# Sage Reference Manual: Algebraic Function Fields

*Release 8.2*

**The Sage Development Team**

**May 06, 2018**

# CONTENTS


| | |
|---|---|
| 1 Function Fields | 1 |
| 2 Function Field Elements | 27 |
| 3 Orders in Function Fields | 35 |
| 4 Ideals in Function Fields | 41 |
| 5 Function Field Morphisms | 45 |
| 6 Factories to construct Function Fields | 51 |
| 7 Indices and Tables | 53 |
| Python Module Index | 55 |
| Index | 57 |

# CHAPTER ONE

# FUNCTION FIELDS

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.

EXAMPLES:

We create an extension of a rational function fields, and do some simple arithmetic in it:

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

We next make an extension of the above function field, illustrating that arithmetic with a tower of 3 fields is fully supported:

```
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
sage: 1/t
((1/(x^4 + 1))*y^2 + 2*x/(4*x^4 + 4))*t
sage: M.base_field()
Function field in y defined by y^3 + 3*x*y + (4*x^4 + 4)/x
```

```
sage: M.base_field().base_field()
Rational function field in x over Finite Field in a of size 5^2
```

It is also possible to construct function fields over an imperfect base field:

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

and function fields as inseparable extensions:

```
sage: R.<v> = K[]
sage: O.<v> = K.extension(v^5 - x)
```

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

Bases: `sage.rings.ring.Field`

The abstract base class for all function fields.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: isinstance(K, sage.rings.function_field.function_field.FunctionField)
True
```

**`characteristic`**()

Return the characteristic of this function field.

EXAMPLES:

```
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
```

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

Create an extension L = K[y]/(f(y)) of a function field, defined by a univariate polynomial in one variable over this function field K.

INPUT:

- `f` – a univariate polynomial over self
- `names` – None or 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)
```

**is_finite**()

Return whether this function field is finite, which it is not.

EXAMPLES:

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

**is_perfect**()

Return whether this field is perfect, i.e., its characteristic is $p = 0$ or every element has a $p$-th root.

EXAMPLES:

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

**order**(*x*, *check=True*)

Return the order in this function field generated over the maximal order by x or the elements of x if x is a list.

INPUT:

- `x` – element of self, or a list of elements of self
- `check` – bool (default: True); if True, check that x really generates an order

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ); R.<y> = K[]; 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, not yet supported:

```
sage: Z = K.order([x,x^2]); Z
Traceback (most recent call last):
...
NotImplementedError
```

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

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

INPUT:

- `basis` – a list of elements of self
- `check` – bool (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 this function field

EXAMPLES:

```
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: 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: 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: O = L.order_with_basis([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:

```
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 elemnts 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),
 x/(2*x^2 + 2),
 0,
 1/x,
 ...]
```

```
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: L.some_elements()
[1,
 y,
 1/x*y,
 ((1/4*x + 1/4)/(1/4*x^2 - 1/2*x + 1/4))*y - 1/2*x/(1/4*x^2 - 1/2*x + 1/4),
 -1/-x,
 (1/(x - 1))*y,
 (1/(x + 1))*y,
 (1/(2*x + 2))*y,
 0,
 ...]
```

**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:

```
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()]))); v
Valuation on rational function field induced by [ Gauss valuation induced by␣
↪2-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_polymod`**(*polynomial*, *names*, *element_class=<type 'sage.rings.function_field.functi*, *category=Category of function fields*)

Bases: *sage.rings.function_field.function_field.FunctionField*

A function field defined by a univariate polynomial, as an extension of the base field.

EXAMPLES:

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

```
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:

```
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
sage: 1/z
((x/(-x^4 - 1))*y^4 - 2*x^2/(-x^4 - 1))*z - 1
sage: z * (1/z)
1
```

We drill down the tower of function fields:

```
sage: M.base_field()
Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x
sage: M.base_field().base_field()
Rational function field in x over Rational Field
sage: M.base_field().base_field().constant_field()
Rational Field
sage: M.constant_base_field()
Rational Field
```

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

```
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
```

**base_field**()

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

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.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`.

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: 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 |--> x)
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)
```

**constant_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
```

**constant_field**()

Return the algebraic closure of the constant field of the base field in this 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=None*)

Return the degree of this function field over the function field `base`.

INPUT:

- `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 derivation of the function field over the constant base field.

A derivation on $R$ is a map $R \to R$ satisfying $D(\alpha+\beta) = D(\alpha) + D(\beta)$ and $D(\alpha\beta) = \beta D(\alpha) + \alpha D(\beta)$ for all $\alpha, \beta \in R$. For a function field which is a finite extension of $K(x)$ with $K$ perfect, the derivations form a one-dimensional $K$-vector space generated by the derivation returned by this method.

OUTPUT:

- a derivation of the function field

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(3))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: 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
sage: d(x)
1
sage: d(x^3)
0
sage: d(x*y)
0
sage: d(y)
2/x*y
```

Derivations are linear and satisfy Leibniz's law:

```
sage: d(x+y) == d(x) + d(y)
True
sage: d(x*y) == x*d(y) + y*d(x)
True
```

If the field is a separable extension of the base field, the derivation extending a derivation of the base function field is uniquely determined. Proposition 11 of [GT1996] describes how to compute the extension. We apply the formula described there to the generator of the space of derivations on the base field.

