CONTENTS

1 Function Fields 3
2 Elements of function fields 51
3 Orders of function fields 67
4 Ideals of function fields 89
5 Places of function fields 113
6 Divisors of function fields 121
7 Differentials of function fields 127
8 Valuation rings of function fields 133
9 Morphisms of function fields 137
10 Factories to construct function fields 147
11 A Support Module 151
12 Indices and Tables 153
Python Module Index 155
Index 157
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 function fields over finite fields, number fields, and the algebraic closure of $\mathbb{Q}$. 
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: 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)
```

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

```
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
(1/(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: 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 + 3*x/(x^4 + 1))*t
```

(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.extension(v^5 - x); O
```

Function fields over the rational field are supported:

```
sage: F.<x> = FunctionField(QQ)
sage: R.<Y> = F[]
sage: L.<y> = F.extension(Y^2 - x^8 - 1)
sage: O = L.maximal_order()
sage: I = O.ideal(x, y - 1)
sage: P = I.place()
sage: D = P.divisor()
sage: D.basis_function_space()
[1]
sage: (2*D).basis_function_space()
[1]
sage: (3*D).basis_function_space()
[1]
sage: (4*D).basis_function_space()
[1, 1/x^4*y + 1/x^4]
```

Function fields over the algebraic field are supported:

```
sage: K.<x> = FunctionField(QQ); _.<Y> = K[]
sage: F.<y> = K.extension(Y^3 - x^2*(x^2 + x + 1)^2)
sage: O = F.maximal_order()
sage: I = O.ideal(y)
sage: I.divisor()
2*Place (x, y, (1/(x^3 + x^2 + x))*y^2) + 2*Place (x^2 + x + 1, y, (1/(x^3 + x^2 + x))*y^2)
```

```
sage: K.<x> = FunctionField(QQ); _.<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) + Place (x^2 + 1, x*y)
```
1.1 Global function fields

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) 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),
```

(continues on next page)
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 \((x^3 + x + 1, y + x^2)\),
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 \((x^3 + x^2 + 1, y + x^2 + x + 1)\)

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

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

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
- Brent Baccala (2019-12-20): added function fields over number fields and QQbar

\textbf{class} \texttt{sage.rings.function_field.function_field.FunctionField}(\texttt{base\_field, names, category=Category of function fields})

\texttt{Bases: sage.rings.ring.Field}

Abstract base class for all function fields.

\textbf{INPUT}:

- \texttt{base\_field} – field; the base of this function field
- \texttt{names} – string that gives the name of the generator

\textbf{EXAMPLES}:

\[
\text{sage: } K.<x> = \text{FunctionField}(\text{QQ})
\]
\[
\text{sage: } K
\]

Rational function field in x over Rational Field
characteristic()

Return the characteristic of the function field.

EXAMPLES:

```
sage: K.<x> = FunctionField(QQ)
sage: K.characteristic()
0
sage: K.<x> = FunctionField(QQbar)
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
```

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
sage: m(x,10)
s^2 + s^3 + 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)
```

1.1. Global function fields
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^6 + 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
  To:  Laurent Series Ring in s over Finite Field of size 2
sage: m(x)
s^-1 + O(s^19)
sage: m = K.completion(p, prec=infinity); m
Completion map:
  From: Rational function field in x over Finite Field of size 2
  To:  Lazy Laurent Series Ring in s over Finite Field of size 2
sage: f = m(x); f
s^-1 + ...
sage: f.coefficient(100)
0
sage: K.<x> = FunctionField(QQ); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 - x)
sage: O = L.maximal_order()
sage: decomp = O.decomposition(K.maximal_order().ideal(x - 1))
sage: pls = (decomp[0][0].place(), decomp[1][0].place())
sage: m = L.completion(pls[0]); m
Completion map:
  From: Function field in y defined by y^2 - x
  To:  Laurent Series Ring in s over Rational Field
sage: xe = m(x)
sage: ye = m(y)
sage: ye^2 - xe == 0
True
sage: decomp2 = O.decomposition(K.maximal_order().ideal(x^2 + 1))
sage: pls2 = decomp2[0][0].place()

(continues on next page)
sage: ye = m(y)
sage: ye^2 - xe == 0
True

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: _.<Y> = K[]

sage: L.<y> = K.extension(Y^3 - (t^3 - 1)/(t^3 - 2))
sage: L.divisor_group()
Divisor group of Function field in y defined by y^3 + (-t^3 + 1)/(t^3 - 2)

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

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

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

Return the order generated by *x* over the maximal infinite order.

**INPUT:**

- *x* – element or a list of elements of the function field
- **check** – boolean; 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)
L.order_infinite(y)  # todo: not implemented
```

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

Orders with multiple generators, not yet supported:

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

**order_infinite_with_basis**(*basis*, **check=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
- **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
K.<x> = FunctionField(QQ); R.<y> = K[]
L.<y> = K.extension(y^3 + x^3 + 4*x + 1)
O = L.order_infinite_with_basis([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
O.basis()
(1/x*y + 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
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
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:

```python
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:

```python
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:**

```python
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:

```python
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:

```python
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)

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

sage: K.<t> = FunctionField(QQ)
sage: K.place_set()
Set of places of Rational function field in t over Rational Field

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 + y + x + 1/x)
sage: L.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,
  (continues on next page)
\( \frac{x}{x^2 - 1} \), 
\( \frac{x}{x^2 + 1} \), 
\( \frac{1}{2} \cdot \frac{x}{x^2 + 1} \), 
0, 
\( \frac{1}{x} \),
...

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)

valuation\( (\text{prime}) \)
Return the discrete valuation on this function field defined by \( \text{prime} \).

INPUT:

- \( \text{prime} \) – 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
Value 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)
```
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)  
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
```

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

Bases: sage.rings.function_field.function_field.FunctionField_simple

Function fields of characteristic zero.

