 * Y + x)
sage: L.gaps()
[1, 2, 3]
```

**genus ()**

Return the genus of the function field.

**EXAMPLES:**

```python
sage: F.<a> = GF(16)
sage: K.<x> = FunctionField(F); K
Rational function field in x over Finite Field in a of size 2^4
sage: R.<t> = PolynomialRing(K)
sage: L.<y> = K.extension(t^4+t-x^5)
sage: L.genus()
6
```

The genus is computed by the Hurwitz genus formula.
higher_derivation()  
Return the higher derivation (also called the Hasse-Schmidt derivation) for the function field.  
The higher derivation of the function field is uniquely determined with respect to the separating element \(x\) of the base rational function field \(k(x)\).

**EXAMPLES:**

```python
sage: K.<x>=FunctionField(GF(5)); _.<Y>=K[]  
sage: L.<y>=K.extension(Y^3 - (x^3 - 1)/(x^3 - 2))  
sage: L.higher_derivation()  
Higher derivation map:  
  From: Function field in y defined by y^3 + (4*x^3 + 1)/(x^3 + 3)  
  To:  Function field in y defined by y^3 + (4*x^3 + 1)/(x^3 + 3)
```

maximal_order()  
Return the maximal order of the function field.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2));  
sage: R.<t> = PolynomialRing(K);  
sage: F.<y> = K.extension(t^4 + x^12*t^2 + x^18*t + x^21 + x^18);  
sage: O = F.maximal_order()  
sage: O.basis()  
(1, 1/x^4*y, 1/x^11*y^2 + 1/x^2, 1/x^15*y^3 + 1/x^6*y)
```

maximal_order_infinite()  
Return the maximal infinite order of the function field.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2));  
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)  
sage: F.maximal_order_infinite()  
Maximal infinite order of Function field in y defined by y^3 + x^6 + x^4 + x^2  
sage: K.<x> = FunctionField(GF(2));  
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x);  
sage: L.maximal_order_infinite()  
Maximal infinite order of Function field in y defined by y^2 + y + (x^2 + 1)/x
```

place_set()  
Return the set of all places of the function field.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2));  
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x);  
sage: L.place_set()  
Set of places of Function field in y defined by y^2 + y + (x^2 + 1)/x
```

places(\(degree=1\))  
Return a list of the places with degree.

**INPUT:**

- \(degree\) – positive integer (default: 1)

**EXAMPLES:**
sage: F.<a> = GF(2)
sage: K.<x> = FunctionField(F)
sage: R.<t> = PolynomialRing(K)
sage: L.<y> = K.extension(t^4 + t - x^5)
sage: L.places(1)
[Place (1/x, 1/x^4*y^3), Place (x, y), Place (x, y + 1)]

**places_finite**(degree=1)

Return a list of the finite places with degree.

**INPUT:**

• degree – positive integer (default: 1)

**EXAMPLES:**

sage: F.<a> = GF(2)
sage: K.<x> = FunctionField(F)
sage: R.<t> = PolynomialRing(K)
sage: L.<y> = K.extension(t^4 + t - x^5)
sage: L.places_finite(1)
[Place (x, y), Place (x, y + 1)]

**places_infinite**(degree=1)

Return a list of the infinite places with degree.

**INPUT:**

• degree – positive integer (default: 1)

**EXAMPLES:**

sage: F.<a> = GF(2)
sage: K.<x> = FunctionField(F)
sage: R.<t> = PolynomialRing(K)
sage: L.<y> = K.extension(t^4 + t - x^5)
sage: L.places_infinite(1)
[Place (1/x, 1/x^4*y^3)]

**residue_field**(place, name=None)

Return the residue field associated with the place along with the maps from and to the residue field.

**INPUT:**

• place – place of the function field

• name – string; name of the generator of the residue field

The domain of the map to the residue field is the discrete valuation ring associated with the place.

The discrete valuation ring is defined as the ring of all elements of the function field with nonnegative valuation at the place. The maximal ideal is the set of elements of positive valuation. The residue field is then the quotient of the discrete valuation ring by its maximal ideal.

If an element not in the valuation ring is applied to the map, an exception TypeError is raised.

**EXAMPLES:**

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: R, fr_R, to_R = L.residue_field(p)
sage: R
Finite Field of size 2
sage: f = 1 + y
sage: f.valuation(p)
-1
sage: to_R(f)
Traceback (most recent call last):
  ... TypeError: ...
sage: (1+1/f).valuation(p)
0
sage: to_R(1 + 1/f)
1
sage: [fr_R(e) for e in R]
[0, 1]

\texttt{weierstrass\_places()}

Return all Weierstrass places of the function field.

\textbf{EXAMPLES:}

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3 * Y + x)
sage: L.weierstrass_places()
[Place (1/x, 1/x^3*y^2 + 1/x),
 Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1),
 Place (x, y),
 Place (x + 1, (x^3 + 1)*y + x + 1),
 Place (x^3 + x + 1, y + 1),
 Place (x^3 + x^2 + 1, y + x^2 + 1),
 Place (x^3 + x^2 + 1, y + x^2 + x + 1)]

\textbf{class sage.rings.function_field.function_field.FunctionField\_global\_integral (polynomial, names)}

Bases: \texttt{sage.rings.function_field.function_field.FunctionField\_global}

Global function fields defined by an irreducible and separable polynomial, which is integral over the maximal order of the base rational function field with a finite constant field.

\texttt{equation\_order()}

Return the equation order of the function field.

\textbf{EXAMPLES:}

sage: K.<x> = FunctionField(GF(2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: F.equation_order()
Order in Function field in y defined by y^3 + x^6 + x^4 + x^2

\texttt{equation\_order\_infinite()}

Return the infinite equation order of the function field.

This is by definition $o[b]$ where $b$ is the primitive integral element from \texttt{primitive\_integral\_element\_infinite()} and $o$ is the maximal infinite order of the base rational function field.
EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: F.equation_order_infinite()
Infinite order in Function field in y defined by y^3 + x^6 + x^4 + x^2
```

```
primitive_integal_element_infinite()
Return a primitive integral element over the base maximal infinite order.
This element is integral over the maximal infinite order of the base rational function field and the function field is a simple extension by this element over the base order.
```

```python
sage: K.<x> = FunctionField(GF(2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: b = F.primitive_integal_element_infinite(); b
1/x^2*y
sage: b.minimal_polynomial('t')
t^3 + (x^4 + x^2 + 1)/x^4
```

```
class sage.rings.function_field.function_field.FunctionField_polymod(polynomial, names, category=None)

Bases: sage.rings.function_field.function_field.FunctionField

Function fields defined by a univariate polynomial, as an extension of the base field.
```

INPUT:

- polynomial – univariate polynomial over a function field
- names – tuple of length 1 or string; variable names
- category – category (default: category of function fields)

EXAMPLES:

We make a function field defined by a degree 5 polynomial over the rational function field over the rational numbers:

```python
sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x)); L
Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x
```

We next make a function field over the above nontrivial function field L:

```python
sage: S.<z> = L[]
sage: M.<z> = L.extension(z^2 + y*z + y); M
Function field in z defined by z^2 + y*z + y
```

We drill down the tower of function fields:
Warning: It is not checked if the polynomial used to define the function field is irreducible. Hence it is not guaranteed that this object really is a field! This is illustrated below.

```python
sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(x^2 - y^2)
sage: (y - x)*(y + x)
0
sage: 1/(y - x)
1
sage: y - x == 0; y + x == 0
False False
```

**Element**

alias of `sage.rings.function_field.element.FunctionFieldElement_polymod`

**base_field()**

Return the base field of the function field. This function field is presented as \( L = K[y]/(f(y)) \), and the base field is by definition the field \( K \).

**EXAMPIES:**

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.base_field()
Rational function field in x over Rational Field
```

**change_variable_name(name)**

Return a field isomorphic to this field with variable(s) name.

**INPUT:**

- name – a string or a tuple consisting of a strings, the names of the new variables starting with a generator of this field and going down to the rational function field.

**OUTPUT:**

A triple \( F, f, t \) where \( F \) is a function field, \( f \) is an isomorphism from \( F \) to this field, and \( t \) is the inverse of \( f \).

**EXAMPIES:**

```python
sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: R.<z> = L[]
sage: M.<z> = L.extension(z^2 - y)
```

(continues on next page)
sage: M.change_variable_name('zz')
(Function field in zz defined by zz^2 - y,
 Function Field morphism:
    From: Function field in zz defined by zz^2 - y
    To:   Function field in z defined by z^2 - y
    Defn: zz |--> z
          y |--> y
          x |--> x,
 Function Field morphism:
    From: Function field in z defined by z^2 - y
    To:   Function field in zz defined by zz^2 - y
    Defn: z |--> zz
          y |--> y
          x |--> x)

sage: M.change_variable_name(('zz','yy'))
(Function field in zz defined by zz^2 - yy, Function Field morphism:
    From: Function field in zz defined by zz^2 - yy
    To:   Function field in z defined by z^2 - y
    Defn: zz |--> z
            yy |--> y
            x |--> x,
    Function Field morphism:
    From: Function field in z defined by z^2 - y
    To:   Function field in zz defined by zz^2 - yy
    Defn: z |--> zz
            y |--> yy
            x |--> xx)

sage: M.change_variable_name(('zz','yy','xx'))
(Function field in zz defined by zz^2 - yy,
 Function Field morphism:
    From: Function field in zz defined by zz^2 - yy
    To:   Function field in z defined by z^2 - y
    Defn: zz |--> z
            yy |--> y
            xx |--> x,
    Function Field morphism:
    From: Function field in z defined by z^2 - y
    To:   Function field in zz defined by zz^2 - yy
    Defn: z |--> zz
            y |--> yy
            x |--> xx)

current_base_field()
Return the constant field of the base rational function field.

EXAMPLES:

sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x)); L
Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x
sage: L.constant_base_field()
Rational Field
sage: S.<z> = L[]
sage: M.<z> = L.extension(z^2 - y)
sage: M.constant_base_field()
Rational Field

current_domain()
Return the algebraic closure of the constant field of the base field in the function field.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.constant_field()
Traceback (most recent call last):
...  
NotImplementedError
```

degree (base=\text{None})
Return the degree of the function field over the function field base.

INPUT:

- \text{base} – a function field (default: None), a function field from which this field has been constructed as a finite extension.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x)); L
Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x
sage: L.degree()
5
sage: L.degree(L)
1
sage: R.<z> = L[]
sage: M.<z> = L.extension(z^2 - y)
sage: M.degree(L)
2
sage: M.degree(K)
10
```

derivation()
Return a generator of the space of derivations over the constant base ring of this function field \( K \).

A derivation on \( K \) is map \( K \rightarrow K \) with \( D(\alpha + \beta) = D(\alpha) + D(\beta) \) and \( D(\alpha \beta) = \beta D(\alpha) + \alpha D(\beta) \) for all \( \alpha, \beta \in K \).

If the base field \( k \) of \( K \) is perfect, then the derivations on \( K \) form a one-dimensional \( K \)-vector space. (More generally, this is true if \( K \) is separable over \( k \), or in other words if the corresponding curve is geometrically reduced over \( k \); this is automatically the case if \( k \) is perfect.) We apply the techniques from [GT1996] to find a generator of this one-dimensional vector space, which is then returned by the algorithm.

ALGORITHM:

If \( K \) is a separable extension of another function field \( F \) between \( K \) and \( k \), then Proposition 11 of [GT1996] describes how to compute the unique extension of a derivation on \( F \) to \( K \); we then apply this algorithm to the generator of the space of derivations on \( F \), which we may calculate inductively. If \( K \) is not given as a separable extension of another function field, then we find a field isomorphic to \( K \) that is a separable extension of a rational function field over \( k \) by using \texttt{separable_model()} . This part of the algorithm uses the assumption that \( k \) is perfect.

EXAMPLES:
```python
tsage: K.<x> = FunctionField(GF(3))
tsage: R.<y> = K[]
tsage: L.<y> = K.extension(y^2 - x)
tsage: d = L.derivation(); d
Derivation map:
  From: Function field in y defined by y^2 + 2*x
  To: Function field in y defined by y^2 + 2*x
  Defn: y |--> 2/x*y
  x |--> 1
sage: d(x)
1
sage: d(x^3)
0
sage: d(x*y)
0
sage: d(y)
2/x*y
```

This also works for inseparable extensions:

```python
tsage: R.<y> = K[]
tsage: L.<y> = K.extension(y^3 - x)
tsage: d = L.derivation(); d
Derivation map:
  From: Function field in y defined by y^3 + 2*x
  To: Function field in y defined by y^3 + 2*x
  Defn: y |--> 1
  x |--> 0
sage: d(x^2)
0
sage: d(y^2)
2*y
sage: d(x*y)
x
```

equation_order()

Return the equation order of the function field.

If we view the function field as being presented as $K[y]/(f(y))$, then the order generated by the class of $y$ is returned. If $f$ is not monic, then `_make_monic_integral()` is called, and instead we get the order generated by some integral multiple of a root of $f$.

EXAMPLES:

```python
tsage: K.<x> = FunctionField(QQ); R.<y> = K[]
tsage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
tsage: O = L.equation_order()
tsage: O.basis()
(1, x*y, x^2*y^2, x^3*y^3, x^4*y^4)
```

We try an example, in which the defining polynomial is not monic and is not integral:

```python
tsage: K.<x> = FunctionField(QQ); R.<y> = K[]
tsage: L.<y> = K.extension(x^2*y^5 - 1/x); L
Function field in y defined by x^2*y^5 - 1/x
sage: O = L.equation_order()
sage: O.basis()
(1, x*y, x^2*y^2, x^3*y^3, x^4*y^4)
```

22 Chapter 1. Function Fields
gen \((n=0)\)

Return the \(n\)-th generator of the function field. By default, \(n\) is 0; any other value of \(n\) leads to an error. The generator is the class of \(y\), if we view the function field as being presented as \(K[y]/(f(y))\).

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.gen()
Traceback (most recent call last):
  ... IndexError: there is only one generator
```

genus()

Return the genus of the function field.

For now, the genus is computed using Singular.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[
```
We make a morphism of the base rational function field:

```
sage: phi = K.hom(x+1); phi
Function Field endomorphism of Rational function field in x over Rational
  Defn: x |--> x + 1
sage: phi(x^3 - 3)
x^3 + 3*x^2 + 3*x - 2
sage: (x+1)^3-3
x^3 + 3*x^2 + 3*x - 2
```

We make a morphism by specifying where the generators and the base generators go:

```
sage: L.hom([-y, x])
Function Field endomorphism of Function field in y defined by y^2 - x^3 - 1
  Defn: y |--> -y
  x |--> x
```

You can also specify a morphism on the base:

```
sage: R1.<q> = K[]
sage: L1.<q> = K.extension(q^2 - (x+1)^3 - 1)
sage: L1.hom(q, base_morphism=phi)
Function Field morphism:
  From: Function field in y defined by y^2 - x^3 - 1
  To:   Function field in q defined by q^2 - x^3 - 3*x^2 - 3*x - 2
  Defn: y |--> q
  x |--> x + 1
```

We make another extension of a rational function field:

```
sage: K2.<t> = FunctionField(QQ); R2.<w> = K2[]
sage: L2.<w> = K2.extension((4*w)^2 - (t+1)^3 - 1)
```

We define a morphism, by giving the images of generators:

```
sage: f = L2.hom([4*w, t+1]); f
Function Field morphism:
  From: Function field in y defined by y^2 - x^3 - 1
  To:   Function field in w defined by 16*w^2 - t^3 - 3*t^2 - 3*t - 2
  Defn: y |--> 4*w
  x |--> t + 1
```

Evaluation works, as expected:

```
sage: f(y+x)
4*w + t + 1
sage: f(x*y + x/(x^2+1))
(4*t + 4)*w + (t + 1)/(t^2 + 2*t + 2)
```

We make another extension of a rational function field:

```
sage: K3.<yy> = FunctionField(QQ); R3.<xx> = K3[]
sage: L3.<xx> = K3.extension(yy^2 - xx^3 - 1)
```
This is the function field $L$ with the generators exchanged. We define a morphism to $L$:

```
sage: g = L3.hom([x,y]); g
Function Field morphism:
    From: Function field in xx defined by -xx^3 + yy^2 - 1
    To:   Function field in y defined by y^2 - x^3 - 1
    Defn: xx |--> x
          yy |--> y
```

**is_separable** *(base=``None``)*

Return whether this is a separable extension of `base`.

**INPUT:**

- `base` – a function field from which this field has been created as an extension or `None` (default: `None`); if `None`, then return whether this is a separable extension over its base field.

**EXAMPLES:**

```
sage: K.<x> = FunctionField(GF(2))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: L.is_separable()
False
sage: R.<z> = L[]
sage: M.<z> = L.extension(z^3 - y)
sage: M.is_separable()
True
sage: M.is_separable(K)
False
sage: K.<x> = FunctionField(GF(5))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.is_separable()
True
sage: K.<x> = FunctionField(GF(5))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^5 - 1)
sage: L.is_separable()
False
```

**maximal_order** *(*)

Return the maximal order of the function field.

**EXAMPLES:**

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.maximal_order()  # todo: not implemented
```

**maximal_order_infinite** *(*)

Return the maximal infinite order of the function field.

**EXAMPLES:**

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.maximal_order_infinite()  # todo: not implemented
```

1.1. Global function fields 25
monic_integral_model(names=None)

Return a function field isomorphic to this field but which is an extension of a rational function field
with defining polynomial that is monic and integral over the constant base field.

INPUT:

- names – a string or a tuple of up to two strings (default: None), the name of the generator of the
  field, and the name of the generator of the underlying rational function field (if a tuple); if not given, then
  the names are chosen automatically.

OUTPUT:

A triple \((F, f, t)\) where \(F\) is a function field, \(f\) is an isomorphism from \(F\) to this field, and \(t\) is the inverse
of \(f\).

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(x^2*y^5 - 1/x); L
Function field in y defined by x^2*y^5 - 1/x
sage: A, from_A, to_A = L.monic_integral_model('z')
sage: A
Function field in z defined by z^5 - x^12
sage: from_A
Function Field morphism:
    From: Function field in z defined by z^5 - x^12
    To: Function field in y defined by x^2*y^5 - 1/x
    Defn: z |--> x^3*y
          x |--> x
sage: to_A
Function Field morphism:
    From: Function field in y defined by x^2*y^5 - 1/x
    To: Function field in z defined by z^5 - x^12
    Defn: y |--> 1/x^3*z
          x |--> x
sage: to_A(y)
1/x^3*z
sage: from_A(to_A(y))
y
sage: from_A(to_A(1/y))
x^3*y^4
sage: from_A(to_A(1/y)) == 1/y
True
```

This also works for towers of function fields:

```python
sage: R.<z> = L[]
sage: M.<z> = L.extension(z^2*y - 1/x)
sage: M.monic_integral_model()
(Function Field in z defined by z^10 - x^18, Function Field morphism:
    From: Function field in z defined by z^10 - x^18
    To:    Function field in z defined by y*z^2 - 1/x
    Defn: z |--> x^2*z
            x |--> x, Function Field morphism:
    From: Function field in z defined by y*z^2 - 1/x
    To:    Function field in z defined by z^10 - x^18
    Defn: z |--> 1/x^2*z
            y |--> 1/x^15*z^8
            x |--> x)
```
ngens()

Return the number of generators of the function field over its base field. This is by definition 1.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.ngens()
1
```

polynomial()

Return the univariate polynomial that defines the function field, that is, the polynomial \( f(y) \) so that the function field is of the form \( K[y]/(f(y)) \).

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.polynomial()
y^5 - 2*x*y + (-x^4 - 1)/x
```

polynomial_ring()

Return the polynomial ring used to represent elements of the function field. If we view the function field as being presented as \( K[y]/(f(y)) \), then this function returns the ring \( K[y] \).

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: L.polynomial_ring()
Univariate Polynomial Ring in y over Rational function field in x over Rational Field
```

primitive_element()

Return a primitive element over the underlying rational function field.

If this is a finite extension of a rational function field \( K(x) \) with \( K \) perfect, then this is a simple extension of \( K(x) \), i.e., there is a primitive element \( y \) which generates this field over \( K(x) \). This method returns such an element \( y \).

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2-x)
sage: R.<z> = L[]
sage: M.<z> = L.extension(z^2-y)
sage: R.<z> = L[]
sage: N.<u> = L.extension(z^2-x-1)
sage: N.primitive_element()
u + y
sage: M.primitive_element()
z
sage: L.primitive_element()
y
```

This also works for inseparable extensions:
### random_element (*args, **kwds*)
Create a random element of the function field. Parameters are passed onto the random_element method of the base field.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - (x^2 + x))
sage: L.random_element()  # random
((x^2 - x + 2/3)/(x^2 + 1/3*x - 1))*y^2 + ((-1/4*x^2 + 1/2*x - 1)/(-5/2*x + 2/3))*y
+ (-1/2*x^2 - 4)/(-12*x^2 + 1/2*x - 1/95)
```

### separable_model (names=None)
Return a function field isomorphic to this field which is a separable extension of a rational function field.

**INPUT:**

- *names* – a tuple of two strings or `None` (default: `None`); the second entry will be used as the variable name of the rational function field, the first entry will be used as the variable name of its separable extension. If `None`, then the variable names will be chosen automatically.

**OUTPUT:**

A triple `(F, f, t)` where `F` is a function field, `f` is an isomorphism from `F` to this function field, and `t` is the inverse of `f`.

**ALGORITHM:**

Suppose that the constant base field is perfect. If this is a monic integral inseparable extension of a rational function field, then the defining polynomial is separable if we swap the variables (Proposition 4.8 in Chapter VIII of [Lang2002].) The algorithm reduces to this case with `monic_integral_model()`.