The general inseparable case is not implemented yet (see trac ticket #16562, trac ticket #16564.)'

**equation_order**()

If we view self as being presented as K[y]/(f(y)), then this function returns the order generated by the class of y. 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:

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

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

```
sage: K.<x> = FunctionField(QQ); 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: O = L.equation_order()
sage: O.basis()
(1, x^3*y, x^6*y^2, x^9*y^3, x^12*y^4)
```

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

Return the `n`-th generator of this 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 self as being presented as 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.gen()
y
sage: L.gen(1)
```

```
Traceback (most recent call last):
...
IndexError: Only one generator.
```

**genus**()

Return the genus of this function field For now, the genus is computed using singular

EXAMPLES:

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

**hom**(*im_gens*, *base_morphism=None*)

Create a homomorphism from self to another function field.

INPUT:

- `im_gens` – a list of images of the generators of self and of successive base rings.
- `base_morphism` – (default: None) a homomorphism of the base ring, after the im_gens are used. Thus if im_gens has length 2, then base_morphism should be a morphism from self.base_ring().base_ring().

EXAMPLES:

We create a rational function field, and a quadratic extension of it:

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

We make the field automorphism that sends y to -y:

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

Evaluation works:

```
sage: f(y*x - 1/x)
-x*y - 1/x
```

We try to define an invalid morphism:

```
sage: f = L.hom(y+1)
Traceback (most recent call last):
...
ValueError: invalid morphism
```

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

Return whether the defining polynomial of the function field is separable, i.e., whether the gcd of the defining polynomial and its derivative is constant.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(5)); 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)); R.<y> = K[]
sage: L.<y> = K.extension(y^5 - 1)
sage: L.is_separable()
False
```

**maximal_order**()

Return the maximal_order of self. If we view self as L = K[y]/(f(y)), then this is the ring of elements of L that are integral over K.

EXAMPLES:

This is not yet implemented…:

```
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()
Traceback (most recent call last):
...
NotImplementedError
```

**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:

```
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:

```
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 this 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 this function field, i.e., the polynomial f(y) so that this 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 this function field. If we view this 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:

```
sage: K.<x> = FunctionField(GF(2))
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: M.primitive_element()
z
```

**random_element**(**args*, ***kwds*)

Create a random element of this function field. Parameters are passed onto the random_element method of the base_field.

EXAMPLES:

```
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)
```

**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:

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

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^3 + 2*x
  Defn: z |--> z + y,
 Function Field morphism:
  From: Function field in z defined by z^3 + 2*x
  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: z |--> 2/x*z^6 + 2*z^3 + z + 2*x
        y |--> 1/x*z^6 + z^3 + x
        x |--> 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 |--> 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 |--> z
        y |--> z^2
        x |--> x)
```

**vector_space** (*base=None*)

Return a vector space $V$ and isomorphisms from this field to $V$ and from $V$ to this field.

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:

- `V` – a vector space over base field
- `from_V` – an isomorphism from V to this field
- `to_V` – an isomorphism from this field to V

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 Rational
↪Field
sage: from_V
Isomorphism morphism:
```

```
  From: Vector space of dimension 5 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
sage: to_V
Isomorphism morphism:
  From: Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x
  To:   Vector space of dimension 5 over Rational function field in x over␣
↪Rational Field
```

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 morphism:
  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 morphism:
  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 morphism:
  From: Vector space of dimension 10 over Rational function field in x over␣
↪Rational Field
  To:   Function field in z defined by z^2 - y, Isomorphism morphism:
  From: Function field in z defined by z^2 - y
  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*, *element_class=<type 'sage.rings.function_field.functi*, *category=Category of function fields*)

Bases: *sage.rings.function_field.function_field.FunctionField*

A rational function field K(t) in one variable, over an arbitrary base field.