EXAMPLES:
```
sage: K.<x> = FunctionField(QQ); _.<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 + (-x^3 + 1)/(x^3 - 2)
sage: L.characteristic()
0
```

```
higher_derivative()

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(QQ); _.<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 + (-x^3 + 1)/(x^3 - 2)
  To:   Function field in y defined by y^3 + (-x^3 + 1)/(x^3 - 2)
```

class sage.rings.function_field.function_field.FunctionField_char_zero_integral(polynomial, names, category=None):

Bases: sage.rings.function_field.function_field.FunctionField_char_zero, sage.rings.function_field.function_field.FunctionField_integral

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

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

Bases: sage.rings.function_field.function_field.FunctionField_simple

Global function fields.

INPUT:

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

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
Function field in y defined by y^3 + (4*x^3 + 1)/(x^3 + 3)
```

The defining equation needs not be monic:

```python
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]
```

or may define a trivial extension:

```python
sage: K.<x> = FunctionField(GF(5)); _.<Y> = K
sage: L.<y> = K.extension(Y-1)
sage: L.genus()
0
```

**L_polynomial** *(name)*

Return the L-polynomial of the function field.

INPUT:

- name – (default: t) name of the variable of the polynomial

EXAMPLES:
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: F.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: F.L_polynomial()
2*t^2 + t + 1

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:
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]

get_place(degree)
Return a place of degree.

INPUT:
• degree – a positive integer

OUTPUT: a place of degree if any exists; otherwise None

EXAMPLES:
sage: F.<a> = GF(2)
sage: K.<x> = FunctionField(F)
sage: R.<Y> = PolynomialRing(K)
sage: L.<y> = K.extension(Y^4 + Y - x^5)
sage: L.get_place(1)
Place (x, y)
sage: L.get_place(2)
Place (x, y^2 + y + 1)
sage: L.get_place(3)
Place (x^3 + x^2 + 1, y + x^2 + x)
sage: L.get_place(4)
Place (x + 1, x^5 + 1)
sage: L.get_place(5)
Place (x^5 + x^3 + x^2 + x + 1, y + x^4 + 1)
sage: L.get_place(6)
Place (x^3 + x^2 + 1, y^2 + y + x^2)
sage: L.get_place(7)
Place (x^7 + x + 1, y + x^6 + x^5 + x^4 + x^3 + x)
sage: L.get_place(8)

higher_derivative()
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:
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)
```

number_of_rational_places(r=1)

Return the number of rational places of the function field whose constant field extended by degree \( r \).

INPUT:

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

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: F.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: F.number_of_rational_places()
4
sage: [F.number_of_rational_places(r) for r in [1..10]]
[4, 8, 4, 16, 44, 56, 116, 288, 508, 968]
```

places(degree=1)

Return a list of the places with degree.

INPUT:

- degree – positive integer (default: 1)

EXAMPLES:

```python
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)]

weierstrass_places()
Return all Weierstrass places of the function field.

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 + 1, y + x^2),
 Place (x^3 + x + 1, y + x^2 + 1),
 Place (x^3 + x^2 + 1, y + x^2 + x + 1)]

class sage.rings.function_field.function_field.FunctionField_global_integral(polynomial, names)
Bases: sage.rings.function_field.function_field.FunctionField_global, sage.rings.function_field.function_field.FunctionField_integral

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

class sage.rings.function_field.function_field.FunctionField_integral(polynomial, names, category=None)
Bases: sage.rings.function_field.function_field.FunctionField_simple

Integral function fields.
A function field is integral if it is defined by an irreducible separable polynomial, which is integral over the maximal order of the base rational function field.
equation_order()  
Return the equation order 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.equation_order()
Order in Function field in y defined by y^3 + x^6 + x^4 + x^2
```

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 `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_integral_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.

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: b = F.primitive_integral_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)  
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:

```
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 the 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
```

**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 $K$. 


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.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:**

```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)
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 z defined by z^2 - y
  To: Function field in zz defined by zz^2 - yy
  Defn: z |--> 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'))(continues on next page)```
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 base constant field 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)); 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 function field.

EXAMPLES:

sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[
    sage: L.<y> = K.extension(Y^5 - x)
    Traceback (most recent call last):
    ... NotImplementedError

degree(base=None)
Return the degree of the 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()
derivation()

Return a generator of the space of derivations over the constant base ring of this function field $K$.

A derivation on $K$ is a map $K \to K$ 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 separable_model(). This part of the algorithm uses the assumption that $k$ is perfect.

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
          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:

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

different()

Return the different of the function field.

EXAMPLES:

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

sage: F.<y> = K.extension(Y^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)
```

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:

```
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)
```

free_module(base=None, basis=None, map=True)

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 this subfield $R$, which defaults to the base field of this function field.
• basis – a basis for this field over the base.
• maps – boolean (default True), whether to return $R$-linear maps to and from $V$.

OUTPUT:
• a vector space over the base function field
• an isomorphism from the vector space to the field (if requested)
• an isomorphism from the field to the vector space (if requested)

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.free_module()
sage: V
Vector space of dimension 5 over Rational function field in x over Rational Field
sage: from_V
Isomorphism:
    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:
    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^2 V.0 + (1/x)^2 V.1
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)
M.free_module()
(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.free_module(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 z defined by z^2 - y, Isomorphism:
    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)
```

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:

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

```sage
K.<x> = FunctionField(QQ); R.<y> = K[]
L.<y> = K.extension(y^3 - (x^3 + 2*x*y + 1/x))
```
hom\((\text{im}_\text{gens}, \text{base}_\text{morphism}=\text{None})\)

Create a homomorphism from the function field to another function field.

**INPUT:**

- \(\text{im}_\text{gens}\) – list of images of the generators of the function field and of successive base rings.
- \(\text{base}_\text{morphism}\) – homomorphism of the base ring, after the \(\text{im}_\text{gens}\) are used. Thus if \(\text{im}_\text{gens}\) has length 2, then \(\text{base}_\text{morphism}\) should be a morphism from the base ring of the base ring of the function field.