**REFERENCES:**

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3)
sage: L.separable_model(('t','w'))
(Function field in t defined by t^3 + w^2,
 Function Field morphism:
    From: Function field in t defined by t^3 + w^2
    To:   Function field in y defined by y^2 + x^3
    Defn: t |--> x
           w |--> y,
 Function Field morphism:
    From: Function field in y defined by y^2 + x^3
    To:   Function field in t defined by t^3 + w^2
    Defn: y |--> w
           x |--> t)
```
This also works for non-integral polynomials:

```python
sage: K.<x> = FunctionField(GF(2))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2/x - x^2)
sage: L.separable_model()
(Function field in y_ defined by y_^3 + x_^2,
  Function Field morphism:
  From: Function field in y_ defined by y_^3 + x_^(2)
  To:   Function field in y defined by 1/x*y^2 + x^2
  Defn: y_ |--> x
         x_ |--> y,
  Function Field morphism:
  From: Function field in y defined by 1/x*y^2 + x^2
  To:   Function field in y_ defined by y_^(3) + x_^(2)
  Defn: y |--> x_
         x |--> y_)
```

If the base field is not perfect this is only implemented in trivial cases:

```python
sage: k.<t> = FunctionField(GF(2))
sage: k.is_perfect()
False
sage: K.<x> = FunctionField(k)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^3 - t)
sage: L.separable_model()
(Function field in y defined by y^3 + t,
  Function Field endomorphism of Function field in y defined by y^3 + t
  Defn: y |--> y
  x |--> x,
  Function Field endomorphism of Function field in y defined by y^3 + t
  Defn: y |--> y
  x |--> x)
```

Some other cases for which a separable model could be constructed are not supported yet:

```python
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - t)
sage: L.separable_model()
Traceback (most recent call last):
  ...
NotImplementedError: constructing a separable model is only implemented for simple_model
```

```python
simple_model (name=None)
```

Return a function field isomorphic to this field which is a simple extension of a rational function field.

**INPUT:**

- name – a string (default: None), the name of generator of the simple extension. If None, then the name of the generator will be the same as the name of the generator of this function field.

**OUTPUT:**

A triple \((F, f, t)\) where \(F\) is a field isomorphic to this field, \(f\) is an isomorphism from \(F\) to this function field and \(t\) is the inverse of \(f\).

**EXAMPLES:**

A tower of four function fields:
The fields $N$ and $M$ as simple extensions of $K$:

```
sage: N.simple_model()
(Function field in v defined by $v^8 - x$,
 Function Field morphism:
   From: Function field in v defined by $v^8 - x$
   To:   Function field in v defined by $v^2 - u$
   Defn: v |--> v,
 Function Field morphism:
   From: Function field in v defined by $v^2 - u$
   To:   Function field in v defined by $v^8 - x$
   Defn: v |--> v
   u |--> v^2
   z |--> v^4
   x |--> x)
sage: M.simple_model()
(Function field in u defined by $u^4 - x$,
 Function Field morphism:
   From: Function field in u defined by $u^4 - x$
   To:   Function field in u defined by $u^2 - z$
   Defn: u |--> u,
 Function Field morphism:
   From: Function field in u defined by $u^2 - z$
   To:   Function field in u defined by $u^4 - x$
   Defn: u |--> u
   z |--> u^2
   x |--> x)
```

An optional parameter name can be used to set the name of the generator of the simple extension:

```
sage: M.simple_model(name='t')
(Function field in t defined by $t^4 - x$, Function Field morphism:
   From: Function field in t defined by $t^4 - x$
   To:   Function field in u defined by $u^2 - z$
   Defn: t |--> u,
 Function Field morphism:
   From: Function field in u defined by $u^2 - z$
   To:   Function field in t defined by $t^4 - x$
   Defn: u |--> t
   z |--> t^2
   x |--> x)
```

An example with higher degrees:

```
sage: K.<x> = FunctionField(GF(3)); R.<y> = K[]
sage: L.<y> = K.extension(y^5-x); R.<z> = L[]
sage: M.<z> = L.extension(z^3-x)
sage: M.simple_model()
(Function field in z defined by $z^{15} + x*z^{12} + x^2*z^9 + 2*x^3*z^6 + 2*x^4*z^3 + 2*x^5 + 2*x^3$,
 Function Field morphism:
   From: Function field in z defined by $z^{15} + x*z^{12} + x^2*z^9 + 2*x^3*z^6 + 2*x^4*z^3 + 2*x^5 + 2*x^3$
   To:   Function field in z defined by $z^{15} + x*z^{12} + x^2*z^9 + 2*x^3*z^6 + 2*x^4*z^3 + 2*x^5 + 2*x^3$
   Defn: _ |--> _)
```

(continues on next page)
To: Function field in z defined by $z^3 + 2x$
Defn: $z \mapsto z + y$,

Function Field morphism:
From: Function field in z defined by $z^3 + 2x$
To: Function field in z defined by $z^{15} + xz^{12} + x^2z^9 + 2x^3z^6 + ...$
\[ \rightarrow 2x^4z^3 + 2x^5 + 2x^3 \]
Defn: $z \mapsto 2/xz^6 + 2z^3 + z + 2x$
y $\mapsto 1/xz^6 + z^3 + x$
x $\mapsto x$)

This also works for inseparable extensions:

```
sage: K.<x> = FunctionField(GF(2)); R.<y> = K[

sage: L.<y> = K.extension(y^2-x); R.<z> = L[

sage: M.<z> = L.extension(z^2-y)

sage: M.simple_model()

( Function field in z defined by $z^4 + x$, Function Field morphism:
  From: Function field in z defined by $z^4 + x$
  To: Function field in z defined by $z^2 + y$
  Defn: $z \mapsto z$, Function Field morphism:
  From: Function field in z defined by $z^2 + y$
  To: Function field in z defined by $z^4 + x$
  Defn: $z \mapsto z$
y $\mapsto z^2$
x $\mapsto x$)
```

vector_space(base=None)

Return a vector space and isomorphisms from the field to and from the vector space.

This function allows us to identify the elements of this field with elements of a vector space over the base field, which is useful for representation and arithmetic with orders, ideals, etc.

**INPUT:**
- `base` – a function field (default: None), the returned vector space is over `base` which defaults to the base field of this function field.

**OUTPUT:**
- a vector space over the base function field
- an isomorphism from the vector space to the field
- an isomorphism from the field to the vector space

**EXAMPLES:**

We define a function field:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[

sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x)); L

Function field in y defined by $y^5 - 2*x*y + (-x^4 - 1)/x$
```

We get the vector spaces, and maps back and forth:

```
sage: V, from_V, to_V = L.vector_space()
sage: V

Vector space of dimension 5 over Rational function field in x over RationalField
```

(continues on next page)
We convert an element of the vector space back to the function field:

```
sage: from_V(V.1)
y
```

We define an interesting element of the function field:

```
sage: a = 1/L.0; a
(x/(x^4 + 1))*y^4 - 2*x^2/(x^4 + 1)
```

We convert it to the vector space, and get a vector over the base field:

```
sage: to_V(a)
(-2*x^2/(x^4 + 1), 0, 0, 0, x/(x^4 + 1))
```

We convert to and back, and get the same element:

```
sage: from_V(to_V(a)) == a
True
```

In the other direction:

```
sage: v = x*V.0 + (1/x)*V.1
sage: to_V(from_V(v)) == v
True
```

And we show how it works over an extension of an extension field:

```
sage: R2.<z> = L[]; M.<z> = L.extension(z^2 -y)
sage: M.vector_space()
(Vector space of dimension 2 over Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x, Isomorphism:
  From: Vector space of dimension 2 over Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x
  To:  Function field in z defined by z^2 - y, Isomorphism:
    From: Function field in z defined by z^2 - y
    To:  Vector space of dimension 2 over Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x)
```

We can also get the vector space of $M$ over $K$:

```
sage: M.vector_space(K)
(Vector space of dimension 10 over Rational function field in x over Rational Field, Isomorphism:
  From: Vector space of dimension 10 over Rational function field in x over Rational Field
  To:  Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x, Isomorphism:
    From: Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x
    To:  Vector space of dimension 10 over Rational function field in x over Rational Field)
```
class sage.rings.function_field.function_field.RationalFunctionField(constant_field, names, category=None)

Rational function field in one variable, over an arbitrary base field.

INPUT:

• constant_field – arbitrary field
• names – string or tuple of length 1

EXAMPLES:

```python
sage: K.<t> = FunctionField(GF(3)); K
Rational function field in t over Finite Field of size 3
sage: K.gen()
t
sage: 1/t + t^3 + 5
(t^4 + 2*t + 1)/t
```

There are various ways to get at the underlying fields and rings associated to a rational function field:

```python
sage: K.<t> = FunctionField(QQ)
sage: L = FunctionField(QQ, 'tbar')  # give variable name as second input
sage: K.hom(L.gen())
Function Field morphism:
  From: Rational function field in t over Rational Field
  To:   Rational function field in tbar over Rational Field
  Defn: t |--> tbar
```

Element

alias of sage.rings.function_field.element.FunctionFieldElement_rational

base_field()

Return the base field of the rational function field, which is just the function field itself.

EXAMPLES:
change_variable_name(name)

Return a field isomorphic to this field with variable name.

INPUT:

• name – a string or a tuple consisting of a single string, the name of the new variable

OUTPUT:

A triple $F, f, t$ where $F$ is a rational function field, $f$ is an isomorphism from $F$ to this field, and $t$ is the inverse of $f$.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ)
sage: L,f,t = K.change_variable_name('y')
sage: L,f,t
(Rational function field in y over Rational Field,
 Function Field morphism:
   From: Rational function field in y over Rational Field
   To:   Rational function field in x over Rational Field
   Defn: y |--> x,
 Function Field morphism:
   From: Rational function field in x over Rational Field
   To:   Rational function field in y over Rational Field
   Defn: x |--> y)
sage: L.change_variable_name('x')[0] is K
True
```

constant_base_field()

Return the field of which the rational function field is a transcendental extension.

EXAMPLES:

```python
sage: K.<t> = FunctionField(QQ)
sage: K.constant_base_field()
Rational Field
```

constant_field()

Return the field of which the rational function field is a transcendental extension.

EXAMPLES:

```python
sage: K.<t> = FunctionField(QQ)
sage: K.constant_field()
Rational Field
```

degree(base=None)

Return the degree over the base field of the rational function field. Since the base field is the rational function field itself, the degree is 1.

INPUT:

• base – the base field of the vector space; must be the function field itself (the default)

EXAMPLES:
derivation()  
Return a derivation of the rational function field over the constant base field.  
The derivation maps the generator of the rational function field to 1.  

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(3))
sage: m = K.derivation(); m
Derivation map:
    From: Rational function field in x over Finite Field of size 3
    To:   Rational function field in x over Finite Field of size 3
        Defn: x |--> 1
sage: m(x)
1
```

different()  
Return the different of the rational function field.  
For a rational function field, the different is simply the zero divisor.  

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ)
sage: K.different()
0
```

equation_order()  
Return the maximal order of the function field.  
Since this is a rational function field it is of the form $K(t)$, and the maximal order is by definition $K[t]$, where $K$ is the constant field.  

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ)
sage: K.maximal_order()
Maximal order of Rational function field in t over Rational Field
sage: K.equation_order()
Maximal order of Rational function field in t over Rational Field
```

equation_order_infinite()  
Return the maximal infinite order of the function field.  
By definition, this is the valuation ring of the degree valuation of the rational function field.  

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ)
sage: K.maximal_order_infinite()
Maximal infinite order of Rational function field in t over Rational Field
sage: K.equation_order_infinite()
Maximal infinite order of Rational function field in t over Rational Field
```

extension$(f, names=None)$  
Create an extension $L = K[y]/(f(y))$ of the rational function field.
INPUT:
- \( f \) – univariate polynomial over self
- \( \text{names} \) – string or length-1 tuple

OUTPUT:
- a function field

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: K.extension(y^5 - x^3 - 3*x + x*y)
Function field in y defined by y^5 + x*y - x^3 - 3*x
```

A nonintegral defining polynomial:

```
sage: K.<t> = FunctionField(QQ); R.<y> = K[]
sage: K.extension(y^3 + (1/t)*y + t^3/(t+1))
Function field in y defined by y^3 + 1/t*y + t^3/(t + 1)
```

The defining polynomial need not be monic or integral:

```
sage: K.extension(t*y^3 + (1/t)*y + t^3/(t+1))
Function field in y defined by t*y^3 + 1/t*y + t^3/(t + 1)
```

**field()**
Return the underlying field, forgetting the function field structure.

EXAMPLES:

```
sage: K.<t> = FunctionField(GF(7))
sage: K.field()
Fraction Field of Univariate Polynomial Ring in t over Finite Field of size 7
```

See also:

`sage.rings.fraction_field.FractionField_1poly_field.function_field()`

**gen (n=0)**
Return the \( n \)-th generator of the function field. If \( n \) is not 0, then an IndexError is raised.

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ); K.gen()
t
sage: K.gen().parent()
Rational function field in t over Rational Field
sage: K.gen(1)
Traceback (most recent call last):
...
IndexError: Only one generator.
```

**genus ()**
Return the genus of the function field, namely 0.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: K.genus()
0
```
\textbf{hom}(\textit{im\_gens}, \textit{base\_morphism}=\text{None})

Create a homomorphism from \texttt{self} to another ring.

\textbf{INPUT}:

- \textit{im\_gens} – exactly one element of some ring. It must be invertible and transcendental over the image of \textit{base\_morphism}; this is not checked.

- \textit{base\_morphism} – a homomorphism from the base field into the other ring. If \text{None}, try to use a coercion map.

\textbf{OUTPUT}:

- a map between function fields

\textbf{EXAMPLES}:

We make a map from a rational function field to itself:

\begin{verbatim}
sage: K.<x> = FunctionField(GF(7))
sage: K.hom( (x^4 + 2)/x)
Function Field endomorphism of Rational function field in x over Finite Field of size 7
  Defn: x |--> (x^4 + 2)/x
\end{verbatim}

We construct a map from a rational function field into a non-rational extension field:

\begin{verbatim}
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^3 + 6*x^3 + x)
sage: f = K.hom(y^2 + y + 2); f
Function Field morphism:
  From: Rational function field in x over Finite Field of size 7
  To:   Function field in y defined by y^3 + 6*x^3 + x
  Defn: x |--> y^2 + y + 2
sage: f(x)
y^2 + y + 2
sage: f(x^2)
5*y^2 + (x^3 + 6*x + 4)*y + 2*x^3 + 5*x + 4
\end{verbatim}

\textbf{maximal\_order()} 

Return the maximal order of the function field.

Since this is a rational function field it is of the form \(K(t)\), and the maximal order is by definition \(K[t]\), where \(K\) is the constant field.

\textbf{EXAMPLES}:

\begin{verbatim}
sage: K.<t> = FunctionField(QQ)
sage: K.maximal_order()
Maximal order of Rational function field in t over Rational Field
sage: K.equation_order()
Maximal order of Rational function field in t over Rational Field
\end{verbatim}

\textbf{maximal\_order\_infinite()} 

Return the maximal infinite order of the function field.

By definition, this is the valuation ring of the degree valuation of the rational function field.

\textbf{EXAMPLES}:
sage: K.<t> = FunctionField(QQ)
sage: K.maximal_order_infinite()
Maximal infinite order of Rational function field in t over Rational Field

sage: K.equation_order_infinite()
Maximal infinite order of Rational function field in t over Rational Field

ngens()
Return the number of generators, which is 1.

EXAMPLES:
sage: K.<t> = FunctionField(QQ)
sage: K.ngens()
1

place_set()
Return the set of all places of the function field.

EXAMPLES:
sage: K.<t> = FunctionField(GF(7))
sage: K.place_set()
Set of places of Rational function field in t over Finite Field of size 7

polynomial_ring(var='x')
Return a polynomial ring in one variable over the rational function field.

INPUT:
- var – string; name of the variable

EXAMPLES:
sage: K.<x> = FunctionField(QQ)
sage: K.polynomial_ring()
Univariate Polynomial Ring in x over Rational function field in x over Rational Field
sage: K.polynomial_ring('T')
Univariate Polynomial Ring in T over Rational function field in x over Rational Field

random_element(*args, **kwds)
Create a random element of the rational function field.

Parameters are passed to the random_element method of the underlying fraction field.

EXAMPLES:
sage: FunctionField(QQ, 'alpha').random_element()  # random
(-1/2*alpha^2 - 4)/(-12*alpha^2 + 1/2*alpha - 1/95)

vector_space(base=None)
Return a vector space $V$ and isomorphisms from the field to $V$ and from $V$ to the field.

This function allows us to identify the elements of this field with elements of a one-dimensional vector space over the field itself. This method exists so that all function fields (rational or not) have the same interface.

INPUT:
- base – the base field of the vector space; must be the function field itself (the default)
• a vector space $V$ over base field
• an isomorphism from $V$ to the field
• the inverse isomorphism from the field to $V$

EXAMPLES:

```sage
code
sage: K.<x> = FunctionField(QQ)
sage: K.vector_space()
(Vector space of dimension 1 over Rational function field in x over Rational
Field, Isomorphism:
From: Vector space of dimension 1 over Rational function field in x over
Rational Field
To: Rational function field in x over Rational Field,
Isomorphism:
From: Rational function field in x over Rational Field
To: Vector space of dimension 1 over Rational function field in x over
Rational Field)
```

class sage.rings.function_field.function_field.RationalFunctionField_global(constant_field, names, category=None)

Rational function field over finite fields.

collection: sage.rings.function_field.function_field.RationalFunctionField

Bases: sage.rings.function_field.function_field.RationalFunctionField

completion(place, name=None, prec=None, gen_name=None)

Return the completion of the function field at the place

INPUT:
• place – place
• name – string; name of the series variable
• prec – positive integer; default precision
• gen_name – string; name of the generator of the residue field; used only when the place is non-rational

EXAMPLES:

```sage
code
sage: K.<x> = FunctionField(GF(2))
sage: p = K.places_finite()[0]; p
Place (x)
sage: m = K.completion(p); m
Completion map:
From: Rational function field in x over Finite Field of size 2
To: Laurent Series Ring in s over Finite Field of size 2
sage: m(1/(x+1))
1 + s + s^2 + s^3 + s^4 + s^5 + s^7 + s^8 + s^9 + s^{10} + s^{11} + s^{12} + s^{13} + s^{14} + s^{15} + s^{16} + s^{17} + s^{18} + s^{19} + O(s^{20})
sage: p = K.place_infinite(); p
Place (1/x)
sage: m = K.completion(p); m
Completion map:
From: Rational function field in x over Finite Field of size 2
(continues on next page)```
higher_derivation()  
Return the higher derivation for the function field.
This is also called the Hasse-Schmidt derivation.

EXAMPLES:

```python
sage: F.<x> = FunctionField(GF(5))
sage: d = F.higher_derivation()
sage: [d(x^5,i) for i in range(10)]
[x^5, 0, 0, 0, 1, 0, 0, 0, 0, 0]
sage: [d(x^7,i) for i in range(10)]
[x^7, 2*x^6, x^5, 0, 0, x^2, 2*x, 1, 0, 0]
```

place_infinite()  
Return the unique place at infinity.

EXAMPLES:

```python
sage: F.<x> = FunctionField(GF(5))
sage: F.place_infinite()
Place (1/x)
```

places(degree=1)  
Return all places of the degree.

INPUT:

- degree – (default: 1) a positive integer

EXAMPLES:

```python
sage: F.<x> = FunctionField(GF(5))
sage: F.places()
[Place (1/x), Place (x), Place (x + 1), Place (x + 2), Place (x + 3), Place (x + 4)]
```

places_finite(degree=1)  
Return the finite places of the degree.

INPUT:

- degree – (default: 1) a positive integer

EXAMPLES:

```python
sage: F.<x> = FunctionField(GF(5))
sage: F.places_finite()
[Place (x), Place (x + 1), Place (x + 2), Place (x + 3), Place (x + 4)]
```

residue_field(place, name=None)  
Return the residue field of the place along with the maps from and to it.
INPUT:

- place – place of the function field
- name – string; name of the generator of the residue field

EXAMPLES:

```
sage: F.<x> = FunctionField(GF(5))
sage: p = F.places_finite(2)[0]
sage: R, fr_R, to_R = F.residue_field(p)
sage: R
Finite Field in z2 of size 5^2
sage: to_R(x) in R
True
```

```
sage: from sage.rings.function_field.function_field import is_FunctionField
sage: is_FunctionField(QQ)
False
sage: is_FunctionField(FunctionField(QQ, 't'))
True
```

```
sage: from sage.rings.function_field.function_field import is_RationalFunctionField
sage: is_RationalFunctionField(QQ)
False
sage: is_RationalFunctionField(FunctionField(QQ, 't'))
True
```

1.1. Global function fields
ELEMENTS OF FUNCTION FIELDS

Sage provides arithmetic with elements of function fields.

EXAMPLES:

Arithmetic with rational functions:

```sage
definitions:
sage: K.<t> = FunctionField(QQ)
sage: f = t - 1
go: g = t^2 - 3
go: h = f^2/g^3
sage: h.valuation(t-1)
2
sage: h.valuation(t)
0
sage: h.valuation(t^2 - 3)
-3
```

Derivatives of elements in separable extensions:

```sage
definitions:
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: (y^3 + x).derivative()
((x^2 + 1)/x^2)*y + (x^4 + x^3 + 1)/x^3
```

The divisor of an element of a global function field:

```sage
definitions:
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: y.divisor()
- Place (1/x, 1/x*y)
- Place (x, x*y)
+ 2*Place (x + 1, x*y)
```

AUTHORS:

- William Stein: initial version
- Robert Bradshaw (2010-05-27): cythonize function field elements
- Julian Rueth (2011-06-28): treat zero correctly
- Maarten Derickx (2011-09-11): added doctests, fixed pickling
- Kwankyu Lee (2017-04-30): added elements for global function fields

```python
class sage.rings.function_field.element.FunctionFieldElement
    Bases: sage.structure.element.FieldElement
```
Abstract base class for function field elements.

EXAMPLES:

```
sage: t = FunctionField(QQ,'t').gen()
sage: isinstance(t, sage.rings.function_field.element.FunctionFieldElement)
True
```

**characteristic_polynomial** (*args, **kwds*)

Return the characteristic polynomial of the element. Give an optional input string to name the variable in the characteristic polynomial.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3); R.<z> = L[]
sage: M.<z> = L.extension(z^3 - y^2*z + x)
sage: x.characteristic_polynomial('W')
W - x
sage: y.characteristic_polynomial('W')
W^2 - x*W + 4*x^3
sage: z.characteristic_polynomial('W')
W^3 + (-x*y + 4*x^3)*W + x
```

**charpoly** (*args, **kwds*)

Return the characteristic polynomial of the element. Give an optional input string to name the variable in the characteristic polynomial.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3); R.<z> = L[]
sage: M.<z> = L.extension(z^3 - y^2*z + x)
sage: x.characteristic_polynomial('W')
W - x
sage: y.characteristic_polynomial('W')
W^2 - x*W + 4*x^3
sage: z.characteristic_polynomial('W')
W^3 + (-x*y + 4*x^3)*W + x
```

**derivative**

Return the derivative of the element.

The derivative is with respect to the generator of the base rational function field, over which the function field is a separable extension.

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ)
sage: f = (t + 1) / (t^2 - 1/3)
sage: f.derivative()
(-t^2 - 2*t - 1/3)/(t^4 - 2/3*t^2 + 1/9)
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: (y^3 + x).derivative()
((x^2 + 1)/x^2)*y + (x^4 + x^3 + 1)/x^3
```

**differential**

Return the differential \(dx\) where \(x\) is the element.
EXAMPLES:

```python
sage: K.<t> = FunctionField(QQ)
sage: f = 1 / t
sage: f.differential()
(-1/t^2) d(t)

sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[

sage: L.<y> = K.extension(Y^2 + Y + x +1/x)
sage: (y^3 + x).differential()
(((x^2 + 1)/x^2)*y + (x^4 + x^3 + 1)/x^3) d(x)
```

`is_integral()`

Determine if the element is integral over the maximal order of the base field.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[

sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: y.is_integral()
True
sage: (y/x).is_integral()
True
sage: (y/x)^2 - (y/x) + 4*x
0
sage: (y/x^2).is_integral()
False
sage: (y/x).minimal_polynomial('W')
W^2 - W + 4*x
```

`matrix(base=None)`

Return the matrix of multiplication by this element, interpreting this element as an element of a vector space over `base`.

INPUT:

- `base` – a function field (default: None), if None, then the matrix is formed over the base field of this function field.

EXAMPLES:

A rational function field:

```python
sage: K.<t> = FunctionField(QQ)
sage: t.matrix()
[t]
sage: (1/(t+1)).matrix()
[1/(t + 1)]
```

Now an example in a nontrivial extension of a rational function field:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[

sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: y.matrix()
[ 0 1]
[-4*x^3 x]
sage: y.matrix().charpoly('Z')
Z^2 - x*Z + 4*x^3
```

An example in a relative extension, where neither function field is rational:
We show that this matrix does indeed work as expected when making a vector space from a function field:

```
sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: V, from_V, to_V = L.vector_space()
sage: y5 = to_V(y^5); y5
((x^4 + 1)/x, 2*x, 0, 0, 0)
sage: y4y = to_V(y^4) * y.matrix(); y4y
((x^4 + 1)/x, 2*x, 0, 0, 0)
sage: y5 == y4y
True
```

minimal_polynomial(*args, **kwds)

Return the minimal polynomial of the element. Give an optional input string to name the variable in the characteristic polynomial.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3); R.<z> = L[]
sage: M.<z> = L.extension(z^3 - y^2*z + x)
sage: x.minimal_polynomial('W')
W - x
sage: y.minimal_polynomial('W')
W^2 - x*W + 4*x^3
sage: z.minimal_polynomial('W')
W^3 + (-x*y + 4*x^3)*W + x
```

minpoly(*args, **kwds)

Return the minimal polynomial of the element. Give an optional input string to name the variable in the characteristic polynomial.
EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3); R.<z> = L[]
sage: x.minimal_polynomial('W')
  W - x
sage: y.minimal_polynomial('W')
  W^2 - x*W + 4*x^3
sage: z.minimal_polynomial('W')
  W^3 + (-x*y + 4*x^3)*W + x
```

**norm()**

Return the norm of the element.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: y.norm()
  4*x^3
```

The norm is relative:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3); R.<z> = L[]
sage: M.<z> = L.extension(z^3 - y^2*z + x)
sage: z.norm()
  -x
sage: z.norm().parent()
  Function field in y defined by y^2 - x*y + 4*x^3
```

**trace()**

Return the trace of the element.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: y.trace()
  x
```

class `sage.rings.function_field.element.FunctionFieldElement_global`

Bases: `sage.rings.function_field.element.FunctionFieldElement_polymod`

Elements of global function fields

**divisor()**

Return the divisor for the element.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: y.divisor()  # Place (1/x, 1/x*y)
  - Place (1/x, 1/x*y)
  - Place (x, x*y)
  + 2*Place (x + 1, x*y)
```
divisor_of_poles()
Return the divisor of poles for the element.

EXAMPLES:
```
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: (x/y).divisor_of_poles()
Place (1/x, 1/x*y) + 2*Place (x + 1, x*y)
```

divisor_of_zeros()
Return divisor of zeros for the element.

EXAMPLES:
```
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: (x/y).divisor_of_zeros()
3*Place (x, x*y)
```

higher_derivative(i, separating_element=None)
Return the \(i\)-th order higher derivative of the element with respect to the separating element.

INPUT:
- \(i\) – nonnegative integer
- separating_element – separating element of the function field; the default is the generator of the base rational function field

EXAMPLES:
```
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: (y^3 + x).higher_derivative(2)
1/x^3*y + (x^6 + x^4 + x^3 + x^2 + x + 1)/x^5
```

valuation(place)
Return the valuation of the element at the place.

INPUT:
- place – a place of the function field

EXAMPLES:
```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_infinite()[0]
sage: y.valuation(p)
-1
```

class sage.rings.function_field.element.FunctionFieldElement_polymod
Bases: sage.rings.function_field.element.FunctionFieldElement

Elements of a finite extension of a function field.

EXAMPLES:
```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
```
element()

Return the underlying polynomial that represents the element.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); R.<T> = K[]
sage: L.<y> = K.extension(T^2 - x*T + 4*x^3)
sage: f = y/x^2 + x/(x^2+1); f
t^2*y + t/(t^2 + 1)
sage: f.element()
t^2*y + t/(t^2 + 1)
```

list()

Return the list of the coefficients representing the element.

If the function field is \( K[y]/(f(y)) \), then return the coefficients of the reduced presentation of the element as a polynomial in \( K[y] \).

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: a = ~(2*y + 1/x); a
(-1/8*x^2/(x^5 + 1/8*x^2 + 1/16))*y + (1/8*x^3 + 1/16*x)/(x^5 + 1/8*x^2 + 1/16)
sage: a.list()
[(1/8*x^3 + 1/16*x)/(x^5 + 1/8*x^2 + 1/16), -1/8*x^2/(x^5 + 1/8*x^2 + 1/16)]
sage: (x*y).list()
[0, x]
```

class sage.rings.function_field.element.FunctionFieldElement_rational

Bases: sage.rings.function_field.element.FunctionFieldElement

Elements of a rational function field.

EXAMPLES:

```python
sage: K.<t> = FunctionField(QQ); K
Rational function field in t over Rational Field
sage: t^2 + 3/2*t
t^2 + 3/2*t
sage: FunctionField(QQ,'t').gen()^3
t^3
```

denominator()

Return the denominator of the rational function.

EXAMPLES:

```python
sage: K.<t> = FunctionField(QQ)
sage: f = (t+1) / (t^2 - 1/3); f
(t + 1)/(t^2 - 1/3)
sage: f.denominator()
t^2 - 1/3
```
**divisor()**

Return the divisor of the element.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2))
sage: f = 1/(x^3 + x^2 + x)
sage: f.divisor()
3*Place (1/x)
   - Place (x)
   - Place (x^2 + x + 1)
```

**divisor_of_poles()**

Return the divisor of poles for the element.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2))
sage: f = 1/(x^3 + x^2 + x)
sage: f.divisor_of_poles()
Place (x)
   + Place (x^2 + x + 1)
```

**divisor_of_zeros()**

Return the divisor of zeros for the element.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2))
sage: f = 1/(x^3 + x^2 + x)
sage: f.divisor_of_zeros()
3*Place (1/x)
```

**element()**

Return the underlying fraction field element that represents the element.

**EXAMPLES:**

```python
sage: K.<t> = FunctionField(GF(7))
sage: t.element()
t
sage: type(t.element())
<... 'sage.rings.fraction_field_FpT.FpTElement'>

sage: K.<t> = FunctionField(GF(131101))
sage: t.element()
t
sage: type(t.element())
<... 'sage.rings.fraction_field_element.FractionFieldElement_1poly_field'>
```

**factor()**

Factor the rational function.

**EXAMPLES:**

```python
sage: K.<t> = FunctionField(QQ)
sage: f = (t+1) / (t^2 - 1/3)
sage: f.factor()
(t + 1) * (t^2 - 1/3)^-1
```

(continues on next page)
.. code::

    sage: (7*f).factor()
    (7) * (t + 1) * (t^2 - 1/3)^-1
    sage: ((7*f).factor()).unit()
    7
    sage: (f^3).factor()
    (t + 1)^3 * (t^2 - 1/3)^-3

**higher_derivative** *(i, separating_element=None)*

Return the *i*-th derivative of the element with respect to the separating element.

**INPUT:**

* i – nonnegative integer

* separating_element – separating element of the function field; the default is the generator of the rational function field

**EXAMPLES:**

.. code::

    sage: K.<t> = FunctionField(GF(2))
    sage: f = t^2
    sage: f.higher_derivative(2)
    1

**inverse_mod** *(I)*

Return an inverse of the element modulo the integral ideal *I*, if *I* and the element together generate the unit ideal.

**EXAMPLES:**

.. code::

    sage: K.<x> = FunctionField(QQ)
    sage: O = K.maximal_order(); I = O.ideal(x^2+1)
    sage: t = O(x+1).inverse_mod(I); t
    -1/2*x + 1/2
    sage: (t*(x+1) - 1) in I
    True

**is_square** *

Return whether the element is a square.

**EXAMPLES:**

.. code::

    sage: K.<t> = FunctionField(QQ)
    sage: t.is_square()
    False
    sage: (t^2/4).is_square()
    True
    sage: f = 9 * (t+1)^6 / (t^2 - 2*t + 1); f.is_square()
    True
    sage: K.<t> = FunctionField(GF(5))
    sage: (-t^2).is_square()
    True
    sage: (-t^2).sqrt()
    2*t

**list** *

Return a list with just the element.
The list represents the element when the rational function field is viewed as a (one-dimensional) vector space over itself.

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ)
sage: t.list()
[t]
```

**numerator()**

Return the numerator of the rational function.

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ)
sage: f = (t+1) / (t^2 - 1/3); f
(t + 1)/(t^2 - 1/3)
sage: f.numerator()
t + 1
```

**sqrt(all=False)**

Return the square root of the rational function.

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ)
sage: f = t^2 - 2 + 1/t^2; f.sqrt()
(t^2 - 1)/t
sage: f = t^2; f.sqrt(all=True)
[t, -t]
```

**valuation(place)**

Return the valuation of the rational function at the place.

Rational function field places are associated with irreducible polynomials.

INPUT:

- place – a place or an irreducible polynomial

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ)
sage: f = (t - 1)^2*(t + 1)/(t^2 - 1/3)^3
sage: f.valuation(t - 1)
2
sage: f.valuation(t)
0
sage: f.valuation(t^2 - 1/3)
-3
sage: K.<x> = FunctionField(GF(2))
sage: p = K.places_finite()[0]
sage: (1/x^2).valuation(p)
-2
```

```
sage.rings.function_field.element.is_FunctionFieldElement(x)
Return True if x is any type of function field element.
```

EXAMPLES:
```python
sage: t = FunctionField(QQ,'t').gen()
sage: sage.rings.function_field.element.is_FunctionFieldElement(t)
True
sage: sage.rings.function_field.element.is_FunctionFieldElement(0)
False
```

```
sage: sage.rings.function_field.element.make_FunctionFieldElement

    make_FunctionFieldElement(parent, element_class, representing_element)

Used for unpickling FunctionFieldElement objects (and subclasses).

EXAMPLES:
```
```python
sage: from sage.rings.function_field.element import make_FunctionFieldElement
sage: K.<x> = FunctionField(QQ)
sage: make_FunctionFieldElement(K, K.element_class, (x+1)/x)
(x + 1)/x
```
```
ORDERS OF FUNCTION FIELDS

An order of a function field is a subring that is, as a module over the base maximal order, finitely generated and of maximal rank \(n\), where \(n\) is the extension degree of the function field. All orders are subrings of maximal orders.

A rational function field has two maximal orders: maximal finite order \(\mathcal{O}\) and maximal infinite order \(\mathcal{O}_\infty\). The maximal order of a rational function field over constant field \(k\) is just the polynomial ring \(\mathcal{O} = k[x]\). The maximal infinite order is the set of rational functions whose denominator has degree greater than or equal to that of the numerator.

EXAMPLES:

```sage
sage: K.<x> = FunctionField(QQ)
sage: O = K.maximal_order()
sage: I = O.ideal(1/x); I
Ideal (1/x) of Maximal order of Rational function field in x over Rational Field
sage: 1/x in O
False
sage: Oinf = K.maximal_order_infinite()
sage: 1/x in Oinf
True
```

In an extension of a rational function field, an order over the maximal finite order is called a finite order while an order over the maximal infinite order is called an infinite order. Thus a function field has one maximal finite order \(\mathcal{O}\) and one maximal infinite order \(\mathcal{O}_\infty\). There are other non-maximal orders such as equation orders:

```sage
sage: K.<x> = FunctionField(GF(3)); R.<y> = K[]
sage: L.<y> = K.extension(y^3-y-x)
sage: O = L.equation_order()
sage: 1/y in O
False
sage: x/y in O
True
```

Sage provides an extensive functionality for computations in maximal orders of global function fields. For example, you can decompose a prime ideal of a rational function field in an extension:

```sage
sage: K.<x> = FunctionField(GF(2)); _.<t> = K[]
sage: o = K.maximal_order()
sage: p = o.ideal(x+1)
sage: p.is_prime()
True
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: O = F.maximal_order()
sage: O.decomposition(p)
[(Ideal (x + 1, y + 1) of Maximal order
```

(continues on next page)
of Function field in y defined by y^3 + x^6 + x^4 + x^2, 1, 1),
(Ideal (x + 1, y^2 + y + 1) of Maximal order
of Function field in y defined by y^3 + x^6 + x^4 + x^2, 2, 1])

sage: p1,relative_degree,ramification_index = O.decomposition(p)[1]
sage: p1.parent()
Monoid of ideals of Maximal order of Function field in y
defined by y^3 + x^6 + x^4 + x^2
sage: relative_degree
2
sage: ramification_index
1

AUTHORS:

- William Stein (2010): initial version
- Maarten Derickx (2011-09-14): fixed ideal_with_gens_over_base() for rational function fields
- Julian Rueth (2011-09-14): added check in _element_constructor_
- Kwankyu Lee (2017-04-30): added maximal orders of global function fields

class sage.rings.function_field.order.FunctionFieldMaximalOrder

Base class of maximal orders of function fields.

class sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite

Base class of maximal infinite orders of function fields.

class sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_global

Maximal infinite orders of global function fields.

INPUT:

- field – function field

EXAMPLES:

sage: K.<x> = FunctionField(GF(2)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: F.maximal_order_infinite()
Maximal infinite order of Function field in y defined by y^3 + x^6 + x^4 + x^2

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: L.maximal_order_infinite()
Maximal infinite order of Function field in y defined by y^2 + y + (x^2 + 1)/x

basis()

Return a basis of this order as a module over the maximal order of the base function field.

EXAMPLES:

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: Oinf = L.maximal_order_infinite()
sage: Oinf.basis()
(1, 1/x^2*y, (1/(x^4 + x^3 + x^2))*y^2)

coordinate_vector(e)

Return the coordinates of \(e\) with respect to the basis of the order.

INPUT:

- \(e\) – element of the function field

The returned coordinates are in the base maximal infinite order if and only if the element is in the order.

EXAMPLES:

sage: K.<x> = FunctionField(GF(2)); _.<t> = K[]
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: Oinf = F.maximal_order_infinite()
sage: f = 1/y^2
sage: f in Oinf
True
sage: Oinf.coordinate_vector(f)
((x^3 + x^2 + x)/(x^4 + 1), x^3/(x^4 + 1))

decomposition()

Return prime ideal decomposition of \(pO_{\infty}\) where \(p\) is the unique prime ideal of the maximal infinite order of the rational function field.

EXAMPLES:

sage: K.<x> = FunctionField(GF(2)); _.<t> = K[]
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: Oinf = F.maximal_order_infinite()
sage: Oinf.decomposition()
[(Ideal (1/x,1/x^2*y + 1) of Maximal infinite order of Function field in y defined by y^3 + x^6 + x^4 + x^2, 1, 1),
 (Ideal (1/x,1/x^4*y^2 + 1/x^2*y + 1) of Maximal infinite order of Function field in y defined by y^2 + y + (x^2 + 1)/x, 1, 1)
of Function field in y defined by $y^3 + x^6 + x^4 + x^2, 2, 1$)

```
  sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[
  sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
  sage: Oinf = L.maximal_order_infinite()
  sage: Oinf.decomposition()
  [(Ideal (1/x,1/x*y) of Maximal infinite order of Function field in y
defined by $y^2 + y + (x^2 + 1)/x, 1, 2)]]
```

different()
Return the different ideal of the maximal infinite order.

**EXAMPLES:**

```
  sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[
  sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
  sage: Oinf = L.maximal_order_infinite()
  sage: Oinf.different()
  Ideal (1/x) of Maximal infinite order of Function field in y
defined by $y^2 + y + (x^2 + 1)/x$
```

gen(n=0)
Return the n-th generator of the order.
The basis elements of the order are generators.

**EXAMPLES:**

```
  sage: K.<x> = FunctionField(GF(2)); _.<t> = K[
  sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
  sage: Oinf = F.maximal_order_infinite()
  sage: Oinf.gen()
  1
  sage: Oinf.gen(1)
  1/x^2*y
  sage: Oinf.gen(2)
  (1/(x^4 + x^3 + x^2))*y^2
  sage: Oinf.gen(3)
  Traceback (most recent call last):
  ...  
  IndexError: there are only 3 generators
```

ideal(*gens)
Return the ideal generated by gens.

**INPUT:**

- gens – tuple of elements of the function field

**EXAMPLES:**

```
  sage: K.<x> = FunctionField(GF(2)); _.<t> = K[
  sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
  sage: Oinf = F.maximal_order_infinite()
  sage: I = Oinf.ideal(x,y); I
  Ideal (x^2) of Maximal infinite order of Function field in y
defined by $y^3 + x^6 + x^4 + x^2$
```

(continues on next page)
```python
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: Oinf = L.maximal_order_infinite()
sage: I = Oinf.ideal(x,y); I
Ideal (x) of Maximal infinite order of Function field in y defined by y^2 + y +
˓→ (x^2 + 1)/x

ideal_with_gens_over_base(gens)

Return the ideal generated by gens as a module.

INPUT:

- gens – tuple of elements of the function field

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2)); R.<t> = K[
```  
```python
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: Oinf = F.maximal_order_infinite()
sage: Oinf.ideal_with_gens_over_base((x^2, y, (1/(x^2 + x + 1))*y^2))
Ideal (x^2) of Maximal infinite order of Function field in y defined by y^3 + x^6 + x^4 + x^2
```  

ngens()

Return the number of generators of the order.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2)); R.<t> = K[
```  
```python
sage: L.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: Oinf = L.maximal_order_infinite()
sage: Oinf.ngens()
3
```  

```python
class sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_rational(field, cat,
```  
```python
gory=None)
```

Bases: sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite

Maximal infinite orders of rational function fields.

INPUT:

- field – a rational function field

EXAMPLES:

```python
sage: K.<t> = FunctionField(GF(19)); K
Rational function field in t over Finite Field of size 19
sage: R = K.maximal_order_infinite(); R
Maximal infinite order of Rational function field in t over Finite Field of size
˓→ 19
```  

basis()

Return the basis (=1) of the order as a module over the polynomial ring.

EXAMPLES:

```
```python
sage: K.<t> = FunctionField(GF(19))
sage: O = K.maximal_order()
sage: O.basis()
(1,)
```

**gen**(\(n=0\))

Return the \(n\)-th generator of self. Since there is only one generator \(n\) must be 0.

**EXAMPLES:**

```python
sage: O = FunctionField(QQ,'y').maximal_order()
sage: O.gen()
y
sage: O.gen(1)
Traceback (most recent call last):
  ... IndexError: there is only one generator
```

**ideal**(\(*gens*)

Return the fractional ideal generated by \(gens\).