EXAMPLES:

```
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:

```
sage: K.<t> = FunctionField(GF(7))
sage: K.base_field()
Rational function field in t over Finite Field of size 7
sage: K.field()
Fraction Field of Univariate Polynomial Ring in t over Finite Field of size 7
sage: K.constant_field()
Finite Field of size 7
sage: K.maximal_order()
Maximal order in Rational function field in t over Finite Field of size 7
```

We define a morphism:

```
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
```

**base_field**()

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

EXAMPLES:

```
sage: K.<t> = FunctionField(GF(7))
sage: K.base_field()
Rational function field in t over Finite Field of size 7
```

**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:

```
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 that this rational function field is a transcendental extension of.

EXAMPLES:

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

**constant_field**()

Return the field that this rational function field is a transcendental extension of.

EXAMPLES:

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

**degree**(*base=None*)

Return the degree over the base field of this 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:

```
sage: K.<t> = FunctionField(QQ)
sage: K.degree()
1
```

**derivation**()

Return a derivation of the rational function field over the constant base field.

OUTPUT:

- a derivation of the rational function 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
sage: m(x)
1
```

**equation_order**()

Return the maximal order of this 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].

EXAMPLES:

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

**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 this 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 this function field This is always equal 0 for a rational function field

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ);
sage: K.genus()
0
```

**hom**(*im_gens*, *base_morphism=None*)

Create a homomorphism from self to another ring.

INPUT:

- **im_gens – exactly one element of some ring. It must be invertible and transcendental over** the image of base_morphism; this is not checked.
- **base_morphism – a homomorphism from the base field into the other ring.** If None, try to use a coercion map.

OUTPUT:

- a map between function fields

EXAMPLES:

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

```
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
```

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

```
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
```

**maximal_order**()

Return the maximal order of this 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].

EXAMPLES:

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

**polynomial_ring**(*var='x'*)

Return a polynomial ring in one variable over this rational function field.

INPUT:

- `var` – a string (default: 'x')

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 this 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 this field to $V$ and from $V$ to this 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)

OUTPUT:

- `V` – a vector space over base field
- `from_V` – an isomorphism from V to this field
- `to_V` – an isomorphism from this field to V

EXAMPLES:

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

`sage.rings.function_field.function_field.`**is_FunctionField**(*x*)

Return True if `x` is of function field type.

EXAMPLES:

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

sage.rings.function_field.function_field.**is_RationalFunctionField**(*x*)

Return `True` if `x` is of rational function field type.

EXAMPLES:

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

# CHAPTER TWO

# FUNCTION FIELD ELEMENTS

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

**class** sage.rings.function_field.function_field_element.**FunctionFieldElement**

Bases: sage.structure.element.FieldElement

The abstract base class for function field elements.

EXAMPLES:

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

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

Return the characteristic polynomial of this function field 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 this function field 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
```

**is_integral**()

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

EXAMPLES:

```
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:

```
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:

```
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:

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

```
sage: M.<T> = L[]
sage: Z.<alpha> = L.extension(T^3 - y^2*T + x)
sage: alpha.matrix()
[          0           1           0]
[          0           0           1]
[         -x x*y - 4*x^3           0]
sage: alpha.matrix(K)
[           0            0            1            0            0            ␣
↪0]
[           0            0            0            1            0            ␣
↪0]
[           0            0            0            0            1            ␣
↪0]
[           0            0            0            0            0            ␣
↪1]
[          -x            0       -4*x^3            x            0            ␣
↪0]
[           0           -x       -4*x^4 -4*x^3 + x^2            0            ␣
↪0]
sage: alpha.matrix(Z)
[alpha]
```

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 this function field 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 this function field 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
```

**norm**()

Return the norm of this function field 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 this function field 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.function_field_element.**FunctionFieldElement_polymod**

Bases: *sage.rings.function_field.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)
sage: x*y + 1/x^3
x*y + 1/x^3
```

**element**()

Return the underlying polynomial that represents this element.

**EXAMPLES::** 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 1/x^2*y + x/(x^2 + 1) sage: f.element() 1/x^2*y + x/(x^2 + 1) sage: type(f.element()) <class 'sage.rings.polynomial.polynomial_ring.PolynomialRing_field_with_category.element_class'>

**list**()

Return a list of coefficients of self, i.e., if self is an element of a function field K[y]/(f(y)), then return the coefficients of the reduced presentation as a polynomial in K[y].

EXAMPLES:

```
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
(-x^2/(8*x^5 + x^2 + 1/2))*y + (2*x^3 + x)/(16*x^5 + 2*x^2 + 1)
sage: a.list()
[(2*x^3 + x)/(16*x^5 + 2*x^2 + 1), -x^2/(8*x^5 + x^2 + 1/2)]
sage: (x*y).list()
[0, x]
```

**class** sage.rings.function_field.function_field_element.**FunctionFieldElement_rational**

Bases: *sage.rings.function_field.function_field_element.FunctionFieldElement*

Elements of a rational function field.