**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: (phi(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: L.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 = 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+t)
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

```
\textbf{is\_separable(base=None)}
```

Return whether this is a separable extension of base.

\textbf{INPUT:}

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

\textbf{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()
Maximal order of Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x

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()
Maximal infinite order of Function field in y defined by y^5 - 2*x*y + (-x^4 - 1)/x

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

1.1. Global function fields
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: N.<z> = L.extension(z^2*y - 1/x)
sage: N.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
                    x |--> x
```

(continues on next page)
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:

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 the 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)

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 [Lan2002].) The algorithm reduces to this case with monic_integral_model().

EXAMPLES:

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: \( t \mapsto x \)
   \( w \mapsto y \),

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: \( y \mapsto w \)
   \( x \mapsto t \).

This also works for non-integral polynomials:

\begin{verbatim}
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/\( x\)^\( y\) + \( x\)^2
   Defn: \( y_\) \mapsto x
   \( x_\) \mapsto y,
Function Field morphism:
   From: Function field in \( y \) defined by \( 1/\( x\)^\( y\) + \( x\)^2
   To: Function field in \( y_\) defined by \( y_\)^3 + \( x_\)^2
   Defn: \( y \mapsto x_\)
   \( x \mapsto y_\)
\end{verbatim}

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

\begin{verbatim}
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 \mapsto y
   \( x \mapsto x
   Function Field endomorphism of Function field in \( y \) defined by \( y^3 + t
   Defn: \( y \mapsto y
   \( x \mapsto x
\end{verbatim}

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

\begin{verbatim}
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - t)
sage: L.separable_model()

Traceback (most recent call last):
\end{verbatim}
NotImplementedError: constructing a separable model is only implemented for function fields over a perfect constant base field

**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:

```python
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:

```python
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:
An example with higher degrees:

```python
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:

```python
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)
```

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

Bases: sage.rings.function_field.function_field.FunctionField_polymod

Function fields defined by irreducible and separable polynomials over rational function fields.
constant_field()

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

EXAMPLES:

```sage
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
```

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:

```sage
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
```

genus()

Return the genus of the function field.

EXAMPLES:

```sage
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.

places_above(p)

Return places lying above p.

INPUT:

- p – place of the base rational function field.

EXAMPLES:

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

```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, 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]
```
class sage.rings.function_field.function_field.RationalFunctionField(constant_field, names, category=None)

Bases: sage.rings.function_field.function_field.FunctionField

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

INPUT:

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

EXAMPLES:

```sage
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
```

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

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

```sage
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 of Rational function field in t over Finite Field of size 7
```

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

We define a morphism:

```sage
sage: K.<t> = FunctionField(QQ)
sage: L = FunctionField(QQ, 'tbar') # give variable name as second input
sage: K.hom(L.gen())
Function Field morphism:
```

(continues on next page)
From: Rational function field in t over Rational Field
To:  Rational function field in tbar over Rational Field
Defn: t |--> tbar

Here are some calculations over a number field:

sage: R.<x> = FunctionField(QQ)
sage: L.<y> = R[]
sage: F.<y> = R.extension(y^2 - (x^2+1))

sage: (y/x).divisor()
- Place (x, y - 1)
- Place (x, y + 1)
+ Place (x^2 + 1, y)

sage: A.<z> = QQ[]
sage: NF.<i> = NumberField(z^2+1)
sage: R.<x> = FunctionField(NF)
sage: L.<y> = R[]
sage: F.<y> = R.extension(y^2 - (x^2+1))

sage: (x/y*x.differential()).divisor()
-2*Place (1/x, 1/x*y - 1)
- Place (1/x, 1/x*y + 1)
+ Place (x, y - 1)
+ Place (x, y + 1)

sage: (x/y).divisor()
- Place (x - i, y)
+ Place (x, y - 1)
+ Place (x, y + 1)
- Place (x + i, y)

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:

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')

(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

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

EXAMPLES:

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

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

EXAMPLES:

sage: K.<t> = FunctionField(QQ)
sage: K.constant_base_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:

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

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:
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
- names – string or length-1 tuple

OUTPUT:

- a function field

EXAMPLES:
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:

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)

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

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()
From: Rational function field in x over Rational Field
To: Vector space of dimension 1 over Rational function field in x over Rational Field)

\textbf{gen}(n=0)

Return the n-th generator of the function field. If n is not 0, then an IndexError is raised.

\textbf{EXAMPLES:}

\begin{verbatim}
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.
\end{verbatim}

\textbf{genus}()

Return the genus of the function field, namely 0.

\textbf{EXAMPLES:}

\begin{verbatim}
sage: K.<x> = FunctionField(QQ)
sage: K.genus()
0
\end{verbatim}

\textbf{hom}(im_gens, base_morphism=None)

Create a homomorphism from 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 base_morphism; this is not checked.
- \textit{base_morphism} – a homomorphism from the base field into the other ring. If 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
\end{verbatim}
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 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

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.
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

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 RationalField

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)

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)
True

class sage.rings.function_field.function_field.RationalFunctionField_char_zero(constant_field, names, category=None)
Bases: sage.rings.function_field.function_field.RationalFunctionField
Rational function fields of characteristic zero.

higher_derivation()
Return the higher derivation for the function field.
This is also called the Hasse-Schmidt derivation.
EXAMPLES:

sage: F.<x> = FunctionField(QQ)
sage: d = F.higher_derivation()
sage: [d(x^5,i) for i in range(10)]
[x^5, 5*x^4, 10*x^3, 10*x^2, 5*x, 1, 0, 0, 0, 0]
sage: [d(x^9,i) for i in range(10)]
[x^9, 9*x^8, 36*x^7, 84*x^6, 126*x^5, 126*x^4, 84*x^3, 36*x^2, 9*x, 1]

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

1.1. Global function fields
Rational function field over finite fields.

**get_place**(degree)

Return a place of degree.

**INPUT:**

- degree – a positive integer

**EXAMPLES:**