**INPUT:**

\- \(gens\) – elements of the function field

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(QQ)
sage: O = K.maximal_order_infinite()
sage: O.ideal(x)
Ideal (x) of Maximal infinite order of Rational function field in x over Rational Field
sage: O.ideal([x,1/x]) == O.ideal(x,1/x) # multiple generators may be given as a list
True
sage: O.ideal(x^3+1,x^3+6)
Ideal (x^3) of Maximal infinite order of Rational function field in x over Rational Field
sage: I = O.ideal((x^2+1)*(x^3+1),(x^3+6)*(x^2+1)); I
Ideal (x^5) of Maximal infinite order of Rational function field in x over Rational Field
sage: O.ideal(I)
Ideal (x^5) of Maximal infinite order of Rational function field in x over Rational Field
```

**ngens**()

Return 1 the number of generators of the order.

**EXAMPLES:**

```python
sage: FunctionField(QQ,'y').maximal_order().ngens()
1
```

**prime_ideal**()

Return the unique prime ideal of the order.

**EXAMPLES:**

```python
sage: 
```
```python
sage: K.<t> = FunctionField(GF(19))
sage: O = K.maximal_order_infinite()
sage: O.prime_ideal()
Ideal (1/t) of Maximal infinite order of Rational function field in t over Finite Field of size 19
```

```python
class sage.rings.function_field.order.FunctionFieldMaximalOrder_global(field)
Bases: sage.rings.function_field.order.FunctionFieldMaximalOrder

Maximal orders of global function fields.

INPUT:

• field – function field to which this maximal order belongs

EXAMPLES:
```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: L.maximal_order()
Maximal order of Function field in y defined by y^4 + x*y + 4*x + 1
```

**basis()**

Return a basis of the order over the maximal order of the base function field.

EXAMPLES:
```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.equation_order()
sage: O.basis()
(1, y, y^2, y^3)
```

**codifferent()**

Return the codifferent ideal of the function field.

EXAMPLES:
```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.maximal_order()
sage: O.codifferent()
Ideal (1, (1/(x^4 + 4*x^3 + 3*x^2 + 6*x + 4))*y^3 + ((5*x^3 + 6*x^2 + x + 6)/(x^4 + 4*x^3 + 3*x^2 + 6*x + 4))*y^2 + (x^3 + 2*x^2 + 2*x + 2)/(x^4 + 4*x^3 + 3*x^2 + 6*x + 4))*y + 6*x/(x^4 + 4*x^3 + 3*x^2 + 6*x + 4)) of Maximal order of Function field in y defined by y^4 + x*y + 4*x + 1
```

**coordinate_vector()**

Return the coordinates of \( e \) with respect to the basis of this order.

EXAMPLES:
```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.maximal_order()
sage: O.coordinate_vector(y)
(0, 1, 0, 0)
sage: O.coordinate_vector(x*y)
(0, x, 0, 0)
```
decomposition(ideal)

Return the decomposition of the prime ideal.

INPUT:

- ideal – prime ideal of the base maximal order

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); R.<t> = K[

sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)

sage: o = K.maximal_order()

sage: O = F.maximal_order()

sage: p = o.ideal(x+1)

sage: O.decomposition(p)

[(Ideal (x + 1, y + 1) of Maximal order
  of Function field in y defined by y^3 + x^6 + x^4 + x^2, 1, 1),
  (Ideal (x + 1, y^2 + y + 1) of Maximal order
  of Function field in y defined by y^3 + x^6 + x^4 + x^2, 2, 1)]
```

different()

Return the different ideal of the function field.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[

sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)

sage: O = L.maximal_order()

sage: O.different()

Ideal (x^4 + 4*x^3 + 3*x^2 + 6*x + 4, y + 2*x^3 + x^2 + 6*x + 1)
  of Maximal order of Function field in y defined by y^4 + x*y + 4*x + 1
```

free_module()

Return the free module formed by the basis over the maximal order of the base field.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[

sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)

sage: O = L.maximal_order()

sage: O.free_module()

Free module of degree 4 and rank 4 over Maximal order of Rational function
  field in x over Finite Field of size 7

User basis matrix:
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

gen(n=0)

Return the n-th generator of the order.

The basis elements of the order are generators.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); R.<t> = K[

sage: L.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)

sage: O = L.maximal_order()

sage: O.gen()

(continues on next page)```
ideal(*gens, **kwargs)
Return the fractional ideal generated by the elements in gens.

INPUT:

• gens – list of generators

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[
    sage: O = K.maximal_order()
    sage: I = O.ideal(x^2-4)
    sage: L.<y> = K.extension(y^2 - x^3 - 1)
    sage: S = L.maximal_order()
    sage: S.ideal(1/y)
    Ideal (1, (1/(x^3 + 1))*y) of Maximal order of Function field in y defined by y^2 + 6*x^3 + 6
    sage: I2 = S.ideal(x^2-4); I2
    Ideal (x^2 + 3) of Maximal order of Function field in y defined by y^2 + 6*x^3 + 6
    sage: I2 == S.ideal(I)
    True
```
```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[

sage: L.<y> = K.extension(y^2 - x^3 - 1)

sage: O = L.equation_order()

sage: I = O.ideal_with_gens_over_base([y]); I
Ideal (y) of Order in Function field in y defined by y^2 + 6*x^3 + 6

sage: y in I
True

sage: y^2 in I
False
```

def `ngens()`:
   Return the number of generators of the order.

   EXAMPLES:
```
sage: K.<x> = FunctionField(GF(2)); _.<t> = K[

sage: L.<y> = K.extension(t^3 - x^2 * (x^2 + x + 1)^2)

sage: Oinf = L.maximal_order()

sage: Oinf.ngens()
3
```

def `p_radical(prime)`:
   Return the prime-radical of the maximal order.

   INPUT:
   
   - `prime` -- prime ideal of the maximal order of the base rational function field

   The algorithm is outlined in Section 6.1.3 of [Coh1993].

   EXAMPLES:
```
sage: K.<x> = FunctionField(GF(2)); _.<t> = K[

sage: F.<y> = K.extension(t^3 - x^2 * (x^2 + x + 1)^2)

sage: o = K.maximal_order()

sage: p = o.ideal(x+1)

sage: O.p_radical(p)
Ideal (x + 1) of Maximal order of Function field in y
defined by y^3 + x^6 + x^4 + x^2
```

def `polynomial()`:
   Return the defining polynomial of the function field of which this is an order.

   EXAMPLES:
```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[

sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)

sage: O = L.equation_order()

sage: O.polynomial()
y^4 + x*y + 4*x + 1
```

class `sage.rings.function_field.order.FunctionFieldMaximalOrder_rational(field)`
   Bases: `sage.rings.function_field.order.FunctionFieldMaximalOrder`

   Maximal orders of rational function fields.

   INPUT:
   
   - `field` -- a function field
EXAMPLES:

```python
sage: K.<t> = FunctionField(GF(19)); K
Rational function field in t over Finite Field of size 19
sage: R = K.maximal_order(); R
Maximal order of Rational function field in t over Finite Field of size 19
```

**basis()**

Return the basis (=1) of the order as a module over the polynomial ring.

EXAMPLES:

```python
sage: K.<t> = FunctionField(GF(19))
sage: O = K.maximal_order()
sage: O.basis()
(1,)
```

**gen (n=0)**

Return the n-th generator of the order. Since there is only one generator n must be 0.

EXAMPLES:

```python
sage: O = FunctionField(QQ,'y').maximal_order()
sage: O.gen()
y
sage: O.gen(1)
Traceback (most recent call last):
...  
IndexError: there is only one generator
```

**ideal (**gens**)**

Return the fractional ideal generated by gens.

INPUT:

- gens – elements of the function field

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ)
sage: O = K.maximal_order()
sage: O.ideal(x)
Ideal (x) of Maximal order of Rational function field in x over Rational Field
sage: O.ideal([x,1/x]) == O.ideal(x,1/x) # multiple generators may be given
True
sage: O.ideal(x^3+1,x^3+6)
Ideal (1) of Maximal order of Rational function field in x over Rational Field
sage: I = O.ideal((x^2+1)*(x^3+1),(x^3+6)*(x^2+1)); I
Ideal (x^2 + 1) of Maximal order of Rational function field in x over Rational Field
sage: O.ideal(I)
Ideal (x^2 + 1) of Maximal order of Rational function field in x over Rational Field
```

**ideal_with_gens_over_base (gens)**

Return the fractional ideal with generators gens.

INPUT:

- gens – elements of the function field
EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order()
sage: O.ideal_with_gens_over_base([x^3+1,-y])
Ideal (x^3 + 1, -y) of Order in Function field in y defined by y^2 - x^3 - 1
```

ngens ()

Return 1 the number of generators of the order.

EXAMPLES:

```
sage: FunctionField(QQ,'y').maximal_order().ngens()
1
```

```
class sage.rings.function_field.order.FunctionFieldOrder (field, ideal_class=<class 'sage.rings.function_field.ideal.FunctionFieldIdeal'>, category=None)
Bases: sage.rings.function_field.order.FunctionFieldOrder_base
Base class for orders in function fields.

class sage.rings.function_field.order.FunctionFieldOrderInfinite (field, ideal_class=<class 'sage.rings.function_field.ideal.FunctionFieldIdeal'>, category=None)
Bases: sage.rings.function_field.order.FunctionFieldOrder_base
Base class for infinite orders in function fields.

class sage.rings.function_field.order.FunctionFieldOrderInfinite_basis (basis, check=True)
Bases: sage.rings.function_field.order.FunctionFieldOrderInfinite
Order given by a basis over the infinite maximal order of the base field.

INPUT:

• basis – elements of the function field
• check – boolean (default: True); if True, check the basis generates an order

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.order_infinite_with_basis([1, y, 1/x^2*y^2, y^3]); O
Infinite order in Function field in y defined by y^4 + x*y + 4*x + 1
```

The basis only defines an order if the module it generates is closed under multiplication and contains the identity element (only checked when check is True):

```
sage: 0 = L.order_infinite_with_basis([1, y, 1/x^2*y^2, y^3]); O
Traceback (most recent call last):
  ... ValueError: the module generated by basis (1, y, 1/x^2*y^2, y^3) must be closed under multiplication
```

The basis also has to be linearly independent and of the same rank as the degree of the function field of its elements (only checked when check is True):
```python
sage: O = L.order_infinite_with_basis([1, y, 1/x^2*y^2, 1 + y]); O
Traceback (most recent call last):
...
ValueError: The given basis vectors must be linearly independent.
```

Note that 1 does not need to be an element of the basis, as long as it is in the module spanned by it:

```python
sage: O = L.order_infinite_with_basis([1 + 1/x*y, 1/x*y, 1/x^2*y^2, 1/x^3*y^3]); O
Infinite order in Function field in y defined by y^4 + x*y + 4*x + 1
sage: O.basis()
(1/x*y + 1, 1/x*y, 1/x^2*y^2, 1/x^3*y^3)
```

**basis()**

Return a basis of this order over the maximal order of the base field.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[

sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)

sage: O = L.equation_order()

sage: O.basis()
(1, y, y^2, y^3)
```

**free_module()**

Return the free module formed by the basis over the maximal order of the base field.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[

sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)

sage: O = L.equation_order()

sage: O.free_module()
Free module of degree 4 and rank 4 over Maximal order of Rational
function field in x over Finite Field of size 7
Echelon basis matrix:
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

**ideal(*gens)**

Return the fractional ideal generated by the elements in *gens*.

**INPUT:**

- *gens* – list of generators or an ideal in a ring which coerces to this order

**EXAMPLES:**

```python
sage: K.<y> = FunctionField(QQ)

sage: O = K.maximal_order()

sage: O.ideal(y)
Ideal (y) of Maximal order of Rational function field in y over Rational Field
sage: O.ideal([y,1/y]) == O.ideal(y,1/y)  # multiple generators may be given
˓→as a list
True
```

A fractional ideal of a nontrivial extension:
ideal_with_gens_over_base(gens)
Return the fractional ideal with basis gens over the maximal order of the base field.

It is not checked that gens really generates an ideal.

INPUT:

• gens – list of elements that are a basis for the ideal over the maximal order of the base field

EXAMPLES:

We construct an ideal in a rational function field:

```
sage: K.<y> = FunctionField(QQ)
sage: O = K.maximal_order()
sage: I = O.ideal([y]); I
Ideal (y) of Maximal order of Rational function field in y over Rational Field
sage: I*I
Ideal (y^2) of Maximal order of Rational function field in y over Rational Field
```

We construct some ideals in a nontrivial function field:

```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[
 sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order(); O
Order in Function field in y defined by y^2 + 6*x^3 + 6
sage: I = O.ideal_with_gens_over_base([1, y]); I
Ideal (1, y) of Order in Function field in y defined by y^2 + 6*x^3 + 6
sage: I.module()
Free module of degree 2 and rank 2 over Maximal order of Rational function field in x over Finite Field of size 7
Echelon basis matrix:
[1 0]
[0 1]
```

There is no check if the resulting object is really an ideal:

```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[
 sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order()
sage: I = O.ideal_with_gens_over_base([y]); I
Ideal (y) of Order in Function field in y defined by y^2 + 6*x^3 + 6
sage: y in I
True
sage: y^2 in I
False
```

polynomial()
Return the defining polynomial of the function field of which this is an order.

EXAMPLES:
```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)

class sage.rings.function_field.order.FunctionFieldOrder_base(field, ideal_class=<class
'sage.rings.function_field.ideal.FunctionFieldIdeal'>, category=None)

structure.parent.Parent

Base class for orders in function fields.

INPUT:

- field -- function field

EXAMPLES:

```python
sage: F = FunctionField(QQ,'y')
sage: F.maximal_order()
Maximal order of Rational function field in y over Rational Field

fraction_field()

Return the function field to which the order belongs.

EXAMPLES:

```python
sage: FunctionField(QQ,'y').maximal_order().fraction_field()
Rational function field in y over Rational Field

function_field()

Return the function field to which the order belongs.

EXAMPLES:

```python
sage: FunctionField(QQ,'y').maximal_order().function_field()
Rational function field in y over Rational Field

ideal_monoid()

Return the monoid of ideals of the order.

EXAMPLES:

```python
sage: FunctionField(QQ,'y').maximal_order().ideal_monoid()
Monoid of ideals of Maximal order of Rational function field in y over
Rational Field

is_field()

Return False since orders are never fields.

EXAMPLES:

```python
sage: FunctionField(QQ,'y').maximal_order().is_field()
False

is_noetherian()

Return True since orders in function fields are noetherian.
```
EXAMPLES:

```python
sage: FunctionField(QQ,'y').maximal_order().is_noetherian()
True
```

**is_subring** *(other)*

Return `True` if the order is a subring of the other order.

**INPUT:**

- `other` – order of the function field or the field itself

**EXAMPLES:**

```python
sage: F = FunctionField(QQ,'y')
sage: O = F.maximal_order()
sage: O.is_subring(F)
True
```

class `sage.rings.function_field.order.FunctionFieldOrder_basis` *(basis, check=True)*

Order given by a basis over the maximal order of the base field.

**INPUT:**

- `basis` – list of elements of the function field
- `check` – (default: `True`) if `True`, check whether the module that `basis` generates forms an order

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.equation_order(); O
Order in Function field in y defined by y^4 + x*y + 4*x + 1
```

The basis only defines an order if the module it generates is closed under multiplication and contains the identity element:

```python
sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^5 - (x^3 + 2*x*y + 1/x))
sage: y.is_integral()
False
sage: L.order(y)
Traceback (most recent call last):
  ... ValueError: the module generated by basis (1, y, y^2, y^3, y^4) must be closed under multiplication
```

The basis also has to be linearly independent and of the same rank as the degree of the function field of its elements (only checked when `check` is `True`):

```python
sage: L.order(L(x))
Traceback (most recent call last):
  ... ValueError: basis (1, x, x^2, x^3, x^4) is not linearly independent
sage: sage.rings.function_field.order.FunctionFieldOrder_basis((y,y^3,y^4,y^5))
Traceback (most recent call last):
```

(continues on next page)
basis()

Return a basis of the order over the maximal order of the base field.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.equation_order()
sage: O.basis()
(1, y, y^2, y^3)
```

cordinate_vector(e)

Return the coordinates of e with respect to the basis of the order.

INPUT:

- e – element of the order or the function field

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.equation_order()
sage: f = (x + y)^3
sage: O.coordinate_vector(f)
(x^3, 3*x^2, 3*x, 1)
```

free_module()

Return the free module formed by the basis over the maximal order of the base function field.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.equation_order()
sage: O.free_module()
Free module of degree 4 and rank 4 over Maximal order of Rational function field in x over Finite Field of size 7
Echelon basis matrix:
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

ideal(*gens)

Return the fractional ideal generated by the elements in gens.

INPUT:

- gens – list of generators or an ideal in a ring which coerces to this order

EXAMPLES:

```python
sage: K.<y> = FunctionField(QQ)
sage: O = K.maximal_order()
```
A fractional ideal of a nontrivial extension:

\begin{verbatim}
 sage: K.<x> = FunctionField(GF(7)); R.<y> = K[
 sage: L.<y> = K.extension(y^2 - x^3 - 1)
 sage: O = L.equation_order(); O
 Order in Function field in y defined by y^2 + 6*x^3 + 6
 sage: I = O.ideal_with_gens_over_base([1, y]); I
 Ideal (1, y) of Order in Function field in y defined by y^2 + 6*x^3 + 6
 sage: I*I
 Ideal (1, 6*x^3 + 1)*y of Order in Function field in y defined by y^2 + 6*x^3 + 6
 sage: I2 = O.ideal(x^2-4); I2
 Ideal (x^2 + 3, (x^2 + 3)*y) of Order in Function field in y defined by y^2 + 6*x^3 + 6
 sage: I2 == S.ideal(I)
 True
\end{verbatim}

\texttt{ideal_with_gens_over_base}(gens)

Return the fractional ideal with basis \texttt{gens} over the maximal order of the base field.

It is not checked that the \texttt{gens} really generates an ideal.

**INPUT:**

- \texttt{gens} – list of elements of the function field

**EXAMPLES:**

We construct an ideal in a rational function field:

\begin{verbatim}
 sage: K.<x> = FunctionField(QQ);
 sage: R.<y> = K[
 sage: O = K.maximal_order(); O
 Maximal order of Rational function field in y over Rational Field
 sage: I = O.ideal([y]); I
 Ideal (y) of Maximal order of Rational function field in y over Rational Field
 sage: I*I
 Ideal (y^2) of Maximal order of Rational function field in y over Rational Field
\end{verbatim}

We construct some ideals in a nontrivial function field:

\begin{verbatim}
 sage: K.<x> = FunctionField(GF(7)); R.<y> = K[
 sage: L.<y> = K.extension(y^2 - x^3 - 1)
 sage: O = L.equation_order(); O
 Order in Function field in y defined by y^2 + 6*x^3 + 6
 sage: I = O.ideal_with_gens_over_base([1, y]); I
 Ideal (1, y) of Order in Function field in y defined by y^2 + 6*x^3 + 6
 sage: I.module()
 Free module of degree 2 and rank 2 over Maximal order of Rational function field
 Echelon basis matrix:
 [1 0]
 [0 1]
\end{verbatim}

There is no check if the resulting object is really an ideal:

Chapter 3. Orders of function fields
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order()
sage: I = O.ideal_with_gens_over_base([y]); I
Ideal (y) of Order in Function field in y defined by y^2 + 6*x^3 + 6
sage: y in I
True
sage: y^2 in I
False

polynomial()

Return the defining polynomial of the function field of which this is an order.

EXAMPLES:

sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 1)
sage: O = L.equation_order()
sage: O.polynomial()
y^4 + x*y + 4*x + 1
IDEALS OF FUNCTION FIELDS

Ideals of an order of a function field include all fractional ideals of the order. Sage provides basic arithmetic with fractional ideals.

The fractional ideals of the maximal order of a global function field forms a multiplicative monoid. Sage allows advanced arithmetic with the fractional ideals. For example, an ideal of the maximal order can be factored into a product of prime ideals.

EXAMPLES:

Ideals in the maximal order of a rational function field:

```python
sage: K.<x> = FunctionField(QQ)
sage: O = K.maximal_order()
sage: I = O.ideal(x^3 + 1); I
Ideal (x^3 + 1) of Maximal order of Rational function field in x over Rational Field
sage: I^2
Ideal (x^6 + 2*x^3 + 1) of Maximal order of Rational function field in x over Rational Field
sage: ~I
Ideal (1/(x^3 + 1)) of Maximal order of Rational function field in x over Rational Field
sage: ~I * I
Ideal (1) of Maximal order of Rational function field in x over Rational Field
```

Ideals in the equation order of an extension of a rational function field:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order()
sage: I = O.ideal(y); I
Ideal (x^3 + 1, -y) of Order in Function field in y defined by y^2 - x^3 - 1
sage: I^2
Ideal (x^3 + 1, (-x^3 - 1)*y) of Order in Function field in y defined by y^2 - x^3 - 1
```

Ideals in the maximal order of a global function field:

```python
sage: K.<x> = FunctionField(GF(2)); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3*y - x)
sage: O = L.maximal_order()
sage: I = O.ideal(y)
sage: I^2
Ideal (x) of Maximal order of Function field in y defined by y^2 + x^3*y + x
sage: ~I
Ideal (1, 1/x*y) of Maximal order of Function field in y defined by y^2 + x^3*y + x
sage: ~I * I
```

(continues on next page)
Ideal (1) of Maximal order of Function field in y defined by \( y^2 + x^3y + x \)

```python
sage: J = O.ideal(x+y) * I
sage: J.factor()
(Ideal (x, y) of Maximal order of Function field in y defined by \( y^2 + x^3y + x \))^2 *
(Ideal (x^3 + x + 1, y + x) of Maximal order of Function field in y defined by \( y^2 + x^3y + x \))^2
```

Ideals in the maximal infinite order of a global function field:

```python
sage: K.<x> = FunctionField(GF(3^2)); R.<t> = K[]
sage: F.<y> = K.extension(t^3 + t^2 - x^4)
sage: Oinf = F.maximal_order_infinite()
sage: I = Oinf.ideal(1/y)
sage: I + I == I
True
sage: I^2
Ideal (1/x^3,1/x^4*y) of Maximal infinite order of Function field in y defined by \( y^3 + y^2 + 2*x^4 \)

sage: ~I
Ideal (x,y) of Maximal infinite order of Function field in y defined by \( y^3 + y^2 + 2*x^4 \)

sage: ~I * I
Ideal (1) of Maximal infinite order of Function field in y defined by \( y^3 + y^2 + 2*x^4 \)

sage: I.factor()
(Ideal (1/x,1/x^3*y^2) of Maximal infinite order of Function field in y defined by \( y^3 + y^2 + 2*x^4 \))^4
```

AUTHORS:

- William Stein (2010): initial version
- Maarten Derickx (2011-09-14): fixed ideal_with_gens_over_base()
- Kwankyu Lee (2017-04-30): added ideals for global function fields

class sage.rings.function_field.ideal.FunctionFieldIdeal(ring)
Bases: sage.structure.element.Element

Base class of fractional ideals of function fields.

INPUT:

- `ring` – ring of the ideal

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(7))
sage: O = K.equation_order()
sage: O.ideal(x^3+1)
Ideal (x^3 + 1) of Maximal order of Rational function field in x over Finite Field of size 7
```

```
base_ring()
```

Return the base ring of this ideal.