EXAMPLES:

```
sage: K.<t> = FunctionField(QQ); K
Rational function field in t over Rational Field
```

**denominator**()

EXAMPLES:

```
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
```

**element**()

Return the underlying fraction field element that represents this element.

EXAMPLES:

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

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

**factor**()

Factor this rational function.

EXAMPLES:

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

```
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
```

**inverse_mod**($I$)

Return an inverse of self modulo the integral ideal $I$, if defined, i.e., if $I$ and self together generate the unit ideal.

EXAMPLES:

```
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**()

Returns whether self is a square.

EXAMPLES:

```
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 of coefficients of self, i.e., if self is an element of a function field K[y]/(f(y)), then return the coefficients of the reduced presentation as a polynomial in K[y]. Since self is a member of a rational function field, this simply returns the list $[self]$

EXAMPLES:

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

**numerator**()

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*)

Returns the square root of self.

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** (*v*)

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.rings.function_field.function_field_element.**is_FunctionFieldElement** (*x*)

Return True if x is any type of function field element.

EXAMPLES:

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

sage.rings.function_field.function_field_element.**make_FunctionFieldElement** (*parent, element_class, representing_element*)

Used for unpickling FunctionFieldElement objects (and subclasses).

EXAMPLES:

```
sage: from sage.rings.function_field.function_field_element import make_
↪FunctionFieldElement
sage: K.<x> = FunctionField(QQ)
sage: make_FunctionFieldElement(K, K._element_class, (x+1)/x)
(x + 1)/x
```

CHAPTER

THREE

# ORDERS IN FUNCTION FIELDS

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_

EXAMPLES:

Maximal orders in rational function fields:

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

Equation orders in extensions of rational function fields:

```
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
```

**class** sage.rings.function_field.function_field_order.**FunctionFieldOrder**(*fraction_field*)

Bases: sage.rings.ring.IntegralDomain

Base class for orders in function fields.

**fraction_field**()

Returns the function field in which this is an order.

EXAMPLES:

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

**function_field**()

Returns the function field in which this is an order.

EXAMPLES:

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

**ideal**(**gens*)

Returns the fractional ideal generated by the elements in `gens`.

INPUT:

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

EXAMPLES:

```
sage: K.<y> = FunctionField(QQ)
sage: O = K.maximal_order()
sage: O.ideal(y)
Ideal (y) of Maximal order in 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:

```
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.equation_order()
sage: S.ideal(1/y)
Ideal (1, (6/(x^3 + 1))*y) of Order in Function field in y defined by y^2 +␣
↪6*x^3 + 6
sage: I2 = S.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
```

**ideal_with_gens_over_base**(*gens*)

Returns the fractional ideal with basis `gens` over the maximal order of the base field. That this is really an ideal is not checked.

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_with_gens_over_base([y]); I
Ideal (y) of Maximal order in Rational function field in y over Rational Field
sage: I*I
Ideal (y^2) of Maximal order in 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 in 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
```

**is_field**(*proof=True*)

Returns False since orders are never fields.

EXAMPLES:

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

**is_finite**()

Returns False since orders are never finite.

EXAMPLES:

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

**is_noetherian**()

Returns True since orders in function fields are noetherian.

EXAMPLES:

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

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

Bases: *sage.rings.function_field.function_field_order.FunctionFieldOrder*

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

**basis**()

Returns a basis of self 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.equation_order()
sage: O.basis()
(1, y, y^2, y^3)
```

**fraction_field()**

Returns the function field in 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.fraction_field()
Function field in y defined by y^4 + x*y + 4*x + 1
```

**free_module()**

Returns 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.equation_order()
sage: O.free_module()
Free module of degree 4 and rank 4 over Maximal order in 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]
```

**polynomial()**

Returns 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.function_field_order.**FunctionFieldOrder_rational**(*function_field*

Bases: sage.rings.ring.PrincipalIdealDomain, *sage.rings.function_field.function_field_order.FunctionFieldOrder*

The maximal order in a rational function field.

**basis()**

Returns the basis (=1) for this order as a module over the polynomial ring.

EXAMPLES:

```
sage: K.<t> = FunctionField(GF(19))
sage: O = K.maximal_order()
sage: O.basis()
(1,)
sage: parent(O.basis()[0])
Maximal order in Rational function field in t over Finite Field of size 19
```