```sage
sage: F.<a> = GF(2)
sage: K.<x> = FunctionField(F)
sage: K.get_place(1)
Place (x)
sage: K.get_place(2)
Place (x^2 + x + 1)
sage: K.get_place(3)
Place (x^3 + x + 1)
sage: K.get_place(4)
Place (x^4 + x + 1)
sage: K.get_place(5)
Place (x^5 + x^2 + 1)
```

**higher_derivation()**

Return the higher derivation for the function field.

This is also called the Hasse-Schmidt derivation.

**EXAMPLES:**

```sage
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
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:**

```sage
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:**

```sage
sage: F.<x> = FunctionField(GF(5))
sage: F.places()
[Place (1/x),
Place (1/x^2 + 1)]
```
places_finite(degree=1)
Return the finite places of the degree.

INPUT:
• degree – (default: 1) a positive integer

EXAMPLES:

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

sage.rings.function_field.function_field.is_FunctionField(x)
Return True if x is a function field.

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 provides arithmetic with elements of function fields.

EXAMPLES:

Arithmetic with rational functions:

```sage
sage: K.<t> = FunctionField(QQ)
sage: f = t - 1
sage: g = t^2 - 3
sage: 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
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
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, 2020-09-01): treat zero correctly; implement nth_root/is_nth_power
- Maarten Derickx (2011-09-11): added doctests, fixed pickling
- Kwankyu Lee (2017-04-30): added elements for global function fields
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

degree()
Return the max degree between the denominator and numerator.

EXAMPLES:

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

sage: FF.<t> = FunctionField(QQ)
sage: f = (t^8); f
(continues on next page)
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)
```

differential()
Return the differential $d x$ where $x$ is the element.

EXAMPLES:

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

divisor()
Return the divisor of the element.

EXAMPLES:

```
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)
```

\textbf{divisor\_of\_poles()}

Return the divisor of poles for the element.

EXAMPLES:

\begin{verbatim}
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)

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)
\end{verbatim}

\textbf{divisor\_of\_zeros()}

Return the divisor of zeros for the element.

EXAMPLES:

\begin{verbatim}
sage: K.<x> = FunctionField(GF(2))
sage: f = 1/(x^3 + x^2 + x)
sage: f.divisor_of_zeros()
3*Place (1/x)

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)
\end{verbatim}

\textbf{evaluate\(\text{(place)}\)}

Return the value of the element at the place.

INPUT:

- place – a function field place

OUTPUT:

If the element is in the valuation ring at the place, then an element in the residue field at the place is returned. Otherwise, \texttt{ValueError} is raised.

EXAMPLES:

\begin{verbatim}
sage: K.<t> = FunctionField(GF(5))
sage: p = K.place_infinite()
sage: f = 1/t^2 + 3
sage: f.evaluate(p)
3

sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: p, = L.places_infinite()
sage: (y + x).evaluate(p)
\end{verbatim}

(continues on next page)
Traceback (most recent call last):
...
ValueError: has a pole at the place
sage: (y/x + 1).evaluate(p)
1

higher_derivative(i, separating_element=None)

Return the \(i\)-th derivative of the element with respect to the separating element.

INPUT:

- \(i\) – nonnegative integer
- \(\text{separating}\_\text{element}\) – a separating element of the function field; the default is the generator of the rational function field

EXAMPLES:

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

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

is_integral()

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

EXAMPLES:

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

is_nth_power(n)

Return whether this element is an \(n\)-th power in the rational function field.

INPUT:

- \(n\) – an integer

OUTPUT:

Returns True if there is an element \(a\) in the function field such that this element equals \(a^n\).
See also:

nth_root()

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(3))
sage: f = (x+1)/(x-1)
sage: f.is_nth_power(2)
False
```

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]
```

(continues on next page)
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: 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: 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:
```
```
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
```

\textbf{n}th\_\textbf{root} \((n)\)

Return an \(n\)-th root of this element in the function field.

\textbf{INPUT}:

\begin{itemize}
  \item \(n\) – an integer
\end{itemize}

\textbf{OUTPUT}:

Returns an element \(a\) in the function field such that this element equals \(a^n\). Raises an error if no such element exists.

\textbf{See also}:

\textit{is_nth_power()}\n
\textbf{EXAMPLES}:

```
sage: K.<x> = FunctionField(GF(3))

sage: R.<y> = K[

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

sage: L(y^27).nth_root(27)
y
```

\textbf{poles}()

Return the list of the poles of the element.

\textbf{EXAMPLES}:

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

sage: f = 1/(x^3 + x^2 + x)

sage: f.poles()
[Place (x), Place (x^2 + x + 1)]
```

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

sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)

sage: (x/y).poles()
[Place (1/x, 1/x*y), Place (x + 1, x*y)]
```

\textbf{trace}()

Return the trace of the element.

\textbf{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

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

sage: K.<x> = FunctionField(QQ); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: O = L.maximal_order()
sage: p = O.ideal(x-1).place()
sage: y.valuation(p)
0

zeros()
Return the list of the zeros of the element.

EXAMPLES:

sage: K.<x> = FunctionField(GF(2))
sage: f = 1/(x^3 + x^2 + x)
sage: f.zeros()
[Place (1/x)]

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

class sage.rings.function_field.element.FunctionFieldElement_polymod

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 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
1/x^2*y + x/(x^2 + 1)
sage: f.element()
1/x^2*y + x/(x^2 + 1)
```

**is_nth_power**(n)

Return whether this element is an n-th power in the function field.

**INPUT:**

- n – an integer

**ALGORITHM:**

If n is a power of the characteristic of the field and the constant base field is perfect, then this uses the algorithm described in Proposition 12 of [GiTr1996].

**See also:**

**nth_root()**

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(4))
sage: R.<y> = K[

```
Returns an element $a$ in the function field such that this element equals $a^n$. Raises an error if no such element exists.

**ALGORITHM:**

If $n$ is a power of the characteristic of the field and the constant base field is perfect, then this uses the algorithm described in Proposition 12 of [GiTr1996].