EXAMPLES:
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order()
sage: I = O.ideal(x^2 + 1)
sage: I.base_ring()
Order in Function field in y defined by y^2 - x^3 - 1

divisor()
Return the divisor corresponding to the ideal.

EXAMPLES:

sage: K.<x> = FunctionField(GF(4))
sage: O = K.maximal_order()
sage: I = O.ideal(x*(x + 1)^2/(x^2 + x + 1))
sage: I.divisor()
Place (x) + 2*Place (x + 1) - Place (x + z2) - Place (x + z2 + 1)

sage: Oinf = K.maximal_order_infinite()
sage: I = Oinf.ideal((x + 1)/(x^3 + 1))
sage: I.divisor()
2*Place (1/x)

sage: K.<x> = FunctionField(GF(2)); _.<T> = PolynomialRing(K)
sage: F.<y> = K.extension(T^3 - x^2*(x^2 + x + 1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(y)
sage: I.divisor()
2*Place (x, (1/(x^3 + x^2 + x))*y^2) + 2*Place (x^2 + x + 1, (1/(x^3 + x^2 + x))*y^2)

sage: Oinf = F.maximal_order_infinite()
sage: I = Oinf.ideal(y)
sage: I.divisor()
-2*Place (1/x, 1/x^4*y^2 + 1/x^2*y + 1) - 2*Place (1/x, 1/x^2*y + 1)

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: O = L.maximal_order()
sage: I = O.ideal(y)
sage: I.divisor()
- Place (x, x*y) + 2*Place (x + 1, x*y)

sage: Oinf = L.maximal_order_infinite()
sage: I = Oinf.ideal(y)
sage: I.divisor()
- Place (1/x, 1/x*y)

divisor_of_poles()
Return the divisor of poles corresponding to the ideal.

EXAMPLES:

sage: K.<x> = FunctionField(GF(4))
sage: O = K.maximal_order()
sage: I = O.ideal(x*(x + 1)^2/(x^2 + x + 1))

\textbf{sage: I.divisor_of_poles()}
Place \((x + z2) + \text{Place (}x + z2 + 1\text{)}\)

\textbf{sage: K.<x> = FunctionField(GF(2))}
\textbf{sage: Oinf = K.maximal_order_infinite()}
\textbf{sage: I = Oinf.ideal((x + 1)/(x^3 + 1))}
\textbf{sage: I.divisor_of_poles()}
0

\textbf{sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]}
\textbf{sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)}
\textbf{sage: O = L.maximal_order()}
\textbf{sage: I = O.ideal(y)}
\textbf{sage: I.divisor_of_poles()}
Place \((x, x*y)\)

\textbf{divisor_of_zeros()}
Return the divisor of zeros corresponding to the ideal.

\textbf{EXAMPLES:}

\textbf{sage: K.<x> = FunctionField(GF(4))}
\textbf{sage: O = K.maximal_order()}
\textbf{sage: I = O.ideal(x*(x + 1)^2/(x^2 + x + 1))}
\textbf{sage: I.divisor_of_zeros()}
Place \((x) + 2*\text{Place (}x + 1\text{)}\)

\textbf{sage: K.<x> = FunctionField(GF(2))}
\textbf{sage: Oinf = K.maximal_order_infinite()}
\textbf{sage: I = Oinf.ideal((x + 1)/(x^3 + 1))}
\textbf{sage: I.divisor_of_zeros()}
2*Place \((1/x)\)

\textbf{sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]}
\textbf{sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)}
\textbf{sage: O = L.maximal_order()}
\textbf{sage: I = O.ideal(y)}
\textbf{sage: I.divisor_of_zeros()}
2*Place \((x + 1, x*y)\)

\textbf{factor()}
Return the factorization of this ideal.

Subclass of this class should define \_\texttt{factor()}\ method that returns a list of prime ideal and multiplicity pairs.

\textbf{EXAMPLES:}

\textbf{sage: K.<x> = FunctionField(GF(4))}
\textbf{sage: O = K.maximal_order()}
\textbf{sage: I = O.ideal(x^3*(x + 1)^2)}
\textbf{sage: I.factor()}
(Ideal \((x)\) of \texttt{Maximal order of Rational function field in} \texttt{x over Finite Field in \texttt{z2 of size 2^2})^3 *\n(Ideal \((x + 1)\) of \texttt{Maximal order of Rational function field in} \texttt{x over Finite Field in \texttt{z2 of size 2^2})^2*
Use SageMath:

```python
sage: K.<x> = FunctionField(GF(2)); _.<T> = PolynomialRing(K)
sage: F.<y> = K.extension(T^3 - x^2*(x^2 + x + 1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(y)
sage: I == I.factor().prod()
True
```
\[
sage: p.place()
Place (1/x)
\]
\[
sage: K.<x> = FunctionField(GF(2)); _.<t> = PolynomialRing(K)
\]
\[
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
\]
\[
sage: O = F.maximal_order()
\]
\[
sage: I = O.ideal(y)
\]
\[
sage: [f.place() for f,_ in I.factor()]
[Place (x, (1/(x^3 + x^2 + x))*y^2),
Place (x^2 + x + 1, (1/(x^3 + x^2 + x))*y^2)]
\]
\[
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K
\]
\[
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
\]
\[
sage: O = L.maximal_order()
\]
\[
sage: I = O.ideal(y)
\]
\[
sage: [f.place() for f,_ in I.factor()]
[Place (x, x*y), Place (x + 1, x*y)]
\]
\[
sage: K.<x> = FunctionField(GF(3^2)); R.<t> = PolynomialRing(K)
\]
\[
sage: F.<y> = K.extension(t^3 + t^2 - x^4)
\]
\[
sage: Oinf = F.maximal_order_infinite()
\]
\[
sage: I = Oinf.ideal(1/x)
\]
\[
sage: I.factor()
(Ideal (1/x,1/x^3*y^2) of Maximal infinite order of Function field
in y defined by y^3 + y^2 + 2*x^4)^3
\]
\[
sage: J = I.factor()[0][0]
\]
\[
sage: J.is_prime()
True
\]
\[
sage: J.place()
Place (1/x, 1/x*y)
\]
\[
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K
\]
\[
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
\]
\[
sage: Oinf = L.maximal_order_infinite()
\]
\[
sage: I = Oinf.ideal(1/x)
\]
\[
sage: I.factor()
(Ideal (1/x,1/x*y) of Maximal infinite order of Function field in y
defined by y^2 + y + (x^2 + 1)/x)^2
\]
\[
sage: J = I.factor()[0][0]
\]
\[
sage: J.is_prime()
True
\]
\[
sage: J.place()
Place (1/x, 1/x*y)
\]

\textbf{ring()}

Return the ring to which this ideal belongs.

\textbf{EXAMPLES:}

\[
sage: K.<x> = FunctionField(GF(7))
\]
\[
sage: O = K.equation_order()
\]
\[
sage: I = O.ideal(x,x^2,x^2+x)
\]
\[
sage: I.ring()
Maximal order of Rational function field in x over Finite Field of size 7
\]

\textbf{class} \texttt{sage.rings.function_field.ideal.FunctionFieldIdealInfinite}\texttt{(ring)}

\textbf{Bases:} \texttt{sage.rings.function_field.ideal.FunctionFieldIdeal}
Base class of ideals of maximal infinite orders

```python
class sage.rings.function_field.ideal.FunctionFieldIdealInfinite_global(ring, ideal):
    Bases: sage.rings.function_field.ideal.FunctionFieldIdealInfinite

Ideals of the infinite maximal order.

INPUT:

- `ring` -- infinite maximal order of the function field
- `ideal` -- ideal in the inverted function field

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(3^2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3+t^2-x^4)
sage: Oinf = F.maximal_order_infinite()
sage: Oinf.ideal(1/y)
Ideal (1/x^2,1/x^4*y^2) of Maximal infinite order of Function field in y defined by y^3 + y^2 + 2*x^4
```

gens()  
Return a set of generators of this ideal.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(3^2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3+t^2-x^4)
sage: Oinf = F.maximal_order_infinite()
sage: I = Oinf.ideal(x+y)
sage: I.gens()
(x, y, 1/x^2*y^2)
```

gens_over_base()  
Return a set of generators of this ideal.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(3^2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3+t^2-x^4)
sage: Oinf = F.maximal_order_infinite()
sage: I = Oinf.ideal(x+y)
sage: I.gens_over_base()
(x, y, 1/x^2*y^2)
```

gens_two()  
Return a set of at most two generators of this ideal.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(3^2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3+t^2-x^4)
```
(continues on next page)
ideal_below()

Return a set of generators of this ideal.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(3^2)); _.<t> = K[]
sage: F.<y> = K.extension(t^3 + t^2 - x^4)
sage: Oinf = F.maximal_order_infinite()
sage: I = Oinf.ideal(1/y^2)
sage: I.ideal_below()
Ideal (x^3) of Maximal order of Rational function field
in x over Finite Field in z2 of size 3^2
```

is_prime()

Return True if this ideal is a prime ideal.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(3^2)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3 + t^2 - x^4)
sage: Oinf = F.maximal_order_infinite()
sage: I = Oinf.ideal(1/x)
sage: I.factor()
(Ideal (1/x,1/x^3*y^2) of Maximal infinite order of Function field
in y defined by y^3 + y^2 + 2*x^4)^3
sage: I.is_prime()
False
sage: J = I.factor()[0][0]
sage: J.is_prime()
True
```

prime_below()

Return the prime of the base order that underlies this prime ideal.
EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(3^2)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3+t^2-x^4)
sage: Oinf = F.maximal_order_infinite()
sage: I = Oinf.ideal(1/x)
sage: I.factor()
(Ideal (1/x,1/x^3*y^2) of Maximal infinite order of Function field
in y defined by y^3 + y^2 + 2*x^4)^3
sage: J = I.factor()[0][0]
sage: J.is_prime()
True
sage: J.prime_below()
Ideal (1/x) of Maximal infinite order of Rational function field
in x over Finite Field in z2 of size 3^2

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: Oinf = L.maximal_order_infinite()
sage: I = Oinf.ideal(1/x)
sage: I.factor()
(Ideal (1/x,1/x*y) of Maximal infinite order of Function field in y
defined by y^2 + y + (x^2 + 1)/x)^2
sage: J = I.factor()[0][0]
sage: J.is_prime()
True
sage: J.prime_below()
Ideal (1/x) of Maximal infinite order of Rational function field in x
over Finite Field of size 2
```

valuation(ideal)

Return the valuation of ideal with respect to this prime ideal.

INPUT:

- ideal – fractional ideal

EXAMPLES:

```python
sage: K.<x>=FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y>=K.extension(Y^2 + Y + x + 1/x)
sage: Oinf = L.maximal_order_infinite()
sage: I = Oinf.ideal(y)
sage: [f.valuation(I) for f, _ in I.factor()]
[-1]
```

class sage.rings.function_field.ideal.FunctionFieldIdealInfinite_module(ring, module)

Bases: sage.rings.function_field.ideal.FunctionFieldIdealInfinite, sage.rings.ideal.Ideal_generic

A fractional ideal specified by a finitely generated module over the integers of the base field.

INPUT:

- ring – order in a function field
- module – module

EXAMPLES:
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order()
sage: O.ideal(y)
Ideal (x^3 + 1, -y) of Order in Function field in y defined by y^2 - x^3 - 1

module()
Return the module over the maximal order of the base field that underlies this ideal.
The formation of the module is compatible with the vector space corresponding to the function field.

EXAMPLES:

```sage
sage: K.<x> = FunctionField(GF(7))
sage: O = K.maximal_order(); O
Maximal order of Rational function field in x over Finite Field of size 7
sage: K.polynomial_ring()
Univariate Polynomial Ring in x over Rational function field in x over Finite Field of size 7
sage: I = O.ideal([x^2 + 1, x*(x^2+1)])
sage: I.gens()
(x^2 + 1,)
sage: I.module()
Free module of degree 1 and rank 1 over Maximal order of Rational function field in x over Finite Field of size 7
Echelon basis matrix:
[x^2 + 1]
sage: V, from_V, to_V = K.vector_space(); V
Vector space of dimension 1 over Rational function field in x over Finite Field of size 7
sage: I.module().is_submodule(V)
True
```

class sage.rings.function_field.ideal.FunctionFieldIdealInfinite_rational(ring, gen)
Bases: sage.rings.function_field.ideal.FunctionFieldIdealInfinite

Fractional ideal of the maximal order of rational function field.

INPUT:
- ring – infinite maximal order
- gen – generator

Note that the infinite maximal order is a principal ideal domain.

EXAMPLES:

```sage
sage: K.<x> = FunctionField(GF(2))
sage: Oinf = K.maximal_order_infinite()
sage: Oinf.ideal(x)
Ideal (x) of Maximal infinite order of Rational function field in x over Finite Field of size 2
```

gen()
Return the generator of this principal ideal.

EXAMPLES:
```python
sage: K.<x> = FunctionField(GF(2))
sage: Oinf = K.maximal_order_infinite()
sage: I = Oinf.ideal((x+1)/(x^3+x), (x^2+1)/x^4)
sage: I.gen()
1/x^2

gens()

Return the generator of this principal ideal.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2))
sage: Oinf = K.maximal_order_infinite()
sage: I = Oinf.ideal((x+1)/(x^3+x), (x^2+1)/x^4)
sage: I.gens()
(1/x^2,)
```

gens_over_base()

Return the generator of this ideal as a rank one module over the infinite maximal order.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2))
sage: Oinf = K.maximal_order_infinite()
sage: I = Oinf.ideal((x+1)/(x^3+x), (x^2+1)/x^4)
sage: I.gens_over_base()
(1/x^2,)
```
is_prime()

Return True if this ideal is a prime ideal.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2))
sage: Oinf = K.maximal_order_infinite()
sage: I = Oinf.ideal(x/(x^2 + 1))
sage: I.is_prime()
True
```

valuation(ideal)

Return the valuation of ideal at this prime ideal.

INPUT:

- ideal – fractional ideal

EXAMPLES:

```python
sage: F.<x> = FunctionField(QQ)
sage: O = F.maximal_order_infinite()
sage: p = O.ideal(1/x)
sage: p.valuation(O.ideal(x/(x+1)))
0
sage: p.valuation(O.ideal(0))
+Infinity
```

class sage.rings.function_field.ideal.FunctionFieldIdeal_global(ring, hnf, denominator=1)

Bases: sage.rings.function_field.ideal.FunctionFieldIdeal
Fractional ideals of canonical function fields

**INPUT:**

- `ring` – order in a function field
- `hnf` – matrix in hermite normal form
- `denominator` – denominator

The rows of `hnf` is a basis of the ideal, which itself is `denominator` times the fractional ideal.

**EXAMPLES:**

```
sage: K.<x> = FunctionField(GF(2)); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3*y - x)
sage: O = L.maximal_order()
sage: O.ideal(y)
Ideal (x, y) of Maximal order of Function field in y defined by y^2 + x^3*y + x
```

`basis_matrix()`

Return the matrix of basis vectors of this ideal as a module.

The basis matrix is by definition the hermite norm form of the ideal divided by the denominator.

**EXAMPLES:**

```
sage: K.<x> = FunctionField(GF(2)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(x,1/y)
sage: I.denominator() * I.basis_matrix() == I.hnf()
True
```

`denominator()`

Return the denominator of this fractional ideal.

**EXAMPLES:**

```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.maximal_order()
sage: I = O.ideal(y/(y+1))
sage: d = I.denominator(); d
x^3
sage: d in O
True
```

`gens()`

Return a set of generators of this ideal.

This provides whatever set of generators as quickly as possible.

**EXAMPLES:**

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 - x^3*Y - x)
sage: O = L.maximal_order()
sage: I = O.ideal(x+y)
sage: I.gens()
(x^4 + x^2 + x, y + x)
```

(continues on next page)
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: O = L.maximal_order()
sage: I = O.ideal(x+y)
sage: I.gens()
(x^3 + 1, y + x)

\texttt{gens\_over\_base}()

Return the generators of this ideal as a module over the maximal order of the base rational function field.

\textbf{EXAMPLES:}

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[

sage: L.<y> = K.extension(Y^2 - x^3*Y - x)
sage: O = L.maximal_order()
sage: I = O.ideal(x+y)
sage: I.gens_over_base()
(x^4 + x^2 + x, y + x)

sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: O = L.maximal_order()
sage: I = O.ideal(x+y)
sage: I.gens_over_base()
(x^3 + 1, y + x)

\texttt{gens\_two}()

Return two generators of this fractional ideal.

If the ideal is principal, one generator may be returned.

\textbf{EXAMPLES:}

sage: K.<x> = FunctionField(GF(2)); _.<t> = K[

sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(y)

\textbf{hnf}()

Return the matrix in hermite normal form representing this ideal.

See also \texttt{denominator}()

\textbf{EXAMPLES:}
ideal_below()  
Return the ideal below this ideal.
This is defined only for integral ideals.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<t> = K[]
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(x,1/y)
sage: I.ideal_below()
Traceback (most recent call last):
...
TypeError: not an integral ideal
sage: J = I.denominator() * I
sage: J.ideal_below()
Ideal (x^3 + x^2 + x) of Maximal order of Rational function field
in x over Finite Field of size 2
```

intersect(other)  
Intersect this ideal with the other ideal as fractional ideals.

INPUT:

• other — ideal

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 - x^3*Y - x)
sage: O = L.maximal_order()
sage: I = O.ideal(x+y)
sage: J = O.ideal(x)
sage: I.intersect(J) == I * J * (I + J)^-1
True
```

is_integral()  
Return True if this is an integral ideal.
EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(x,1/y)
sage: I.is_integral()
False
sage: J = I.denominator() * I
sage: J.is_integral()
True
```

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: O = L.maximal_order()
sage: I = O.ideal(y)
sage: J = I.denominator() * I
sage: J.is_integral()
True
```

\textbf{is\_prime()}

Return \texttt{True} if this ideal is a prime ideal.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(y)
sage: [f.is_prime() for f,_ in I.factor()]
[True, True]
```

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: O = L.maximal_order()
sage: I = O.ideal(y)
sage: [f.is_prime() for f,_ in I.factor()]
[True, True]
```

\textbf{module()}

Return the module, that is the ideal viewed as a module over the base maximal order.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: F.<y> = K.extension(y^2 - x^3 - 1)
sage: O = F.maximal_order()
sage: I = O.ideal(x,1/y)
sage: I.module()
Free module of degree 2 and rank 2 over Maximal order of Rational function field in x over Finite Field of size 7
Echelon basis matrix:
[    1 0]
[0 1/(x^3 + 1)]
```

\textbf{norm()}

Return the norm of this fractional ideal.
EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: O = F.maximal_order()
sage: i1 = O.ideal(x)
sage: i2 = O.ideal(y)
sage: i3 = i1 * i2
sage: i3.norm() == i1.norm() * i2.norm()
True
sage: i1.norm()
x^3
sage: i1.norm() == x ** F.degree()
True
sage: i2.norm()
x^6 + x^4 + x^2
sage: i2.norm() == y.norm()
True
```

```python
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: O = L.maximal_order()
sage: i1 = O.ideal(x)
sage: i2 = O.ideal(y)
sage: i3 = i1 * i2
sage: i3.norm() == i1.norm() * i2.norm()
True
sage: i1.norm()
x^2
sage: i1.norm() == x ** L.degree()
True
sage: i2.norm()
(x^2 + 1)/x
sage: i2.norm() == y.norm()
True
```

prime_below()

Return the prime lying below this prime ideal.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^3-x^2*(x^2+x+1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(y)
sage: [f.prime_below() for f, _ in I.factor()]
```

```
[Ideal (x) of Maximal order of Rational function field in x
over Finite Field of size 2, Ideal (x^2 + x + 1) of Maximal order
of Rational function field in x over Finite Field of size 2]
```

```python
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: O = L.maximal_order()
sage: I = O.ideal(y)
sage: [f.prime_below() for f, _ in I.factor()]
```

```
[Ideal (x) of Maximal order of Rational function field in x over Finite Field
→ of size 2, Ideal (x + 1) of Maximal order of Rational function field in x over Finite Field
→ of size 2]
```
valuation(ideal)
Return the valuation of ideal at this prime ideal.

INPUT:
• ideal – fractional ideal

EXAMPLES:

sage: K.<x> = FunctionField(GF(2)); _.<t> = K[]
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(x, (1/(x^3 + x^2 + x))*y^2)
sage: I.is_prime()  # True
sage: J = O.ideal(y)
sage: I.valuation(J)  # 2

The method closely follows Algorithm 4.8.17 of [Coh1993].

class sage.rings.function_field.ideal.FunctionFieldIdeal_module
Bases: sage.rings.function_field.ideal.FunctionFieldIdeal, sage.rings.ideal.Ideal_generic
A fractional ideal specified by a finitely generated module over the integers of the base field.