```
```

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

Returns the n-th generator of self. Since there is only one generator n must be 0.

EXAMPLES:

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

**ideal** (**gens*)

Returns the fractional ideal generated by gens.

EXAMPLES:

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

**ngens** ()

Returns 1, the number of generators of self.

EXAMPLES:

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

CHAPTER

FOUR

# IDEALS IN FUNCTION FIELDS

AUTHORS:

- William Stein (2010): initial version
- Maarten Derickx (2011-09-14): fixed ideal_with_gens_over_base()

EXAMPLES:

Ideals in the maximal order of a rational function field:

```
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 in Rational function field in x over Rational Field
sage: I^2
Ideal (x^6 + 2*x^3 + 1) of Maximal order in Rational function field in x over␣
↪Rational Field
sage: ~I
Ideal (1/(x^3 + 1)) of Maximal order in Rational function field in x over Rational␣
↪Field
sage: ~I * I
Ideal (1) of Maximal order in Rational function field in x over Rational Field
```

Ideals in the equation order of 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.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
sage: ~I
Ideal (-1, (1/(x^3 + 1))*y) of Order in Function field in y defined by y^2 - x^3 - 1
sage: ~I * I
Ideal (1, y) of Order in Function field in y defined by y^2 - x^3 - 1
sage: I.intersection(~I)
Ideal (x^3 + 1, -y) of Order in Function field in y defined by y^2 - x^3 - 1
```

**class** sage.rings.function_field.function_field_ideal.**FunctionFieldIdeal**(*ring*, *gens*, *coerce=True*)

Bases: sage.rings.ideal.Ideal_generic

A fractional ideal of a function field.

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(7))
sage: O = K.maximal_order()
sage: I = O.ideal(x^3+1)
sage: isinstance(I, sage.rings.function_field.function_field_ideal.
↪FunctionFieldIdeal)
True
```

**class** sage.rings.function_field.function_field_ideal.**FunctionFieldIdeal_module**(*ring*, *module*)

Bases: *sage.rings.function_field.function_field_ideal.FunctionFieldIdeal*

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

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.equation_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
```

**intersection**(*other*)

Return the intersection of the ideals self and `other`.

EXAMPLES:

```
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(y^3); J = O.ideal(y^2)
sage: Z = I.intersection(J); Z
Ideal (x^6 + 2*x^3 + 1, (6*x^3 + 6)*y) of Order in Function field in y␣
↪defined by y^2 + 6*x^3 + 6
sage: y^2 in Z
False
sage: y^3 in Z
True
```

**module**()

Return module over the maximal order of the base field that underlies self.

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

OUTPUT:

- a module over the maximal order of the base field of self

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(7))
sage: O = K.maximal_order(); O
Maximal order in 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_with_gens_over_base([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 in Rational function␣
↪field in x over Finite Field of size 7
User 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
```

`sage.rings.function_field.function_field_ideal.`**`ideal_with_gens`**`(`*R*, *gens*`)`

Return fractional ideal in the order `R` with generators `gens` over `R`.

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: sage.rings.function_field.function_field_ideal.ideal_with_gens(O, [y])
Ideal (x^3 + 1, -y) of Order in Function field in y defined by y^2 - x^3 - 1
```

`sage.rings.function_field.function_field_ideal.`**`ideal_with_gens_over_base`**`(`*R*, *gens*`)`

Return fractional ideal in the order `R` with generators `gens` over the maximal order of the base 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: sage.rings.function_field.function_field_ideal.ideal_with_gens_over_base(O,␣
↪[x^3+1,-y])
Ideal (x^3 + 1, -y) of Order in Function field in y defined by y^2 - x^3 - 1
```

CHAPTER

FIVE

# FUNCTION FIELD MORPHISMS

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

EXAMPLES:

```
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
```

**class** sage.rings.function_field.maps.**FractionFieldToFunctionField**

Bases: *sage.rings.function_field.maps.FunctionFieldIsomorphism*

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 morphism:
    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 morphism:
    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.**FunctionFieldConversionToConstantBaseField**(*parent*)

Bases: sage.categories.map.Map

Conversion map from the function field to its constant base field.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
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*)

Bases: sage.categories.map.Map

A base class for derivations on function fields.

A derivation on $R$ is 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:

```
sage: K.<x> = FunctionField(QQ)
sage: d = K.derivation()
sage: isinstance(d, sage.rings.function_field.maps.FunctionFieldDerivation)
True
```

**is_injective**()

Return whether this derivation is injective.

OUTPUT:

Returns `False` since derivations are never injective.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: d = K.derivation()
sage: d.is_injective()
False
```

**class** sage.rings.function_field.maps.**FunctionFieldDerivation_rational**(*K*, *u*)

Bases: *sage.rings.function_field.maps.FunctionFieldDerivation*

A derivation on a rational function field.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: d = K.derivation()
sage: isinstance(d, sage.rings.function_field.maps.FunctionFieldDerivation_
↪rational)
True
```

**class** sage.rings.function_field.maps.**FunctionFieldDerivation_separable**(*L*, *d*)

Bases: *sage.rings.function_field.maps.FunctionFieldDerivation*

The unique extension of the derivation `d` to `L`.

EXAMPLES:

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

**class** sage.rings.function_field.maps.**FunctionFieldIsomorphism**

Bases: sage.categories.morphism.Morphism

A base class for isomorphisms involving function fields.

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: isinstance(f, sage.rings.function_field.maps.FunctionFieldIsomorphism)
True
```

**is_injective**()

Return True, since this isomorphism is injective.