**See also:**

`is_nth_power()`

**EXAMPLES:**

```python
sage: K.<x> = FunctionField(GF(3))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x)
sage: L(y^3).nth_root(3)
y
sage: L(y^9).nth_root(-9)
1/x*y
```

This also works for inseparable extensions:

```python
sage: K.<x> = FunctionField(GF(3))
sage: R.<y> = K[]
sage: L.<y> = K.extension(y^3 - x^2)
sage: L(x).nth_root(3)^3
x
sage: L(x^9).nth_root(-27)^-27
x^9
```

**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
```

**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
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 the element modulo the integral ideal $I$, if $I$ and the element together generate the unit ideal.

EXAMPLES:

```python
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_nth_power(n)`  
Return whether this element is an $n$-th power in the rational function field.

INPUT:
- $n$ – an integer

OUTPUT:
Returns `True` if there is an element $a$ in the function field such that this element equals $a^n$.

ALGORITHM:
If $n$ is a power of the characteristic of the field and the constant base field is perfect, then this uses the algorithm described in Lemma 3 of [GiTr1996].
See also:

nth_root()

EXAMPLES:

```sage
t.<x> = FunctionField(GF(3))
f = (x+1)/(x-1)
f.is_nth_power(1)  # True
f.is_nth_power(3)  # False
(f^3).is_nth_power(3)  # True
(f^9).is_nth_power(-9)  # True
```

is_square()

Return whether the element is a square.

EXAMPLES:

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

nth_root(n)

Return an n-th root of this element in the function field.

INPUT:

• n – an integer

OUTPUT:

Returns an element a in the rational function field such that this element equals \(a^n\). Raises an error if no such element exists.
ALGORITHM:

If \( n \) is a power of the characteristic of the field and the constant base field is perfect, then this uses the algorithm described in Corollary 3 of [GiTr1996].

See also:

\texttt{is\_nth\_power()}

EXAMPLES:

```
sage: K.<x> = FunctionField(GF(3))
sage: f = (x+1)/(x+2)
sage: f.nth_root(1)
(x + 1)/(x + 2)
sage: f.nth_root(3)
Traceback (most recent call last):
...
ValueError: element is not an n-th power
sage: (f^3).nth_root(3)
(x + 1)/(x + 2)
sage: (f^9).nth_root(-9)
(x + 2)/(x + 1)
```

\texttt{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
```

\texttt{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]
```

\texttt{valuation(place)}

Return the valuation of the rational function at the place.

Rational function field places are associated with irreducible polynomials.

INPUT:

- \texttt{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()
```

(continues on next page)
```python
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.rings.function_field.element.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
```
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
K.<x> = FunctionField(QQ)
sage: 0 = K.maximal_order()
sage: I = 0.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
K.<x> = FunctionField(GF(3)); _.<y> = K[]
sage: L.<y> = K.extension(y^3 - y - x)
sage: 0 = L.equation_order()
sage: 1/y in 0
False
sage: x/y in 0
True
```

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

```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: 0 = F.maximal_order()
(continues on next page)
```
When the base constant field is the algebraic field $\mathbb{Q}$, the only prime ideals of the maximal order of the rational function field are linear polynomials.

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
- Brent Baccala (2019-12-20): support orders in characteristic zero

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

Bases: sage.structure.unique_representation.UniqueRepresentation, sage.rings.function_field.order.FunctionFieldOrder

Base class of maximal orders of function fields.

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

Bases: sage.rings.function_field.order.FunctionFieldMaximalOrder, sage.rings.function_field.order.FunctionFieldOrderInfinite
Base class of maximal infinite orders of function fields.

```python
class sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_polymod:
    def __init__(self, field, category=None):
        pass
```

Maximal infinite orders of function fields.

INPUT:

- `field` – function field

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: 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:

```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: Oinf.basis()
(1, 1/x^2*y, 1/(x^4 + x^3 + x^2 + 1)*y^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: Oinf.basis()
(1, 1/x*y)
```

`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:

```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: f = 1/y^2
sage: f in Oinf
True
```

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

```
sage: K.<x> = FunctionField(QQ); _.<Y> = K[]
sage: F.<y> = K.extension(Y^3 - x^2*(x^2 + x + 1)^2)
sage: Oinf = F.maximal_order_infinite()
sage: Oinf.decomposition()
[(Ideal (1/x^2*y - 1) of Maximal infinite order of Function field in y defined by y^3 - x^6 - 2*x^5 - 3*x^4 - 2*x^3 - x^2, 1, 1),
 (Ideal ((1/(x^4 + x^3 + x^2))*y^2 + 1/x^2*y + 1) of Maximal infinite order of Function field in y defined by y^3 - x^6 - 2*x^5 - 3*x^4 - 2*x^3 - x^2, 2, 1)]
```

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 \(y^2 + y + (x^2 + 1)/x\)

\textbf{gen}(n=0)

Return the \(n\)-th generator of the order.

The basis elements of the order are generators.

**EXAMPLES:**

\begin{verbatim}
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_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
\end{verbatim}

\textbf{ideal}(\(*gens\))

Return the ideal generated by \(gens\).

**INPUT:**

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

**EXAMPLES:**

\begin{verbatim}
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 (y) of 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: 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 \(y^2 + y + (x^2 + 1)/x\)
\end{verbatim}

\textbf{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:**

\begin{verbatim}
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 \(y^2 + y + (x^2 + 1)/x\)
\end{verbatim}
 sage: K.<x> = FunctionField(GF(2)); R.<t> = K[]
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 (y) of Maximal infinite order of Function field in y
defined by y^3 + x^6 + x^4 + x^2

gens()
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_infinite()
sage: Oinf.ngens()
3

class sage.rings.function_field.order.FunctionFieldMaximalOrderInfinite_rational(field, category=None)

Maximal infinite orders of rational function fields.