INPUT:
• ring – an order in a function field
• module – a module of the order

EXAMPLES:

An ideal in an extension of a rational function field:

sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.maximal_order()
sage: I = O.ideal(y)
sage: I  # Ideal (x^3 + 1, -y) of Order in Function field in y defined by y^2 - x^3 - 1
sage: I^2  # Ideal (x^3 + 1, (-x^3 - 1)*y) of Order in Function field in y defined by y^2 - x^3 - 1

gen(i)
Return the i-th generator in the current basis of this ideal.

EXAMPLES:
sage: I = O.ideal(x^2 + 1)
sage: I.gen(1)
(x^2 + 1)*y

gens()
Return a set of generators of this ideal.

EXAMPLES:

sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order()
sage: I = O.ideal(x^2 + 1)
sage: I.gens()
(x^2 + 1, (x^2 + 1)*y)

intersection(other)
Return the intersection of this ideal and other.

EXAMPLES:

sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order(); O
Order in Function field in y defined by y^2 - x^3 - 1
sage: I = O.ideal(y^3); J = O.ideal(y^2)
sage: Z = I.intersection(J); Z
Ideal (x^6 + 2*x^3 + 1, (-x^3 - 1)*y) of Order in Function field in y defined by y^2 - x^3 - 1
sage: y^2 in Z
False
sage: y^3 in Z
True

module()
Return the module over the maximal order of the base field that underlies this ideal.

The formation of the module is compatible with the vector space corresponding to the function field.

OUTPUT:

• a module over the maximal order of the base field of the ideal

EXAMPLES:

sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: O = L.equation_order(); O
Order in Function field in y defined by y^2 - x^3 - 1
sage: I = O.ideal(x^2 + 1)
sage: I.gens()
(x^2 + 1, (x^2 + 1)*y)
sage: I.module()
Free module of degree 2 and rank 2 over Maximal order of Rational function field in x over Rational Field
Echelon basis matrix:
[ x^2 + 1  0 ]
[  0 x^2 + 1 ]
sage: V, from_V, to_V = L.vector_space(); V

(continues on next page)
Vector space of dimension 2 over Rational function field in x over Rational\
\rightarrow Field
sage: I.module().is submodule(V)
True

gen()  
Return the number of generators in the basis.

EXAMPLES:

sage: K.<x> = FunctionField(QQ); R.<y> = K[]  
sage: L.<y> = K.extension(y^2 - x^3 - 1)  
sage: O = L.equation_order()  
sage: I = O.ideal(x^2 + 1)  
sage: I.ngens()
2

class sage.rings.function_field.ideal.FunctionFieldIdeal_rational(ring, gen)
Bases: sage.rings.function_field.ideal.FunctionFieldIdeal

Fractional ideals of the maximal order of a rational function field.

INPUT:
- `ring` – the maximal order of the rational function field.
- `gen` – generator of the ideal, an element of the function field.

EXAMPLES:

sage: K.<x> = FunctionField(QQ)  
sage: O = K.maximal_order()  
sage: I = O.ideal(1/(x^2+x)); I
Ideal (1/(x^2 + x)) of Maximal order of Rational function field in x over Rational Field

denominator()  
Return the denominator of this fractional ideal.

EXAMPLES:

sage: F.<x> = FunctionField(QQ)  
sage: O = F.maximal_order()  
sage: I = O.ideal(x/(x^2+1))  
sage: I.denominator()
x^2 + 1

gen()  
Return the unique generator of this ideal.

EXAMPLES:

sage: K.<x> = FunctionField(GF(4))  
sage: O = K.maximal_order()  
sage: I = O.ideal(x^2+x)  
sage: I.gen()
x^2 + x

gens()  
Return the tuple of the unique generator of this ideal.
EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(4))
sage: O = K.maximal_order()
sage: I = O.ideal(x^2+x)
sage: I.gens()
(x^2 + x,)
```

gens_over_base()
Return the generator of this ideal as a rank one module over the maximal order.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(4))
sage: O = K.maximal_order()
sage: I = O.ideal(x^2+x)
sage: I.gens_over_base()
(x^2 + x,)
```

is_prime()
Return `True` if this is a prime ideal.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ)
sage: O = K.maximal_order()
sage: I = O.ideal(x^3+x^2)
[sage: [f.is_prime() for f,m in I.factor()]
[True, True]
```

module()
Return the module, that is the ideal viewed as a module over the ring.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ)
sage: O = K.maximal_order()
sage: I = O.ideal(x^3+x^2)
sage: I.module()
Free module of degree 1 and rank 1 over Maximal order of Rational
function field in x over Rational Field
Echelon basis matrix:
[x^3 + x^2]
sage: J = 0*I
sage: J.module()
Free module of degree 1 and rank 0 over Maximal order of Rational
function field in x over Rational Field
Echelon basis matrix:
[]
```

valuation(ideal)
Return the valuation of the ideal at this prime ideal.

INPUT:

- `ideal` – fractional ideal

EXAMPLES:
sage: F.<x> = FunctionField(QQ)
sage: O = F.maximal_order()
sage: I = O.ideal(x^2*(x^2+x+1)^3)
sage: [f.valuation(I) for f, _ in I.factor()]
[2, 3]

class sage.rings.function_field.ideal.IdealMonoid(R)
    Bases: sage.structure.unique_representation.UniqueRepresentation, sage.structure.parent.Parent

    The monoid of ideals in orders of function fields.

    INPUT:

    - R -- order

    EXAMPLES:

    sage: K.<x> = FunctionField(GF(2))
sage: O = K.maximal_order()
sage: M = O.ideal_monoid(); M
    Monoid of ideals of Maximal order of Rational function field in x over Finite Field of size 2

    sage: M.ring()
    is O
    True

    ring()

    Return the ring of which this is the ideal monoid.

    EXAMPLES:

    sage: K.<x> = FunctionField(GF(2))
sage: O = K.maximal_order()
sage: M = O.ideal_monoid(); M.ring() is O
    True
PLACES OF FUNCTION FIELDS

Places are defined for arbitrary function fields, but presently Sage can find and compute with places only of rational function fields and global function fields.

The places of a function field correspond, one-to-one, to valuation rings of the function field, each of which defines discrete valuation of the elements of the function field. “Finite” places are in one-to-one correspondence with the prime ideals of the finite maximal order while places “at infinity” are in one-to-one correspondence with the prime ideals of the infinite maximal order.

EXAMPLES:

All rational places of the function field can be computed:

```sage
K.<x> = FunctionField(GF(2)); _.<Y> = K[]
L.<y> = K.extension(Y^3 + x + x^3*Y)
L.places()
[Place (1/x, 1/x^3*y^2 + 1/x),
 Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1),
 Place (x, y)]
```

The residue field associated with a place is given as an extension of the constant field:

```sage
F.<x> = FunctionField(GF(2))
O = F.maximal_order()
p = O.ideal(x^2 + x + 1).place()
```

```sage
k, fr_k, to_k = p.residue_field()
k
Finite Field in z2 of size 2^2
```

The isomorphisms are between the valuation ring and the residue field:

```sage
fr_k
Ring morphism:
 From: Finite Field in z2 of size 2^2
 To: Valuation ring at Place (x^2 + x + 1)
to_k
Ring morphism:
 From: Valuation ring at Place (x^2 + x + 1)
 To: Finite Field in z2 of size 2^2
```

AUTHORS:

- Kwankyu Lee (2017-04-30): initial version

```sage
class FunctionFieldPlace(parent, prime)
Bases: sage.structure.element.Element
```
Places of function fields.

INPUT:

- **field** – function field
- **prime** – prime ideal associated with the place

EXAMPLES:

```python
sage: K.<x>=FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y>=K.extension(Y^3 + x + x^3*Y)
sage: L.places_finite()[0]
Place (x, y)
```

`divisor (multiplicity=1)`

Return the prime divisor corresponding to the place.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(5)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^2-x^3-1)
sage: O = F.maximal_order()
sage: I = O.ideal(x+1,y)
sage: P = I.place()
sage: P.divisor()
Place (x + 1, y)
```

`function_field()`

Return the function field to which the place belongs.

EXAMPLES:

```python
sage: K.<x>=FunctionField(GF(2)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x+x^3*Y)
sage: p = L.places()[0]
sage: p.function_field() == L
True
```

`prime_ideal()`

Return the prime ideal associated with the place.

EXAMPLES:

```python
sage: K.<x>=FunctionField(GF(2)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x+x^3*Y)
sage: p = L.places()[0]
sage: p.prime_ideal()
Ideal (1/x,1/x^3*y^2 + 1/x) of Maximal infinite order of Function field in y defined by y^3 + x^3*y + x
```

**class** `sage.rings.function_field.place.FunctionFieldPlace_global (parent, prime)`

**Bases:** `sage.rings.function_field.place.FunctionFieldPlace`

Places of function fields

`degree()`

Return the degree of the place.

EXAMPLES:
sage: K.<x>=FunctionField(GF(2)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x^3*Y)
sage: OK = K.maximal_order()
sage: OL = L.maximal_order()
sage: p = OK.ideal(x^2 + x + 1)
sage: dec = OL.decomposition(p)
sage: q = dec[0][0].place()
sage: q.degree()  
2

\texttt{gaps()}

Return the gap sequence for the place.

\textbf{EXAMPLES:}

\begin{verbatim}
sage: K.<x>=FunctionField(GF(4)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x^3*Y)
sage: O = L.maximal_order()
sage: p = O.ideal(x,y).place()
sage: p.gaps() # a Weierstrass place
[1, 2, 4]
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
sage: [p.gaps() for p in L.places()]
[[1, 2, 4], [1, 2, 4], [1, 2, 4]]
\end{verbatim}

\texttt{is_infinite_place()}

Return True if the place is above the unique infinite place of the underlying rational function field.

\textbf{EXAMPLES:}

\begin{verbatim}
sage: K.<x>=FunctionField(GF(2)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x^3*Y)
sage: pls = L.places()
sage: [p.is_infinite_place() for p in pls]
[True, True, False]
sage: [p.place_below() for p in pls]
[Place (1/x), Place (1/x), Place (x)]
\end{verbatim}

\texttt{local_uniformizer()}

Return an element of the function field that has a simple zero at the place.

\textbf{EXAMPLES:}

\begin{verbatim}
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
sage: pls = L.places()
sage: [p.local_uniformizer().valuation(p) for p in pls]
[1, 1, 1, 1]
\end{verbatim}

\texttt{place_below()}

Return the place lying below the place.

\textbf{EXAMPLES:}

\begin{verbatim}
sage: K.<x>=FunctionField(GF(2)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x^3*Y)
\end{verbatim}
relative_degree()

Return the relative degree of the place.

EXAMPLES:

residue_field(name=None)

Return the residue field of the place.

INPUT:

• name – string; name of the generator of the residue field

EXAMPLES:

valuation_ring()

Return the valuation ring at the place.
EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: p.valuation_ring()
Valuation ring at Place (x, x*y)
```

class sage.rings.function_field.place.FunctionFieldPlace_rational(parent, prime)

Places of rational function field.

**degree()**

Return the degree of the place.

**EXAMPLES:**

```python
sage: F.<x> = FunctionField(GF(2))
sage: O = F.maximal_order()
sage: i = O.ideal(x^2+x+1)
sage: p = i.place()
sage: p.degree()
2
```

**is_infinite_place()**

Return True if the place is at infinite.

**EXAMPLES:**

```python
sage: F.<x> = FunctionField(GF(2))
sage: F.places()
[Place (1/x), Place (x), Place (x + 1)]
sage: [p.is_infinite_place() for p in F.places()]
[True, False, False]
```

**local_uniformizer()**

Return a local uniformizer of the place.

**EXAMPLES:**

```python
sage: F.<x> = FunctionField(GF(2))
sage: F.places()
[Place (1/x), Place (x), Place (x + 1)]
sage: [p.local_uniformizer() for p in F.places()]
[1/x, x, x + 1]
```

**residue_field(name=None)**

Return the residue field of the place.

**EXAMPLES:**

```python
sage: F.<x> = FunctionField(GF(2))
sage: O = F.maximal_order()
sage: p = O.ideal(x^2 + x + 1).place()
sage: k, fr_k, to_k = p.residue_field()
sage: k
Finite Field in z2 of size 2^2
sage: fr_k
(continues on next page)```
Ring morphism:
  From: Finite Field in z2 of size 2^2
  To:   Valuation ring at Place (x^2 + x + 1)
\texttt{sage: to_k}
Ring morphism:
  From: Valuation ring at Place (x^2 + x + 1)
  To:   Finite Field in z2 of size 2^2

\textbf{valuation\_ring()}

Return the valuation ring at the place.

\textbf{EXAMPLES:}

\begin{verbatim}
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: p.valuation_ring()
Valuation ring at Place (x, x*y)
\end{verbatim}

\textbf{class} \texttt{sage.rings.function\_field.place\_set\_\_place\_Set\_\_field(field)}

\textbf{Bases:} \texttt{sage.structure.unique\_representation.Unique\_Representation, sage.structure.parent.Parent}

Sets of Places of function fields.

\textbf{INPUT:}

\begin{itemize}
  \item \texttt{field} – function field
\end{itemize}

\textbf{EXAMPLES:}

\begin{verbatim}
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x + x^3*Y)
sage: L.place_set()
Set of places of Function field in y defined by y^3 + x^3*y + x
\end{verbatim}

\textbf{Element}

\texttt{alias of FunctionFieldPlace}
Sage allows extensive computations with divisors on global function fields.

**EXAMPLES:**

The divisor of an element of the function field is the formal sum of poles and zeros of the element with multiplicities:

```sage
sage: K.<x> = FunctionField(GF(2)); R.<t> = K[]
sage: L.<y> = K.extension(t^3 + x^3*t + x)
sage: f = x/(y+1)
sage: f.divisor()
- Place (1/x, 1/x^3*y^2 + 1/x)
  + Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1)
  + 3*Place (x, y)
- Place (x^3 + x + 1, y + 1)
```

The Riemann-Roch space of a divisor can be computed. We can get a basis of the space as a vector space over the constant field:

```sage
sage: p = L.places_finite()[0]
sage: q = L.places_infinite()[0]
sage: (3*p + 2*q).basis_function_space()
[1/x*y^2 + x^2, 1, 1/x]
```

We verify the Riemann-Roch theorem:

```sage
sage: D = 3*p - q
sage: index_of_speciality = len(D.basis_differential_space())
sage: D.dimension() == D.degree() - L.genus() + 1 + index_of_speciality
True
```

**AUTHORS:**

- Kwankyu Lee (2017-04-30): initial version

```python
class sage.rings.function_field.divisor.DivisorGroup(field)

Bases:    sage.structure.unique_representation.UniqueRepresentation, sage.structure.parent.Parent

Groups of divisors of function fields.

INPUT:

- field -- function field

EXAMPLES:
```

103
```python
sage: K.<x> = FunctionField(GF(5)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^2 - x^3 - 1)
sage: F.divisor_group()
Divisor group of Function field in y defined by y^2 + 4*x^3 + 4
```

**Element**

alias of *FunctionFieldDivisor*

**function_field()**

Return the function field to which the divisor group is attached.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(5)); _.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^2 - x^3 - 1)
sage: G = F.divisor_group()
sage: G.function_field()
Function field in y defined by y^2 + 4*x^3 + 4
```

class *sage.rings.function_field.divisor.FunctionFieldDivisor*(parent, data)

Divisors of function fields.

**INPUT:**

- parent – divisor group
- data – dict of place and multiplicity pairs

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(2)); R.<t> = K[
]
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: f = x/(y+1)
sage: f.divisor()
Place (1/x, 1/x^4*y^2 + 1/x^2*y + 1)
+ Place (1/x, 1/x^2*y + 1)
+ 3*Place (x, (1/(x^3 + x^2 + x))*y^2)
- 6*Place (x + 1, y + 1)
```

**basis_differential_space()**

Return a basis of the space of differentials $\Omega(D)$ for the divisor $D$.

**EXAMPLES:**

We check the Riemann-Roch theorem:

```python
sage: K.<x>=FunctionField(GF(4)); _.<Y>=K[
]
sage: L.<y>=K.extension(Y^3+x+x^3*Y)
sage: d = 3*L.places()[0]
sage: i = len(d.basis_function_space())
sage: i == d.degree() + 1 - L.genus() + i
True
```

**basis_function_space()**

Return a basis of the Riemann-Roch space of the divisor.

**EXAMPLES:**
sage: K.<x> = FunctionField(GF(5)); R.<t> = K[]
sage: F.<y> = K.extension(t^2 - x^3 - 1)
sage: O = F.maximal_order()
sage: I = O.ideal(x-2)
sage: D = I.divisor()
sage: D.basis_function_space()
[x/(x + 3), 1/(x + 3)]

degree()

Return the degree of the divisor.

EXAMPLES:

sage: K.<x> = FunctionField(GF(4)); _.<Y>=K[
    sage: L.<y> = K.extension(Y^3+x+x^3*Y)
    sage: p1,p2 = L.places()[:2]
    sage: D = 2*p1 - 3*p2
    sage: D.degree()
    -1

dict()

Return the dictionary representing the divisor.

EXAMPLES:

sage: K.<x> = FunctionField(GF(4)); _.<Y>=K[
    sage: L.<y> = K.extension(Y^3+x+x^3*Y)
    sage: f = x/(y+1)
    sage: D = f.divisor()
    sage: D.dict()
    {Place (1/x, 1/x^3*y^2 + 1/x): -1,
     Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1): 1,
     Place (x, y): 3,
     Place (x^3 + x + 1, y + 1): -1}

differential_space()

Return the vector space of the differential space $\Omega(D)$ of the divisor $D$.

OUTPUT:

- a vector space isomorphic to $\Omega(D)$
- an isomorphism from the vector space to the differential space
- the inverse of the isomorphism

EXAMPLES:

sage: K.<x> = FunctionField(GF(5)); R.<t> = K[]
sage: F.<y> = K.extension(t^2 - x^3 - 1)
sage: O = F.maximal_order()
sage: I = O.ideal(x-2)
sage: P1 = I.divisor().support()[0]
sage: Pinf = F.places_infinite()[0]
sage: D = -3*Pinf + P1
sage: V, from_V, to_V = D.differential_space()
sage: all(to_V(from_V(e)) == e for e in V)
True
**dimension()**
Return the dimension of the Riemann-Roch space of the divisor.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(5)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^2-x^3-1)
sage: O = F.maximal_order()
sage: I = O.ideal(x-2)
sage: P1 = I.divisor().support()[0]
sage: Pinf = F.places_infinite()[0]
sage: D = 3*Pinf+2*P1
sage: D.dimension()
5
```

**function_space()**
Return the vector space of the Riemann-Roch space of the divisor.

OUTPUT:
- a vector space, an isomorphism from the vector space to the Riemann-Roch space, and its inverse.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(5)); R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^2-x^3-1)
sage: O = F.maximal_order()
sage: I = O.ideal(x-2)
sage: D = I.divisor()
sage: V, from_V, to_V = D.function_space()
sage: all(to_V(from_V(e)) == e for e in V)
True
```

**list()**
Return the list of place and multiplicity pairs of the divisor.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(4)); _.<Y>=K[

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

**multiplicity(place)**
Return the multiplicity of the divisor at the place.

INPUT:
- place – place of a function field

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(4)); _.<Y>=K[

(continues on next page)
```
support()

Return the support of the divisor.

EXAMPLES:

```sage
K.<x> = FunctionField(GF(4)); _.<Y>=K[
L.<y> = K.extension(Y^3+x+x^3*Y)
f = x/(y+1)
D = f.divisor()
D.support()
```

```
[Place (1/x, 1/x^3*y^2 + 1/x),
 Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1),
 Place (x, y),
 Place (x^3 + x + 1, y + 1)]
```

sage.rings.function_field.divisor.divisor(field, data)

Construct a divisor from the data.

INPUT:

- `field` – function field
- `data` – dictionary of place and multiplicity pairs

EXAMPLES:

```sage
K.<x> = FunctionField(GF(2)); R.<t> = K[
F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
p, q, r = F.places()
from sage.rings.function_field.divisor import divisor
d = prime_divisor(F, p)
```

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

sage.rings.function_field.divisor.prime_divisor(field, place, m=1)

Construct a prime divisor from the place.

INPUT:

- `field` – function field
- `place` – place of the function field
- `m` – (default: 1) a positive integer; multiplicity at the place

EXAMPLES:

```sage
K.<x> = FunctionField(GF(2)); R.<t> = K[
F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
p = F.places()[0]
from sage.rings.function_field.divisor import prime_divisor
d = prime_divisor(F, p)
```
sage: 3 * d == prime_divisor(F, p, 3)
True
DIFFERENTIALS OF FUNCTION FIELDS

Sage provides basic arithmetic and advanced computations with differentials on global function fields.

EXAMPLES:

The module of differentials on a function field forms an one-dimensional vector space over the function field:

```
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x + x^3*Y)
sage: f = x + y
go: g = 1 / y
g: df = f.differential()
g: dg = g.differential()
g: dfdg = f.derivative() / g.derivative()
g: df == dfdg * dg
True
g: df
(x*y^2 + 1/x*y + 1) d(x)
go: df.parent()
Space of differentials of Function field in y defined by y^3 + x^3*y + x
```

We can compute a canonical divisor:

```
sage: k = df.divisor()
sage: k.degree()
4
sage: k.degree() == 2 * L.genus() - 2
True
```

Exact differentials vanish and logarithmic differentials are stable under the Cartier operation:

```
sage: df.cartier()
0
sage: w = 1/f * df
sage: w.cartier() == w
True
```

AUTHORS:

- Kwankyu Lee (2017-04-30): initial version

```
class sage.rings.function_field.differential.DifferentialsSpace(field, category=None)
    Bases: sage.structure.unique_representation.UniqueRepresentation, sage.
           structure.parent.Parent

    Space of differentials of a function field.
```
INPUT:

- field – function field

EXAMPLES:

```python
sage: K.<x>=FunctionField(GF(4)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x+x^3*Y)
sage: L.space_of_differentials()
Space of differentials of Function field in y defined by y^3 + x^3*y + x
```

**Element**

alias of `FunctionFieldDifferential_global`

**basis()**

Return a basis.