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.is_injective()
True
```

**is_surjective**()

Return True, since this isomorphism is surjective.

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.is_surjective()
True
```

**class** sage.rings.function_field.maps.**FunctionFieldMorphism**(*parent*, *im_gen*, *base_morphism*)

Bases: sage.rings.morphism.RingHomomorphism

Base class for morphisms between function fields.

**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: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: f = L.hom(-y); f
Function Field endomorphism of Function field in y defined by y^2 - x
  Defn: y |--> -y
```

**class** sage.rings.function_field.maps.**FunctionFieldMorphism_rational**(*parent*, *im_gen*, *base_morphism*)

Bases: *sage.rings.function_field.maps.FunctionFieldMorphism*

Morphism from a rational function field to a function field.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: f = K.hom(1/x); f
Function Field endomorphism of Rational function field in x over Rational Field
  Defn: x |--> 1/x
```

**class** sage.rings.function_field.maps.**FunctionFieldToFractionField**

Bases: *sage.rings.function_field.maps.FunctionFieldIsomorphism*

Isomorphism from rational function field to the isomorphic fraction field of a polynomial ring.

EXAMPLES:

```
sage: K = QQ['x'].fraction_field()
sage: L = K.function_field()
sage: f = K.coerce_map_from(L); f
Isomorphism morphism:
  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:

```
sage: K = QQ['x'].fraction_field()
sage: L = K.function_field()
sage: f = K.coerce_map_from(L)
sage: f.section()
Isomorphism morphism:
    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.**MapFunctionFieldToVectorSpace**(*K*, *V*)

Bases: *sage.rings.function_field.maps.FunctionFieldIsomorphism*

An isomorphism from a function field to a vector space.

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(); t
Isomorphism morphism:
  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.FunctionFieldIsomorphism*

An isomorphism from a vector space to a function field.

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(); f
Isomorphism morphism:
  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
```

CHAPTER

SIX

# FACTORIES TO CONSTRUCT FUNCTION FIELDS

AUTHORS:

- William Stein (2010): initial version
- Maarten Derickx (2011-09-11): added `FunctionField_polymod_Constructor`, use `@cached_function`
- Julian Rueth (2011-09-14): replaced `@cached_function` with `UniqueFactory`

EXAMPLES:

```
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
```

**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` – a 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)
sage: L.<x> = FunctionField(QQ)
sage: K is L
True
```

**class** sage.rings.function_field.constructor.**FunctionFieldPolymodFactory**

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` – a univariate polynomial over a function field
- `names` – variable names (as a tuple of length 1 or string)
- `category` – a category (defaults to category of function fields)

EXAMPLES:

```
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-y2^2)
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
```

CHAPTER

# SEVEN

# INDICES AND TABLES


- Index
- Module Index
- Search Page

# PYTHON MODULE INDEX

## r


sage.rings.function_field.constructor, 51
sage.rings.function_field.function_field, 1
sage.rings.function_field.function_field_element, 27
sage.rings.function_field.function_field_ideal, 41
sage.rings.function_field.function_field_order, 35
sage.rings.function_field.maps, 45

# INDEX

## B

base_field() (sage.rings.function_field.function_field.FunctionField_polymod method), 8
base_field() (sage.rings.function_field.function_field.RationalFunctionField method), 20
basis() (sage.rings.function_field.function_field_order.FunctionFieldOrder_basis method), 37
basis() (sage.rings.function_field.function_field_order.FunctionFieldOrder_rational method), 38

## C

change_variable_name() (sage.rings.function_field.function_field.FunctionField_polymod method), 8
change_variable_name() (sage.rings.function_field.function_field.RationalFunctionField method), 20
characteristic() (sage.rings.function_field.function_field.FunctionField method), 2
characteristic_polynomial() (sage.rings.function_field.function_field_element.FunctionFieldElement method), 27
charpoly() (sage.rings.function_field.function_field_element.FunctionFieldElement method), 27
constant_base_field() (sage.rings.function_field.function_field.FunctionField_polymod method), 9
constant_base_field() (sage.rings.function_field.function_field.RationalFunctionField method), 21
constant_field() (sage.rings.function_field.function_field.FunctionField_polymod method), 10
constant_field() (sage.rings.function_field.function_field.RationalFunctionField method), 21
create_key() (sage.rings.function_field.constructor.FunctionFieldFactory method), 51
create_key() (sage.rings.function_field.constructor.FunctionFieldPolymodFactory method), 52
create_object() (sage.rings.function_field.constructor.FunctionFieldFactory method), 51
create_object() (sage.rings.function_field.constructor.FunctionFieldPolymodFactory method), 52