INPUT:

  * field – a rational function field

EXAMPLES:

 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:

 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:

 sage: 0 = FunctionField(QQ,'y').maximal_order()
sage: 0.gen()
y
 sage: 0.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:

```
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:

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

prime_ideal()
Return the unique prime ideal of the order.

EXAMPLES:

```
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
```

class sage.rings.function_field.order.FunctionFieldMaximalOrder_global(field)
Bases: sage.rings.function_field.order.FunctionFieldMaximalOrder_polymod

Maximal orders of global function fields.

INPUT:
  • field – function field to which this maximal order belongs

EXAMPLES:
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: 0 = F.maximal_order()
sage: p = o.ideal(x+1)
sage: 0.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, (1/(x^3 + x^2 + x))*y^2 + y + 1) of Maximal order of Function field in y defined by y^3 + x^6 + x^4 + x^2, 2, 1)]
```

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: 0 = F.maximal_order()
sage: p = o.ideal(x+1)
sage: 0.p_radical(p)
Ideal (x + 1) of Maximal order of Function field in y defined by y^3 + x^6 + x^4 + x^2
```

class sage.rings.function_field.order.FunctionFieldMaximalOrder_polymod(field, ideal_class=<class 'sage.rings.function_field.ideal.FunctionFieldIdeal_polymod'>)

Bases: sage.rings.function_field.order.FunctionFieldMaximalOrder

Maximal orders of extensions of function fields.

basis()

Return a basis of the order over the maximal order of the base 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: 0 = L.equation_order()
sage: 0.basis()
(1, y, y^2, y^3)

sage: K.<x> = FunctionField(QQ)
sage: R.<t> = PolynomialRing(K)
sage: F.<y> = K.extension(t^4 + x^12*t^2 + x^18*t + x^21 + x^18)
sage: 0 = F.maximal_order()
sage: 0.basis()
(1, 1/x^4*y, 1/x^9*y^2, 1/x^13*y^3)

codifferent()

Return the codifferent 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.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

cordinate_vector(e)

Return the coordinates of e with respect to the basis of this 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: f = (x + y)^3
sage: O.coordinate_vector(f)
(x^3, 3*x^2, 3*x, 1)

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

ALGORITHM:

In principle, we're trying to compute a primary decomposition of the extension of ideal in self (an order, and therefore a ring). However, while we have primary decomposition methods for polynomial rings, we lack any such method for an order. Therefore, we construct self mod ideal as a finite-dimensional algebra, a construct for which we do support primary decomposition.

See https://trac.sagemath.org/attachment/ticket/28094/decomposition.pdf

Todo: Use Kummer's theorem to shortcut this code if possible, like as done in FunctionFieldMaximalOrder_global.decomposition()

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 (y^3 + 2*x)
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:

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

\textbf{ideal(\texttt{\*gens, **kwargs})}

Return the fractional ideal generated by the elements in \texttt{gens}.

INPUT:

- \texttt{gens} – list of generators

EXAMPLES:

```python
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/(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(QQ); 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/(x^3 + 1))*y) of Maximal order of Function field in y defined by y^2 + x^3 - 1
sage: I2 = S.ideal(x^2-4); I2
Ideal (x^2 - 4) of Maximal order of Function field in y defined by y^2 - x^3 - 1
sage: I2 == S.ideal(I)
True
```

\textbf{ideal\_with\_gens\_over\_base(\texttt{gens})}

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

- \texttt{gens} – list of elements that generates the ideal 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^2 - x^3 - 1)
sage: O = L.maximal_order(); O
Maximal order of 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) of Maximal order of 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:

```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x^3 - 1)
sage: Oinf = L.maximal_order()
sage: I = Oinf.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
```

\texttt{ngens()}

Return the number of generators of the order.

EXAMPLES:

```python
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
```

\texttt{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)
sage: 0 = L.equation_order()
sage: 0.polynomial()
y^4 + x*y + 4*x + 1
```

(continues on next page)
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:

    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

    sage: O.basis()
    (1,)

    sage: O.gen()
    y
    sage: O.gen(1)
    Traceback (most recent call last):
    ... IndexError: there is only one generator

    sage: O.ideal(x)
    Ideal (x) of Maximal order of Rational function field in t over Finite Field of size 19
    sage: O.ideal([x,1/x]) == O.ideal(x,1/x)
    # multiple generators may be given as a list
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
```
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):

```python
sage: O = 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]
```

(continues on next page)
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:

```sage
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:

```sage
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 + 6*x^3 + 6
sage: I = O.ideal_with_gens_over_base([1, 1/x^2*y]); I
Ideal (1) of Order in Function field in y defined by y^2 + 6*x^3 + 6
```

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
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
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) of Order in Function field in y defined by y^2 + 6*x^3 + 6
```

(continues on next page)
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:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]

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

```python
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)
sage: O = L.equation_order()
sage: O.polynomial()
y^4 + x*y + 4*x + 1
```

### class `sage.rings.function_field.order.FunctionFieldOrder_base`

Bases: `sage.structure.unique_representation.CachedRepresentation`, `sage.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().function_field()
Rational function field in y over Rational Field
```

**function_field()**

Return the function field to which the order belongs.
EXAMPLES:

```
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:

```
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**(proof=True)
Return False since orders are never fields.