EXAMPLES:

```python
sage: K.<x>=FunctionField(GF(4)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x+x^3*Y)
sage: S = L.space_of_differentials()
sage: S.basis()
Family (d(x),)
```

**function_field()**

Return the function field to which the space of differentials is attached.

EXAMPLES:

```python
sage: K.<x>=FunctionField(GF(4)); _.<Y>=K[]
sage: L.<y>=K.extension(Y^3+x+x^3*Y)
sage: S = L.space_of_differentials()
sage: S.function_field()
Function field in y defined by y^3 + x^3*y + x
```

class `sage.rings.function_field.differential.FunctionFieldDifferential`

Bases: `sage.structure.element.ModuleElement`

Base class for differentials on function fields.

class `sage.rings.function_field.differential.FunctionFieldDifferential_global`

Bases: `sage.rings.function_field.differential.FunctionFieldDifferential`

Differentials on global function fields.

INPUT:

- f – element of the function field
- t – element of the function field; if t is not specified, t is the generator of the base rational function field

EXAMPLES:

```python
sage: F.<x>=FunctionField(GF(7))
sage: f = x/(x^2 + x + 1)
sage: f.differential()
((6*x^2 + 1)/(x^4 + 2*x^3 + 3*x^2 + 2*x + 1)) d(x)
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]  
```
cartier()

Return the image of the differential by the Cartier operator.

The Cartier operator operates on differentials. Let \( x \) be a separating element of the function field. If a differential \( \omega \) is written \( \omega = (f_0^p + f_1^p x + \cdots + f_{p-1}^p x^{p-1})dx \) (prime-power representation), then the Cartier operator maps \( \omega \) to \( f_0^p dx \). It is known that this definition does not depend on the choice of \( x \).

The Cartier operator has interesting properties. Notably, the set of exact differentials is precisely the kernel of the Cartier operator and logarithmic differentials are stable under the Cartier operation.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(4)); _.<Y>=K[]
sage: L.<y> = K.extension(Y^3+x+x^3*Y)
sage: f = x/y
sage: w = 1/f * f.differential()
sage: w.cartier() == w
True
```

divisor()

Return the divisor of the differential.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(5)); _.<Y>=K[]
sage: L.<y> = K.extension(Y^3+x+x^3*Y)
sage: w = (1/y) * y.differential()
sage: w.divisor()
- Place (1/x, 1/x^3*y^2 + 1/x)
- Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1)
- Place (x, y)
+ Place (x + 2, y + 3)
+ Place (x^6 + 3*x^5 + 4*x^4 + 2*x^3 + x^2 + 3*x + 4, y + x^5)
```

monomial_coefficients()

Return a dictionary whose keys are indices of basis elements in the support of \( \text{self} \) and whose values are the corresponding coefficients.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(5)); _.<Y>=K[]
sage: L.<y> = K.extension(Y^3+x+x^3*Y)
sage: d = y.differential()
```

\[
\begin{align*}
\text{sage}: & \quad d \\
& \quad d((4*x/(x^7 + 3))*y^2 + ((4*x^7 + 1)/(x^8 + 3*x))*y + x^4/(x^7 + 3)) \ d(x) \\
\text{sage}: & \quad d.monomial_coefficients() \\
& \quad \{0: (4*x/(x^7 + 3))*y^2 + ((4*x^7 + 1)/(x^8 + 3*x))*y + x^4/(x^7 + 3)\}
\end{align*}
\]

**residue(place)**

Return the residue of the differential at the place.

**INPUT:**
- place – place of the function field

**OUTPUT:**
- an element of the residue field of the place

**EXAMPLES:**

We verify the residue theorem in a rational function field:

\[
\begin{align*}
\text{sage}: & \quad F.<x> = FunctionField(GF(4)) \\
\text{sage}: & \quad f = 0 \\
\text{sage}: & \quad \text{while } f == 0: \\
& \quad \quad \text{....: } f = F.random_element() \\
\text{sage}: & \quad w = 1/f * f.differential() \\
\text{sage}: & \quad d = f.divisor() \\
\text{sage}: & \quad s = d.support() \\
\text{sage}: & \quad \text{sum([w.residue(p).trace() for p in s])} \\
0
\end{align*}
\]

and also in an extension field:

\[
\begin{align*}
\text{sage}: & \quad K.<x> = FunctionField(GF(7)); _.<Y> = K[] \\
\text{sage}: & \quad L.<y> = K.extension(Y^3 + x + x^3*Y) \\
\text{sage}: & \quad f = 0 \\
\text{sage}: & \quad \text{while } f == 0: \\
& \quad \quad \text{....: } f = L.random_element() \\
\text{sage}: & \quad w = 1/f * f.differential() \\
\text{sage}: & \quad d = f.divisor() \\
\text{sage}: & \quad s = d.support() \\
\text{sage}: & \quad \text{sum([w.residue(p).trace() for p in s])} \\
0
\end{align*}
\]
A valuation ring of a function field is associated with a place of the function field. The valuation ring consists of all elements of the function field that have nonnegative valuation at the place.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: R = p.valuation_ring()
sage: R
Valuation ring at Place (x, x*y)
sage: R.place() == p
True
```

Thus any nonzero element or its inverse of the function field lies in the valuation ring, as shown in the following example:

```
sage: f = y/(1+y)
sage: f in R
True
sage: f not in R
False
sage: f.valuation(p)
0
```

The residue field at the place is defined as the quotient ring of the valuaion ring modulo its unique maximal ideal. In a global function field, the residue_field() method returns a finite field isomorphic to the residue field:

```
sage: k,phi,psi = R.residue_field()
sage: k
Finite Field of size 2
sage: phi
Ring morphism:
  From: Finite Field of size 2
  To:   Valuation ring at Place (x, x*y)
sage: psi
Ring morphism:
  From: Valuation ring at Place (x, x*y)
  To:   Finite Field of size 2
sage: psi(f) in k
True
```

AUTHORS:
class sage.rings.function_field.valuation_ring.FunctionFieldValuationRing(field, place, category=None)

Bases: sage.structure.unique_representation.UniqueRepresentation, sage.structure.parent.Parent

Base class for valuation rings of function fields.

INPUT:

• field – function field
• place – place of the function field

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: p.valuation_ring()
Valuation ring at Place (x, x*y)
```

place()

Return the place associated with the valuation ring.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: R = p.valuation_ring()
sage: p == R.place()
True
```

class sage.rings.function_field.valuation_ring.FunctionFieldValuationRing_global(field, place, category=None)

Bases: sage.rings.function_field.valuation_ring.FunctionFieldValuationRing

Valuation rings of global function fields.

residue_field(name=None)

Return the residue field of the valuation ring together with the maps from and to it.

INPUT:

• name – string; name of the generator of the residue field

OUTPUT:

• a finite field isomorphic to the residue field
• a ring homomorphism from the valuation ring to the finite field
• a ring homomorphism from the finite field to the valuation ring

EXAMPLES:
```python
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: R = p.valuation_ring()
sage: k, fr_k, to_k = R.residue_field()
sage: k
Finite Field of size 2
sage: fr_k
Ring morphism:
    From: Finite Field of size 2
    To:   Valuation ring at Place (x, x*y)
sage: to_k
Ring morphism:
    From: Valuation ring at Place (x, x*y)
    To:   Finite Field of size 2
sage: to_k(1/y)
0
sage: to_k(y/(1+y))
1
```
CHAPTER
NINE

MORPHISMS OF FUNCTION FIELDS

Maps and morphisms useful for computations with function fields.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: K.hom(1/x)
Function Field endomorphism of Rational function field in x over Rational Field
  Defn: x |--> 1/x
sage: L.<y> = K.extension(y^2 - x)
sage: K.hom(y)
Function Field morphism:
  From: Rational function field in x over Rational Field
  To:   Function field in y defined by y^2 - x
  Defn: x |--> y
sage: L.hom([y,x])
Function Field endomorphism of Function field in y defined by y^2 - x
  Defn: y |--> y
  x |--> x
sage: L.hom([x,y])
Traceback (most recent call last):
...
ValueError: invalid morphism
```

For global function fields, which have positive characteristics, the higher derivation is available:

```python
sage: K.<x> = FunctionField(GF(2)); _.<Y>=K[

sage: L.<y> = K.extension(Y^3+x+x^3*Y)
sage: L.higher_derivation()
sage: h(y^2, 2)
((x^7 + 1)/x^2)*y^2 + x^3*y
```

AUTHORS:

- William Stein (2010): initial version
- Julian Rüth (2011-09-14, 2014-06-23, 2017-08-21): refactored class hierarchy; added derivation classes; morphisms to/from fraction fields
- Kwankyu Lee (2017-04-30): added higher derivations and completions

```python
class sage.rings.function_field.maps.FractionFieldToFunctionField
    Bases: sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism

    Isomorphism from a fraction field of a polynomial ring to the isomorphic function field.

    EXAMPLES:
```
sage: K = QQ['x'].fraction_field()
sage: L = K.function_field()
sage: f = L.coerce_map_from(K); f
Isomorphism:
  From: Fraction Field of Univariate Polynomial Ring in x over Rational Field
  To:   Rational function field in x over Rational Field

See also:

FunctionFieldToFractionField

section()
  Return the inverse map of this isomorphism.

EXAMPLES:

sage: K = QQ['x'].fraction_field()
sage: L = K.function_field()
sage: f = L.coerce_map_from(K)
sage: f.section()
Isomorphism:
  From: Rational function field in x over Rational Field
  To:   Fraction Field of Univariate Polynomial Ring in x over Rational Field

class sage.rings.function_field.maps.FunctionFieldCompletion
  Bases: sage.categories.map.Map

Base class of completions on function fields.

class sage.rings.function_field.maps.FunctionFieldCompletion_global(field, place, name=None, prec=None, gen_name=None)
  Bases: sage.rings.function_field.maps.FunctionFieldCompletion

Completions on global function fields.

INPUT:
  • field – function field
  • place – place of the function field
  • name – string for the name of the series variable
  • prec – positive integer; default precision
  • gen_name – string; name of the generator of the residue field; used only when place is non-rational

EXAMPLES:

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p = L.places_finite()[0]
sage: m = L.completion(p)
sage: m
Completion map:
  From: Function field in y defined by y^2 + y + (x^2 + 1)/x
  To:   Laurent Series Ring in s over Finite Field of size 2
sage: m(x)
The variable name of the series can be supplied. If the place is not rational such that the residue field is a proper extension of the constant field, you can also specify the generator name of the extension:

```
sage: p2 = L.places_finite(2)[0]
sage: p2
Place (x^2 + x + 1, x*y + 1)
sage: m2 = L.completion(p2, 't', gen_name='b')
sage: m2(x)
(b + 1) + t + t^2 + t^4 + t^8 + t^16 + O(t^20)
sage: m2(y)
b + b*t + b*t^3 + b*t^4 + (b + 1)*t^5 + (b + 1)*t^7 + b*t^9 + b*t^11
+ b*t^12 + b*t^13 + b*t^15 + b*t^16 + (b + 1)*t^17 + (b + 1)*t^19 + O(t^20)
```

default_precision()

Return the default precision.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[

```

class sage.rings.function_field.maps.FunctionFieldConversionToConstantBaseField(parent)

Conversion map from the function field to its constant base field.

EXAMPLES:

```
sage: QQ.convert_map_from(K)
Conversion map:
  From: Rational function field in x over Rational Field
  To:   Rational Field
```

class sage.rings.function_field.maps.FunctionFieldDerivation(K)

Base class for derivations on function fields.

A derivation on $R$ is a map $R \to R$ with $D(\alpha + \beta) = D(\alpha) + D(\beta)$ and $D(\alpha \beta) = \beta D(\alpha) + \alpha D(\beta)$ for all $\alpha, \beta \in R$.

EXAMPLES:
```python
sage: K.<x> = FunctionField(QQ)
sage: d = K.derivation()
sage: d
Derivation map:
  From: Rational function field in x over Rational Field
  To:  Rational function field in x over Rational Field
  Defn: x |--> 1

is_injective()  
Return False since a derivation is never injective.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ)
sage: d = K.derivation()
sage: d.is_injective()
False
```

```python
class sage.rings.function_field.maps.FunctionFieldDerivation_inseparable(L)
Bases: sage.rings.function_field.maps.FunctionFieldDerivation

A generator of the space of derivations on L.

INPUT:

- L – a function field which is an inseparable extension of its base field.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: d = L.derivation()
```

This also works for iterated non-monic extensions:

```python
sage: K.<x> = FunctionField(GF(2))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - 1/x)
sage: R.<z> = L[]
sage: M.<z> = L.extension(z^2*y - x^3)
sage: M.derivation()
Derivation map:
  From: Function field in z defined by y*z^2 + x^3
  To:  Function field in z defined by y *z^2 + x^3
  Defn: z |--> 1
       y |--> 0
       x |--> 0
```

```python
class sage.rings.function_field.maps.FunctionFieldDerivation_rational(K, u)
Bases: sage.rings.function_field.maps.FunctionFieldDerivation

Derivations on rational function fields.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ)
sage: K.derivation()
Derivation map:
```
class sage.rings.function_field.maps.FunctionFieldDerivation_separable(L, d)

Bases: sage.rings.function_field.maps.FunctionFieldDerivation

Derivations of separable extensions.

EXAMPLES:

sage: K.<x> = FunctionField(QQ)
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: L.derivation()
Derivation map:
  From: Function field in y defined by y^2 - x
  To:   Function field in y defined by y^2 - x
  Defn: y |--> 1/2/x*y
  x |--> 1

class sage.rings.function_field.maps.FunctionFieldHigherDerivation(field)

Bases: sage.categories.map.Map

Base class of higher derivations on function fields.

INPUT:

• field – function field on which the derivation operates

EXAMPLES:

sage: F.<x> = FunctionField(GF(2))
sage: F.higher_derivation()
Higher derivation map:
  From: Rational function field in x over Finite Field of size 2
  To:   Rational function field in x over Finite Field of size 2

class sage.rings.function_field.maps.FunctionFieldHigherDerivation_global(field)

Bases: sage.rings.function_field.maps.FunctionFieldHigherDerivation

Higher derivations of global function fields.

INPUT:

• field – function field on which the derivation operates

EXAMPLES:

sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x + x^3*Y)
sage: h = L.higher_derivation()
sage: h
Higher derivation map:
  From: Function field in y defined by y^3 + x^3*y + x
  To:   Function field in y defined by y^3 + x^3*y + x
sage: h(y^2, 2)
((x^7 + 1)/x^2)*y^2 + x^3*y

From: Rational function field in x over Rational Field
To:   Rational function field in x over Rational Field
Defn: x |--> 1
class sage.rings.function_field.maps.FunctionFieldHigherDerivation_rational(field)
Bases: sage.rings.function_field.maps.FunctionFieldHigherDerivation

Higher derivations of rational function fields.

INPUT:

• field – function field on which the derivation operates

EXAMPLES:

```sage
test_F.<x> = FunctionField(GF(2))
test_h = test_F.higher_derivation()
test_h
Higher derivation map:
  From: Rational function field in x over Finite Field of size 2
  To:   Rational function field in x over Finite Field of size 2
test_sage: test_h(x^2, 2)
1
```

class sage.rings.function_field.maps.FunctionFieldLinearMap
Bases: sage.categories.morphism.SetMorphism

Linear map to function fields.

class sage.rings.function_field.maps.FunctionFieldLinearMapSection
Bases: sage.categories.morphism.SetMorphism

Section of linear map from function fields.

class sage.rings.function_field.maps.FunctionFieldMorphism(parent, im_gen, base_morphism)
Bases: sage.rings.morphism.RingHomomorphism

Base class for morphisms between function fields.

EXAMPLES:

```sage
test_K.<x> = FunctionField(QQ)
test_f = test_K.hom(1/x); test_f
Function Field endomorphism of Rational function field in x over Rational Field
  Defn: x |--> 1/x
```

class sage.rings.function_field.maps.FunctionFieldMorphism_polymod(parent, im_gen, base_morphism)
Bases: sage.rings.function_field.maps.FunctionFieldMorphism

Morphism from a finite extension of a function field to a function field.

EXAMPLES:

```sage
test_K.<x> = FunctionField(GF(7)); R.<y> = test_K[]
test_L.<y> = test_K.extension(y^3 + 6*x^3 + x)
test_f = test_L.hom(y^2); test_f
Function Field endomorphism of Function field in y defined by y^3 + 6*x^3 + x
  Defn: y |--> 2*y
test_sage: factor(test_L.polynomial())
y^3 + 6*x^3 + x
test_sage: test_f(y).charpoly('y')
y^3 + 6*x^3 + x
```
class sage.rings.function_field.maps.FunctionFieldMorphism_rational

Bases: sage.rings.function_field.maps.FunctionFieldMorphism

Morphism from a rational function field to a function field.

class sage.rings.function_field.maps.FunctionFieldRingMorphism

Bases: sage.categories.morphism.SetMorphism

Ring homomorphism.

class sage.rings.function_field.maps.FunctionFieldToFractionField

Bases: sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism

Isomorphism from rational function field to the isomorphic fraction field of a polynomial ring.

EXAMPLES:

```python
sage: K = QQ['x'].fraction_field()
sage: L = K.function_field()
sage: f = K.coerce_map_from(L); f
Isomorphism:
  From: Rational function field in x over Rational Field
  To:   Fraction Field of Univariate Polynomial Ring in x over Rational Field
```

See also:

FractionFieldToFunctionField

section()

Return the inverse map of this isomorphism.

EXAMPLES:

```python
sage: K = QQ['x'].fraction_field()
sage: L = K.function_field()
sage: f = K.coerce_map_from(L)
sage: f.section()
Isomorphism:
  From: Fraction Field of Univariate Polynomial Ring in x over Rational Field
  To:   Rational function field in x over Rational Field
```

class sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism

Bases: sage.categories.morphism.Morphism

A base class for isomorphisms between function fields and vector spaces.

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[

sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: V, f, t = L.vector_space()
sage: isinstance(f, sage.rings.function_field.maps.
  →FunctionFieldVectorSpaceIsomorphism)
True

is_injective()

Return True, since the isomorphism is injective.

EXAMPLES:
```
```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: V, f, t = L.vector_space()
sage: f.is_injective()
True
```

**is_surjective()**

Return True, since the isomorphism is surjective.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: V, f, t = L.vector_space()
sage: f.is_surjective()
True
```

**class** `sage.rings.function_field.maps.MapFunctionFieldToVectorSpace(K, V)`

**Bases:** `sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism`

Isomorphism from a function field to a vector space.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: V, f, t = L.vector_space(); t
Isomorphism:
  From: Function field in y defined by y^2 - x*y + 4*x^3
  To: Vector space of dimension 2 over Rational function field in x over
      Rational Field
```

**class** `sage.rings.function_field.maps.MapVectorSpaceToFunctionField(V, K)`

**Bases:** `sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism`

Isomorphism from a vector space to a function field.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: V, f, t = L.vector_space(); f
Isomorphism:
  From: Vector space of dimension 2 over Rational function field in x over
      Rational Field
  To: Function field in y defined by y^2 - x*y + 4*x^3
```

**codomain()**

Return the function field which is the codomain of the isomorphism.

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: V, f, t = L.vector_space()
sage: f.codomain()
Function field in y defined by y^2 - x*y + 4*x^3
```

**domain()**

Return the vector space which is the domain of the isomorphism.
EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: V, f, t = L.vector_space()
sage: f.domain()
Vector space of dimension 2 over Rational function field in x over Rational
```

Field
This module provides factories to construct function fields. These factories are only for internal use.

EXAMPLES:

```sage
sage: K.<x> = FunctionField(QQ); K
Rational function field in x over Rational Field
sage: L.<x> = FunctionField(QQ); L
Rational function field in x over Rational Field
sage: K is L
True
```

AUTHORS:

- William Stein (2010): initial version
- Maarten Derickx (2011-09-11): added `FunctionField_polymodConstructor`, use `@cached_function`
- Julian Rueth (2011-09-14): replaced `@cached_function` with `UniqueFactory`

```python
class sage.rings.function_field.constructor.FunctionFieldExtensionFactory
    Bases: sage.structure.factory.UniqueFactory

    Create a function field defined as an extension of another function field by adjoining a root of a univariate polynomial. The returned function field is unique in the sense that if you call this function twice with an equal polynomial and names it returns the same python object in both calls.

    INPUT:

    - `polynomial` -- univariate polynomial over a function field
    - `names` -- variable names (as a tuple of length 1 or string)
    - `category` -- category (defaults to category of function fields)

    EXAMPLES:

    ```sage
    sage: K.<x> = FunctionField(QQ)
    sage: R.<y>=K[]
    sage: y2 = y+1
    sage: y2 is y
    False
    sage: L.<w>=K.extension(x-y^2)
    sage: M.<w>=K.extension(x-y^2^2)
    sage: L is M
    True
    ```
```
create_key (polynomial, names)
Given the arguments and keywords, create a key that uniquely determines this object.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: R.<y>=K[]
sage: L.<w> = K.extension(x-y^2) # indirect doctest
```

create_object (version, key, **extra_args)
Create the object from the key and extra arguments. This is only called if the object was not found in the cache.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: R.<y>=K[]
sage: L.<w> = K.extension(x-y^2) # indirect doctest
sage: y2 = y*1
sage: M.<w> = K.extension(x-y2^2) # indirect doctest
sage: L is M
True
```

class sage.rings.function_field.constructor.FunctionFieldFactory
Bases: sage.structure.factory.UniqueFactory

Return the function field in one variable with constant field $F$. The function field returned is unique in the sense that if you call this function twice with the same base field and name then you get the same python object back.