## D

degree() (sage.rings.function_field.function_field.FunctionField_polymod method), 10
degree() (sage.rings.function_field.function_field.RationalFunctionField method), 21
denominator() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 31
derivation() (sage.rings.function_field.function_field.FunctionField_polymod method), 10
derivation() (sage.rings.function_field.function_field.RationalFunctionField method), 21

## E

element() (sage.rings.function_field.function_field_element.FunctionFieldElement_polymod method), 30
element() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 31
equation_order() (sage.rings.function_field.function_field.FunctionField_polymod method), 11
equation_order() (sage.rings.function_field.function_field.RationalFunctionField method), 22
extension() (sage.rings.function_field.function_field.FunctionField method), 2

## F

factor() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 31

field() (sage.rings.function_field.function_field.RationalFunctionField method), 22
fraction_field() (sage.rings.function_field.function_field_order.FunctionFieldOrder method), 35
fraction_field() (sage.rings.function_field.function_field_order.FunctionFieldOrder_basis method), 38
FractionFieldToFunctionField (class in sage.rings.function_field.maps), 45
free_module() (sage.rings.function_field.function_field_order.FunctionFieldOrder_basis method), 38
function_field() (sage.rings.function_field.function_field_order.FunctionFieldOrder method), 35
FunctionField (class in sage.rings.function_field.function_field), 2
FunctionField_polymod (class in sage.rings.function_field.function_field), 7
FunctionFieldConversionToConstantBaseField (class in sage.rings.function_field.maps), 46
FunctionFieldDerivation (class in sage.rings.function_field.maps), 46
FunctionFieldDerivation_rational (class in sage.rings.function_field.maps), 46
FunctionFieldDerivation_separable (class in sage.rings.function_field.maps), 47
FunctionFieldElement (class in sage.rings.function_field.function_field_element), 27
FunctionFieldElement_polymod (class in sage.rings.function_field.function_field_element), 30
FunctionFieldElement_rational (class in sage.rings.function_field.function_field_element), 31
FunctionFieldFactory (class in sage.rings.function_field.constructor), 51
FunctionFieldIdeal (class in sage.rings.function_field.function_field_ideal), 41
FunctionFieldIdeal_module (class in sage.rings.function_field.function_field_ideal), 42
FunctionFieldIsomorphism (class in sage.rings.function_field.maps), 47
FunctionFieldMorphism (class in sage.rings.function_field.maps), 47
FunctionFieldMorphism_polymod (class in sage.rings.function_field.maps), 47
FunctionFieldMorphism_rational (class in sage.rings.function_field.maps), 48
FunctionFieldOrder (class in sage.rings.function_field.function_field_order), 35
FunctionFieldOrder_basis (class in sage.rings.function_field.function_field_order), 37
FunctionFieldOrder_rational (class in sage.rings.function_field.function_field_order), 38
FunctionFieldPolymodFactory (class in sage.rings.function_field.constructor), 52
FunctionFieldToFractionField (class in sage.rings.function_field.maps), 48

## G

gen() (sage.rings.function_field.function_field.FunctionField_polymod method), 11
gen() (sage.rings.function_field.function_field.RationalFunctionField method), 22
gen() (sage.rings.function_field.function_field_order.FunctionFieldOrder_rational method), 39
genus() (sage.rings.function_field.function_field.FunctionField_polymod method), 12
genus() (sage.rings.function_field.function_field.RationalFunctionField method), 22

## H

hom() (sage.rings.function_field.function_field.FunctionField_polymod method), 12
hom() (sage.rings.function_field.function_field.RationalFunctionField method), 22