EXAMPLES:

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

**is_noetherian()**
Return True since orders in function fields are noetherian.

EXAMPLES:

```
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:

```
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)*

Bases: `sage.rings.function_field.order.FunctionFieldOrder`

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:

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

```
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):

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

### basis()

Return a basis of the order 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)
```

### coordinate_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:**

```
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:

```sage
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:

```sage
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:

```sage
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: S = L.equation_order()
sage: I = O.ideal(x^2-4)
sage: L.<y> = K.extension(y^4 + x*y + 4*x + 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^4 + 6*x^3 + 6
sage: I2 = S.ideal(x^2-4); I2
Ideal (x^2 + 3) of Order in Function field in y defined by y^4 + 6*x^3 + 6
sage: I2 == S.ideal(I)
True
```

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 the gens really generates an ideal.

INPUT:

- `gens` – list of elements of the function 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) 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...
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^4 + x*y + 4*x + 1)
sage: O = L.equation_order()
sage: O.polynomial()
y^4 + x*y + 4*x + 1
```

```
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
```
CHAPTER
FOUR

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:

```
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:

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

```

Ideals in the maximal order of a global function field:

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

(continues on next page)
Ideal \((1/x^4*y)\) of Maximal order of Function field in \(y\) defined by \(y^3 + y^2 + 2*x^4\)
```
sage: I^2
Ideal (1/x^4*y) of Maximal infinite order of Function field in \(y\) defined by \(y^3 + y^2 + 2*x^4\)
```

Ideals in the maximal infinite order of a global function field:
```
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^4*y) of Maximal infinite order of Function field in \(y\) defined by \(y^3 + y^2 + 2*x^4\)
```

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

```python
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:
```
```
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:**

```python
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)
```

```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.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)
```

```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: 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:**

```python
divisor_of_zeros()

Return the divisor of zeros corresponding to the ideal.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(4))
sage: O = K.maximal_order()
sage: I = 0.ideal(x^2(x + 1)^2/(x^4 + x + 1))
sage: I.divisor_of_zeros()
Place (x) + 2*Place (x + 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_of_zeros()
2*Place (x + 1, x*y)
```

factor()

Return the factorization of this ideal.

Subclass of this class should define `_factor()` method that returns a list of prime ideal and multiplicity pairs.

EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(4))
sage: O = K.maximal_order()
sage: I = 0.ideal(x^3*(x + 1)^2)
```

(continues on next page)
sage: I.factor()
(Ideal (x) of Maximal order of Rational function field in x
over Finite Field in \(z_2\) of size \(2^2\))^3 *
(Ideal (x + 1) of Maximal order of Rational function field in x
over Finite Field in \(z_2\) of size \(2^2\))^2

sage: Oinf = K.maximal_order_infinite()
sage: I = Oinf.ideal((x + 1)/(x^3 + 1))
sage: I.factor()
(Ideal (1/x) of Maximal infinite order of Rational function field in x
over Finite Field in \(z_2\) of size \(2^2\))^2

sage: K.<x> = FunctionField(GF(2)); _.<T> = PolynomialRing(K)
sage: F.<y> = K.extension(T^3 - x^2*(x^2 + x + 1)^2)
sage: 0 = F.maximal_order()
sage: I = 0.ideal(y)
sage: I == I.factor().prod()
True

sage: Oinf = F.maximal_order_infinite()
sage: f= 1/x
sage: I = Oinf.ideal(f)
sage: I.factor()
(Ideal ((1/(x^4 + x^3 + x^2))*y^2 + 1/x^2*y + 1) of Maximal infinite order
of Function field in y defined by y^3 + x^6 + x^4 + x^2) *
(Ideal ((1/(x^4 + x^3 + x^2))*y^2 + 1) of Maximal infinite order
of Function field in y defined by y^3 + x^6 + x^4 + x^2)

sage: K.<x> = FunctionField(QQ); _.<Y> = K[]
sage: F.<y> = K.extension(Y^3 - x^2*(x^2 + x + 1)^2)
sage: 0 = F.maximal_order()
sage: I = 0.ideal(y)
sage: I == I.factor().prod()
True

sage: K.<x> = FunctionField(QQ); _.<Y> = K[]
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
sage: 0 = L.maximal_order()
sage: I = 0.ideal(y)
sage: I == I.factor().prod()
True

gens_reduced()

Return reduced generators.

For now, this method just looks at the generators and sees if any can be removed without changing the ideal. It prefers principal representations (a single generator) over all others, and otherwise picks the generator set with the shortest print representation.

This method is provided so that ideals in function fields have the method `gens_reduced()`, just like ideals of number fields. Sage linear algebra machinery sometimes requires this.

EXAMPLES:
sage: K.<x> = FunctionField(GF(7))
sage: O = K.equation_order()
sage: I = O.ideal(x,x^2,x^2+x)
sage: I.gens_reduced()
(x,)

place()
Return the place associated with this prime ideal.

EXAMPLES:

sage: K.<x> = FunctionField(GF(4))
sage: O = K.maximal_order()
sage: I = O.ideal(x^2 + x + 1)
sage: I.place()
Traceback (most recent call last):
...  
TypeError: not a prime ideal
sage: I = O.ideal(x^3+x+1)
sage: I.place()
Place (x^3 + x + 1)

sage: K.<x> = FunctionField(GF(2))
sage: Oinf = K.maximal_order_infinite()
sage: I = Oinf.ideal((x + 1)/(x^3 + 1))
sage: p = I.factor()[0][0]
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)
[Place (x, (1/(x^3 + x^2 + x))*y^2),
 Place (x^2 + x + 1, (1/(x^3 + x^2 + x))*y^2)]

sage: J = I.factor()[0][0]
sage: J.is_prime()

(continues on next page)
True

```python
sage: J.place()
Place (1/x, 1/x^3*y^2)
```

```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(1/x)
sage: I.factor()
(Ideal (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)
```

### ring()

Return the ring to which this ideal belongs.

**EXAMPLES:**

```python
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
```

### class sage.rings.function_field.ideal.FunctionFieldIdealInfinite(ring)

Bases: `sage.rings.function_field.ideal.FunctionFieldIdeal`

Base class of ideals of maximal infinite orders

### 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:**

```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: 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:

```python
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
```

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

Ideals of the infinite maximal order of an algebraic function field.

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^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)
```
```
sage: I = Oinf.ideal(x+y)
sage: I.gens()
(x, y)

**gens_over_base()**

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(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:**

```
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_two()
  (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(x+y)
  sage: I.gens_two()
  (x,)
```

**ideal_below()**

Return a set of generators of this 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/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^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

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*y) of Maximal infinite order of Function field in y defined by y^2 + y + (x^2 + 1)/x)^2
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:

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^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*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:

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_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: 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:
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:

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:

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:

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_polymod

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 (y) of Maximal order of Function field in y defined by y^2 + x^3*y + x
```

`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)
```

`gens_two()`

Return two generators of this fractional ideal.