INPUT:

- $F$ – field
- names – name of variable as a string or a tuple containing a string

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); K
Rational function field in x over Rational Field
sage: L.<y> = FunctionField(GF(7)); L
Rational function field in y over Finite Field of size 7
sage: R.<z> = L[]
sage: M.<z> = L.extension(z^7-z-y); M
Function field in z defined by z^7 + 6*z + 6*y
```

create_key ($F$, names)
Given the arguments and keywords, create a key that uniquely determines this object.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ) # indirect doctest
```

create_object (version, key, **extra_args)
Create the object from the key and extra arguments. This is only called if the object was not found in the cache.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ) # indirect doctest
sage: L.<x> = FunctionField(QQ) # indirect doctest
```
sage: K is L
True
CHAPTER
ELEVEN

INDICES AND TABLES

- Index
- Module Index
- Search Page
r
sage.rings.function_field.constructor, 127
sage.rings.function_field.differential, 109
sage.rings.function_field.divisor, 103
sage.rings.function_field.element, 43
sage.rings.function_field.function_field, 3
sage.rings.function_field.ideal, 75
sage.rings.function_field.maps, 117
sage.rings.function_field.order, 55
sage.rings.function_field.place, 97
sage.rings.function_field.valuation_ring, 113
INDEX

B
base_field() (sage.rings.function_field.function_field.FunctionField_polymod method), 19
base_field() (sage.rings.function_field.function_field.RationalFunctionField method), 33
base_ring() (sage.rings.function_field.ideal.FunctionFieldIdeal method), 76
basis() (sage.rings.function_field.differential.DifferentialsSpace method), 110
basis() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 61
basis() (sage.rings.function_field.order.FunctionFieldMaximalOrder_rational method), 65
basis() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_global method), 57
basis() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_rational method), 59
basis() (sage.rings.function_field.order.FunctionFieldOrder_basis method), 71
basis() (sage.rings.function_field.order.FunctionFieldOrderInfinite_basis method), 67
basis_differential_space() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 104
basis_function_space() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 104
basis_matrix() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 86

cartier() (sage.rings.function_field.differential.FunctionFieldDifferential_global method), 111
change_variable_name() (sage.rings.function_field.function_field.FunctionField_polymod method), 19
change_variable_name() (sage.rings.function_field.function_field.RationalFunctionField method), 34
characteristic() (sage.rings.function_field.function_field.FunctionField method), 5
characteristic_polynomial() (sage.rings.function_field.element.FunctionFieldElement method), 44
charpoly() (sage.rings.function_field.element.FunctionFieldElement method), 44
codifferent() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 61
codomain() (sage.rings.function_field.maps.MapVectorSpaceToFunctionField method), 124
completion() (sage.rings.function_field.function_field.FunctionField_global method), 13
completion() (sage.rings.function_field.function_field.RationalFunctionField_global method), 39
constant_base_field() (sage.rings.function_field.function_field.FunctionFieldPolymod method), 20
constant_base_field() (sage.rings.function_field.function_field.RationalFunctionField_global method), 34
constant_field() (sage.rings.function_field.function_field.FunctionFieldPolymod method), 20
constant_field() (sage.rings.function_field.function_field.RationalFunctionField_global method), 34
coordinate_vector() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 61
coordinate_vector() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_global method), 57
coordinate_vector() (sage.rings.function_field.order.FunctionFieldOrder BASIS method), 71
create_key() (sage.rings.function_field.constructor.FunctionFieldExtensionFactory method), 127
create_key() (sage.rings.function_field.constructor.FunctionFieldFactory method), 128
create_object() (sage.rings.function_field.constructor.FunctionFieldExtensionFactory method), 128
create_object() (sage.rings.function_field.constructor.FunctionFieldFactory method), 128

decomposition() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 61
decomposition() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_global method), 57
default_precision() (sage.rings.function_field.maps.FunctionFieldCompletion_global method), 119
degree() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 105
degree() (sage.rings.function_field.function_field.FunctionField_polymod method), 21
degree() (sage.rings.function_field.function_field.RationalFunctionField method), 34
degree() (sage.rings.function_field.place.FunctionFieldPlace_global method), 98
degree() (sage.rings.function_field.place.FunctionFieldPlace_rational method), 101
denominator() (sage.rings.function_field.element.FunctionFieldElement_rational method), 49
denominator() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 86
denominator() (sage.rings.function_field.ideal.FunctionFieldIdeal_rational method), 93
derivation() (sage.rings.function_field.function_field.FunctionField_polymod method), 21
derivation() (sage.rings.function_field.function_field.RationalFunctionField method), 35
derivative() (sage.rings.function_field.element.FunctionFieldElement method), 44
dict() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 105
different() (sage.rings.function_field.function_field.FunctionField_global method), 13
different() (sage.rings.function_field.function_field.RationalFunctionField method), 35
different() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 62
different() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_global method), 58
differential() (sage.rings.function_field.element.FunctionFieldElement method), 44
differential_space() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 105
DifferentialsSpace (class in sage.rings.function_field.differential), 109
dimension() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 105
divisor() (in module sage.rings.function_field.divisor), 107
divisor() (sage.rings.function_field.differential.FunctionFieldDifferential_global method), 111
divisor() (sage.rings.function_field.element.FunctionFieldElement_global method), 47
divisor() (sage.rings.function_field.element.FunctionFieldElement_rational method), 49
divisor() (sage.rings.function_field.ideal.FunctionFieldIdeal method), 77
divisor() (sage.rings.function_field.place.FunctionFieldPlace method), 98
divisor_group() (sage.rings.function_field.function_field.FunctionField method), 6
divisor_of_poles() (sage.rings.function_field.element.FunctionFieldElement_global method), 47
divisor_of_poles() (sage.rings.function_field.element.FunctionFieldElement_rational method), 50
divisor_of_zeros() (sage.rings.function_field.element.FunctionFieldElement_global method), 77
divisor_of_zeros() (sage.rings.function_field.element.FunctionFieldElement_rational method), 78
DivisorGroup (class in sage.rings.function_field.divisor), 103
domain() (sage.rings.function_field.maps.MapVectorSpaceToFunctionField method), 124

E

Element (sage.rings.function_field.element.DivisorGroup attribute), 104
Element (sage.rings.function_field.function_field.FunctionField_global attribute), 13
Element (sage.rings.function_field.function_field.FunctionField_polymod attribute), 19
Element (sage.rings.function_field.function_field.RationalFunctionField attribute), 33
Element (sage.rings.function_field.place.PlaceSet attribute), 102
element() (sage.rings.function_field.element.FunctionFieldElement_polymod method), 49
element() (sage.rings.function_field.element.FunctionFieldElement_rational method), 50
equation_order() (sage.rings.function_field.function_field.FunctionField_global_integral method), 17
equation_order() (sage.rings.function_field.function_field.FunctionField_polymod method), 22
equation_order() (sage.rings.function_field.function_field.RationalFunctionField method), 35
equation_order_infinite() (sage.rings.function_field.function_field.FunctionField_global_integral method), 17
equation_order_infinite() (sage.rings.function_field.function_field.RationalFunctionField method), 35
exact_constant_field() (sage.rings.function_field.function_field.FunctionField_global method), 14
extension() (sage.rings.function_field.function_field.FunctionField_field method), 6
extension() (sage.rings.function_field.function_field.RationalFunctionField method), 35

F

factor() (sage.rings.function_field.element.FunctionFieldElement_rational method), 50
factor() (sage.rings.function_field.ideal.FunctionFieldIdeal method), 78
field() (sage.rings.function_field.function_field.RationalFunctionField method), 36
fraction_field() (sage.rings.function_field.order.FunctionFieldOrder_base method), 69
FractionFieldToFunctionField (class in sage.rings.function_field_maps), 117
free_module() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 62
free_module() (sage.rings.function_field.order.FunctionFieldOrder_basis method), 71
free_module() (sage.rings.function_field.order.FunctionFieldOrderInfinite_basis method), 67
function_field() (sage.rings.function_field.differential.DifferentialsSpace method), 110
function_field() (sage.rings.function_field.divisor.DivisorGroup method), 104
function_field() (sage.rings.function_field.order.FunctionFieldOrder_base method), 69
function_field() (sage.rings.function_field.place.FunctionFieldPlace method), 98
function_space() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 106
FunctionField (class in sage.rings.function_field.function_field), 5
FunctionField_global (class in sage.rings.function_field.function_field), 12
FunctionField_global_integral (class in sage.rings.function_field.function_field), 17
FunctionField_polymod (class in sage.rings.function_field.function_field), 18
FunctionFieldCompletion (class in sage.rings.function_field_maps), 118
FunctionFieldCompletion_global (class in sage.rings.function_field_maps), 118
FunctionFieldConversionToConstantBaseField (class in sage.rings.function_field_maps), 119
FunctionFieldDerivation (class in sage.rings.function_field_maps), 119
FunctionFieldDerivation_inseparable (class in sage.rings.function_field_maps), 120
FunctionFieldDerivation_rational (class in sage.rings.function_field_maps), 120
FunctionFieldDerivation_separable (class in sage.rings.function_field_maps), 121
FunctionFieldDifferential (class in sage.rings.function_field_differential), 110
FunctionFieldDifferential_global (class in sage.rings.function_field_differential), 110
FunctionFieldDivisor (class in sage.rings.function_field_divisor), 104
FunctionFieldElement (class in sage.rings.function_field_element), 43
FunctionFieldElement_global (class in sage.rings.function_field_element), 47
FunctionFieldElement_polymod (class in sage.rings.function_field_element), 48
FunctionFieldElement_rational (class in sage.rings.function_field_element), 49
FunctionFieldExtensionFactory (class in sage.rings.function_field_constructor), 127
FunctionFieldFactory (class in sage.rings.function_field_constructor), 128
FunctionFieldHigherDerivation (class in sage.rings.function_field_maps), 121
FunctionFieldHigherDerivation_global (class in sage.rings.function_field_maps), 121
FunctionFieldHigherDerivation_rational (class in sage.rings.function_field_maps), 121
FunctionFieldIdeal (class in sage.rings.function_field_ideal), 76
FunctionFieldIdeal_global (class in sage.rings.function_field_ideal), 85
FunctionFieldIdeal_module (class in sage.rings.function_field_ideal), 91
FunctionFieldIdeal_rational (class in sage.rings.function_field.ideal), 93
FunctionFieldIdealInfinite (class in sage.rings.function_field.ideal), 80
FunctionFieldIdealInfinite_global (class in sage.rings.function_field.ideal), 81
FunctionFieldIdealInfinite_module (class in sage.rings.function_field.ideal), 83
FunctionFieldIdealInfinite_rational (class in sage.rings.function_field.ideal), 84
FunctionFieldLinearMap (class in sage.rings.function_field.maps), 122
FunctionFieldLinearMapSection (class in sage.rings.function_field.maps), 122
FunctionFieldMaximalOrder (class in sage.rings.function_field.order), 56
FunctionFieldMaximalOrder_global (class in sage.rings.function_field.order), 61
FunctionFieldMaximalOrder_rational (class in sage.rings.function_field.order), 64
FunctionFieldMaximalOrderInfinite (class in sage.rings.function_field.order), 56
FunctionFieldMaximalOrderInfinite_global (class in sage.rings.function_field.order), 56
FunctionFieldMaximalOrderInfinite_rational (class in sage.rings.function_field.order), 59
FunctionFieldMorphism (class in sage.rings.function_field.maps), 122
FunctionFieldMorphism_polymod (class in sage.rings.function_field.maps), 122
FunctionFieldMorphism_rational (class in sage.rings.function_field.maps), 122
FunctionFieldOrder (class in sage.rings.function_field.order), 66
FunctionFieldOrder_base (class in sage.rings.function_field.order), 69
FunctionFieldOrder_basis (class in sage.rings.function_field.order), 70
FunctionFieldOrderInfinite (class in sage.rings.function_field.order), 66
FunctionFieldOrderInfinite_basis (class in sage.rings.function_field.order), 66
FunctionFieldPlace (class in sage.rings.function_field.place), 97
FunctionFieldPlace_global (class in sage.rings.function_field.place), 98
FunctionFieldPlace_rational (class in sage.rings.function_field.place), 101
FunctionFieldRingMorphism (class in sage.rings.function_field.maps), 123
FunctionFieldToFractionField (class in sage.rings.function_field.maps), 123
FunctionFieldValuationRing (class in sage.rings.function_field.valuation_ring), 114
FunctionFieldVectorSpaceIsomorphism (class in sage.rings.function_field.maps), 123

gaps() (sage.rings.function_field.function_field.FunctionField_global method), 14
gaps() (sage.rings.function_field.function_field.FunctionFieldPlace_global method), 99
gen() (sage.rings.function_field.function_field.FunctionField_polymod method), 22
gen() (sage.rings.function_field.function_field.RationalFunctionField method), 36
gen() (sage.rings.function_field.ideal.FunctionFieldIdeal_module method), 91
gen() (sage.rings.function_field.ideal.FunctionFieldIdeal_rational method), 93
gen() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_rational method), 84
gen() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 62
gen() (sage.rings.function_field.order.FunctionFieldMaximalOrder_rational method), 65
gen() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_global method), 58
gen() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_rational method), 60
gens() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 86
gens() (sage.rings.function_field.ideal.FunctionFieldIdeal_module method), 92
gens() (sage.rings.function_field.ideal.FunctionFieldIdeal_rational method), 93
gens() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_global method), 81
gens() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_rational method), 85
gens_over_base() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 87
gens_over_base() (sage.rings.function_field.ideal.FunctionFieldIdeal_rational method), 94
gens_over_base() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_global method), 81
is_prime() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 89
is_prime() (sage.rings.function_field.ideal.FunctionFieldIdeal_rational method), 94
is_prime() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_global method), 82
is_prime() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_rational method), 85
is_RationalFunctionField() (in module sage.rings.function_field.function_field), 41
is_separable() (sage.rings.function_field.function_field.FunctionField_polymod method), 25
is_square() (sage.rings.function_field.element.FunctionFieldElement_rational method), 51
is_subring() (sage.rings.function_field.order.FunctionFieldOrder_base method), 70
is_surjective() (sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism method), 124

L
list() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 106
list() (sage.rings.function_field.element.FunctionFieldElement_polymod method), 49
list() (sage.rings.function_field.element.FunctionFieldElement_rational method), 51
local_uniformizer() (sage.rings.function_field.place.FunctionFieldPlace_global method), 99
local_uniformizer() (sage.rings.function_field.place.FunctionFieldPlace_rational method), 101

M
make_FunctionFieldElement() (in module sage.rings.function_field.element), 53
MapFunctionFieldToVectorSpace (class in sage.rings.function_field.maps), 124
MapVectorSpaceToFunctionField (class in sage.rings.function_field.maps), 124
matrix() (sage.rings.function_field.element.FunctionFieldElement method), 45
maximal_order() (sage.rings.function_field.function_field.FunctionField_global method), 15
maximal_order() (sage.rings.function_field.function_field.FunctionField_polymod method), 25
maximal_order() (sage.rings.function_field.function_field.FunctionFieldInfinite_global method), 15
maximal_order() (sage.rings.function_field.function_field.FunctionFieldInfinite_rational method), 37
maximal_order_infinite() (sage.rings.function_field.function_field.FunctionField_global method), 25
maximal_order_infinite() (sage.rings.function_field.function_field.FunctionField_polymod method), 25
maximal_order_infinite() (sage.rings.function_field.function_field.FunctionFieldInfinite_global method), 37
minimal_polynomial() (sage.rings.function_field.element.FunctionFieldElement method), 46
minpoly() (sage.rings.function_field.element.FunctionFieldElement method), 46
module() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 89
module() (sage.rings.function_field.ideal.FunctionFieldIdeal_module method), 92
module() (sage.rings.function_field.ideal.FunctionFieldIdeal_rational method), 94
module() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_module method), 84
monic_integral_model() (sage.rings.function_field.function_field.FunctionField_polymod method), 25
monomial_coefficients() (sage.rings.function_field.differential.FunctionFieldDifferential_global method), 111
multiplicity() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 106

N
ngens() (sage.rings.function_field.function_field.FunctionField_polymod method), 26
ngens() (sage.rings.function_field.function_field.RationalFunctionField method), 38
ngens() (sage.rings.function_field.ideal.FunctionFieldIdeal_module method), 93
ngens() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 64
ngens() (sage.rings.function_field.order.FunctionFieldMaximalOrder_rational method), 66
ngens() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_global method), 59
ngens() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_rational method), 60
norm() (sage.rings.function_field.element.FunctionFieldElement method), 47
norm() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 89
numerator() (sage.rings.function_field.element.FunctionFieldElement_rational method), 52
O

order() (sage.rings.function_field.function_field.FunctionField method), 7
order_infinite() (sage.rings.function_field.function_field.FunctionField method), 7
order_infinite_with_basis() (sage.rings.function_field.function_field.FunctionField method), 8
order_with_basis() (sage.rings.function_field.function_field.FunctionField method), 9

P

p_radical() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 64
place() (sage.rings.function_field.ideal.FunctionFieldIdeal method), 79
place() (sage.rings.function_field.ideal.valuation_ring.FunctionFieldValuationRing method), 114
place_below() (sage.rings.function_field.place.FunctionFieldPlace_global method), 99
place_infinite() (sage.rings.function_field.function_field.RationalFunctionField_global method), 40
place_set() (sage.rings.function_field.function_field.FunctionField_global method), 15
place_set() (sage.rings.function_field.function_field.RationalFunctionField_global method), 38
places() (sage.rings.function_field.function_field.FunctionField_global method), 15
places() (sage.rings.function_field.function_field.RationalFunctionField_global method), 40
places_finite() (sage.rings.function_field.function_field.FunctionField_global method), 16
places_finite() (sage.rings.function_field.function_field.RationalFunctionField_global method), 40
places_infinite() (sage.rings.function_field.function_field.FunctionField_global method), 16
PlaceSet (class in sage.rings.function_field.place), 102
polynomial() (sage.rings.function_field.function_field.FunctionField_function_field_polymod method), 27
polynomial() (sage.rings.function_field.order.FunctionFieldMaximalOrder_global method), 64
polynomial() (sage.rings.function_field.order.FunctionFieldOrder_basis method), 73
polynomial() (sage.rings.function_field.order.FunctionFieldOrderInfinite_basis method), 68
polynomial_ring() (sage.rings.function_field.function_field.FunctionField_function_field_polymod method), 27
polynomial_ring() (sage.rings.function_field.function_field.RationalFunctionField_function_field_polymod method), 38
prime_below() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 90
prime_below() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_global method), 82
prime_divisor() (in module sage.rings.function_field.divisor), 107
prime_ideal() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_rational method), 60
prime_ideal() (sage.rings.function_field.place.FunctionFieldPlacePlace method), 98
primitive_element() (sage.rings.function_field.function_field.FunctionField_function_field_polymod method), 27
primitive_integral_element_infinite() (sage.rings.function_field.function_field.FunctionField_global_integral method), 18

R

random_element() (sage.rings.function_field.function_field.FunctionField_function_field_polymod method), 28
random_element() (sage.rings.function_field.function_field.RationalFunctionField_method), 38
RationalFunctionField (class in sage.rings.function_field.function_field), 33
RationalFunctionField_global (class in sage.rings.function_field.function_field), 39
relative_degree() (sage.rings.function_field.place.FunctionFieldPlace_global method), 100
residue() (sage.rings.function_field.differential.FunctionFieldDifferential_global method), 112
residue_field() (sage.rings.function_field.function_field.FunctionField_function_field.global_method), 16
residue_field() (sage.rings.function_field.function_field.RationalFunctionField_global method), 40
residue_field() (sage.rings.function_field.place.FunctionFieldPlace_global method), 100
residue_field() (sage.rings.function_field.place.FunctionFieldPlace_rational method), 101
residue_field() (sage.rings.function_field.valuation_ring.FunctionFieldValuationRing_global method), 114
ring() (sage.rings.function_field.ideal.FunctionFieldIdeal_method), 80
ring() (sage.rings.function_field.ideal.IdealMonoid_method), 95
S
sage.rings.function_field.constructor (module), 127
sage.rings.function_field.differential (module), 109
sage.rings.function_field.divisor (module), 103
sage.rings.function_field.element (module), 43
sage.rings.function_field.function_field (module), 3
sage.rings.function_field.ideal (module), 75
sage.rings.function_field.maps (module), 117
sage.rings.function_field.order (module), 55
sage.rings.function_field.place (module), 97
sage.rings.function_field.valuation_ring (module), 113
section() (sage.rings.function_field.maps.FractionFieldToFunctionField method), 118
section() (sage.rings.function_field.maps.FunctionFieldToFractionField method), 123
separable_model() (sage.rings.function_field.function_field.FunctionField_polymod method), 28
simple_model() (sage.rings.function_field.function_field.FunctionField_polymod method), 29
some_elements() (sage.rings.function_field.function_field.FunctionField method), 10
space_of_differentials() (sage.rings.function_field.function_field.FunctionField method), 10
sqrt() (sage.rings.function_field.element.FunctionFieldElement_rational method), 52
support() (sage.rings.function_field.divisor.FunctionFieldDivisor method), 107

T
trace() (sage.rings.function_field.element.FunctionFieldElement method), 47

V
valuation() (sage.rings.function_field.element.FunctionFieldElement_global method), 48
valuation() (sage.rings.function_field.element.FunctionFieldElement_rational method), 52
valuation() (sage.rings.function_field.function_field.FunctionField method), 11
valuation() (sage.rings.function_field.ideal.FunctionFieldIdeal_global method), 90
valuation() (sage.rings.function_field.ideal.FunctionFieldIdeal_rational method), 94
valuation() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_global method), 83
valuation() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite_rational method), 85
valuation_ring() (sage.rings.function_field.place.FunctionFieldPlace_global method), 100
valuation_ring() (sage.rings.function_field.place.FunctionFieldPlace_rational method), 102
vector_space() (sage.rings.function_field.function_field.FunctionField_polymod method), 31
vector_space() (sage.rings.function_field.function_field.RationalFunctionField method), 38

W
weierstrass_places() (sage.rings.function_field.function_field.FunctionField_global method), 17