## I

ideal() (sage.rings.function_field.function_field_order.FunctionFieldOrder method), 36
ideal() (sage.rings.function_field.function_field_order.FunctionFieldOrder_rational method), 39
ideal_with_gens() (in module sage.rings.function_field.function_field_ideal), 43
ideal_with_gens_over_base() (in module sage.rings.function_field.function_field_ideal), 43
ideal_with_gens_over_base() (sage.rings.function_field.function_field_order.FunctionFieldOrder method), 36
intersection() (sage.rings.function_field.function_field_ideal.FunctionFieldIdeal_module method), 42
inverse_mod() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 32
is_field() (sage.rings.function_field.function_field_order.FunctionFieldOrder method), 37
is_finite() (sage.rings.function_field.function_field.FunctionField method), 3
is_finite() (sage.rings.function_field.function_field_order.FunctionFieldOrder method), 37

is_FunctionField() (in module sage.rings.function_field.function_field), 24
is_FunctionFieldElement() (in module sage.rings.function_field.function_field_element), 33
is_injective() (sage.rings.function_field.maps.FunctionFieldDerivation method), 46
is_injective() (sage.rings.function_field.maps.FunctionFieldIsomorphism method), 47
is_integral() (sage.rings.function_field.function_field_element.FunctionFieldElement method), 28
is_noetherian() (sage.rings.function_field.function_field_order.FunctionFieldOrder method), 37
is_perfect() (sage.rings.function_field.function_field.FunctionField method), 3
is_RationalFunctionField() (in module sage.rings.function_field.function_field), 25
is_separable() (sage.rings.function_field.function_field.FunctionField_polymod method), 13
is_square() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 32
is_surjective() (sage.rings.function_field.maps.FunctionFieldIsomorphism method), 47

## L

list() (sage.rings.function_field.function_field_element.FunctionFieldElement_polymod method), 30
list() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 32

## M

make_FunctionFieldElement() (in module sage.rings.function_field.function_field_element), 33
MapFunctionFieldToVectorSpace (class in sage.rings.function_field.maps), 48
MapVectorSpaceToFunctionField (class in sage.rings.function_field.maps), 49
matrix() (sage.rings.function_field.function_field_element.FunctionFieldElement method), 28
maximal_order() (sage.rings.function_field.function_field.FunctionField_polymod method), 14
maximal_order() (sage.rings.function_field.function_field.RationalFunctionField method), 23
minimal_polynomial() (sage.rings.function_field.function_field_element.FunctionFieldElement method), 29
minpoly() (sage.rings.function_field.function_field_element.FunctionFieldElement method), 29
module() (sage.rings.function_field.function_field_ideal.FunctionFieldIdeal_module method), 42
monic_integral_model() (sage.rings.function_field.function_field.FunctionField_polymod method), 14

## N

ngens() (sage.rings.function_field.function_field.FunctionField_polymod method), 15
ngens() (sage.rings.function_field.function_field.RationalFunctionField method), 23
ngens() (sage.rings.function_field.function_field_order.FunctionFieldOrder_rational method), 39
norm() (sage.rings.function_field.function_field_element.FunctionFieldElement method), 30
numerator() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 32

## O

order() (sage.rings.function_field.function_field.FunctionField method), 3
order_with_basis() (sage.rings.function_field.function_field.FunctionField method), 4

## P

polynomial() (sage.rings.function_field.function_field.FunctionField_polymod method), 15
polynomial() (sage.rings.function_field.function_field_order.FunctionFieldOrder_basis method), 38
polynomial_ring() (sage.rings.function_field.function_field.FunctionField_polymod method), 15
polynomial_ring() (sage.rings.function_field.function_field.RationalFunctionField method), 23
primitive_element() (sage.rings.function_field.function_field.FunctionField_polymod method), 16

## R

random_element() (sage.rings.function_field.function_field.FunctionField_polymod method), 16
random_element() (sage.rings.function_field.function_field.RationalFunctionField method), 24
rational_function_field() (sage.rings.function_field.function_field.FunctionField method), 4

RationalFunctionField (class in sage.rings.function_field.function_field), 20

## S

sage.rings.function_field.constructor (module), 51
sage.rings.function_field.function_field (module), 1
sage.rings.function_field.function_field_element (module), 27
sage.rings.function_field.function_field_ideal (module), 41
sage.rings.function_field.function_field_order (module), 35
sage.rings.function_field.maps (module), 45
section() (sage.rings.function_field.maps.FractionFieldToFunctionField method), 45
section() (sage.rings.function_field.maps.FunctionFieldToFractionField method), 48
simple_model() (sage.rings.function_field.function_field.FunctionField_polymod method), 16
some_elements() (sage.rings.function_field.function_field.FunctionField method), 5
sqrt() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 32

## T

trace() (sage.rings.function_field.function_field_element.FunctionFieldElement method), 30

## V

valuation() (sage.rings.function_field.function_field.FunctionField method), 5
valuation() (sage.rings.function_field.function_field_element.FunctionFieldElement_rational method), 33
vector_space() (sage.rings.function_field.function_field.FunctionField_polymod method), 18
vector_space() (sage.rings.function_field.function_field.RationalFunctionField method), 24