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

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)
sage: I
Ideal (y) of Maximal order of Function field in y defined by y^3 + x^6 + x^4 + x^2
sage: ~I  # indirect doctest
Ideal ((1/(x^6 + x^4 + x^2)))*y^2) of Maximal 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: O = L.maximal_order()
sage: I = O.ideal(y)
sage: I
Ideal (y) of Maximal order of Function field in y defined by y^2 + y + x + 1/x
sage: ~I  # indirect doctest
Ideal ((x/(x^2 + 1))*y + x/(x^2 + 1)) of Maximal order of Function field in y defined by y^2 + y + (x^2 + 1)/x
```
class sage.rings.function_field.ideal.FunctionFieldIdeal_module(ring, 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.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
```
```
}\)
```
```

Return the \(i\)-th generator in the current basis 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.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: 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
    Vector space of dimension 2 over Rational function field in x over Rational˓→Field
    sage: I.module().is_submodule(V)
    True

ngens()

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_polymod(ring, hnf, denominator=1)

Fractional ideals of algebraic 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:**

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

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

sage: 0 = L.maximal_order()

sage: 0.ideal(y)
Ideal (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:**

```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: 0 = F.maximal_order()

sage: I = 0.ideal(x,1/y)

sage: I.denominator() * I.basis_matrix() == I.hnf()
True
```

**denominator()**

Return the denominator of this fractional ideal.

**EXAMPLES:**

```python
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
```

```python
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/(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:**
```python
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)

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)
```

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

EXAMPLES:

```python
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)
```

hnf()  
Return the matrix in hermite normal form representing this ideal.

See also denominator()

EXAMPLES:

```python
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)); I.hnf()
[x^6 + x^3 0]
[ x^3 + 1 1]
```

```python
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*(y+1)); I.hnf()
[x^6 + x^3 0]
[ x^3 + 1 1]
```

ideal_below()  
Return the ideal below this ideal.

This is defined only for integral ideals.
EXAMPLES:

```sage
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
```

```sage
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(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) of Maximal order of Rational function field
in x over Finite Field of size 2
```

```sage
sage: K.<x> = FunctionField(QQ); _.<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 Rational Field
```

`intersect(other)`

Intersect this ideal with the other ideal as fractional ideals.

**INPUT:**

* other – ideal

**EXAMPLES:**

```sage
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
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
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(x,1/y)
sage: I.is_integral()
False
sage: J = I.denominator() * I
sage: J.is_integral()
True
```

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

**is_prime()**

Return True if this ideal is a prime ideal.

**EXAMPLES:**

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

(continues on next page)
sage: I = O.ideal(y)
sage: [f.is_prime() for f, _ in I.factor()]
[True, True]
sage: K.<x> = FunctionField(QQ); _.<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]

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)]

norm()

Return the norm of this fractional 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: 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

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)
(continues on previous page)
\begin{verbatim}
  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
\end{verbatim}

**prime_below()**

Return the prime lying below this prime ideal.

**EXAMPLES:**

\begin{verbatim}
  sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[
  sage: F.<y> = K.extension(Y^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]
  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]
  sage: F.<y> = K.extension(Y^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 Rational Field,
   Ideal (x^2 + x + 1) of Maximal order of Rational function field in x over Rational Field]
\end{verbatim}

**valuation(ideal)**

Return the valuation of \texttt{ideal} at this prime ideal.

**INPUT:**

- \texttt{ideal} – fractional ideal

**EXAMPLES:**
The method closely follows Algorithm 4.8.17 of [Coh1993].

```python
sage: K.<x> = FunctionField(GF(2)); _.<t> = K[
```
```
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
```
```
sage: 0 = F.maximal_order()
```
```
sage: I = 0.ideal(x, (1/(x^3 + x^2 + x))*y^2)
```
```
sage: I.is_prime()
```
True
```
```
sage: J = 0.ideal(y)
```
```
sage: I.valuation(J)
```
2
```
```
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[
```
```
sage: L.<y> = K.extension(Y^2 + Y + x + 1/x)
```
```
sage: 0 = L.maximal_order()
```
```
sage: I = 0.ideal(y)
```
```
sage: [f.valuation(I) for f, _ in I.factor()]
```
[-1, 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:

```python
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:

```python
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:

```python
sage: K.<x> = FunctionField(GF(4))
```
```
sage: 0 = K.maximal_order()
```
```
sage: I = 0.ideal(x^2+x)
```
(continues on next page)
sage: I.gen()
x^2 + x

gens()
Return the tuple of 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.gens()
(x^2 + x,)

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

EXAMPLES:

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:

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:

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
IdealMonoid of ideals of Maximal order of Rational function field in x over Finite Field of size 2
```

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
```
CHAPTER FIVE

PLACES OF FUNCTION FIELDS

The places of a function field correspond, one-to-one, to valuation rings of the function field, each of which defines a discrete valuation for 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 a function field can be computed:

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

The homomorphisms 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)
sage: 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
- Brent Baccala (2019-12-20): function fields of characteristic zero

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

INPUT:

• parent – place set of a function field
• prime – prime ideal associated with the place

EXAMPLES:

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

divisor(multiplicity=1)
    Return the prime divisor corresponding to the place.

EXAMPLES:

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

function_field()
    Return the function field to which the place belongs.

EXAMPLES:

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

prime_ideal()
    Return the prime ideal associated with the place.

EXAMPLES:

```sage
t sage: K.<x>=FunctionField(GF(2)); _.<Y>=K[]
t sage: L.<y>=K.extension(Y^3+x+x^3*y)
t sage: p = L.places()[0]
t sage: p.prime_ideal()
Ideal (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_polymod(parent, prime)
Bases: sage.rings.function_field.place.FunctionFieldPlace

Places of extensions of function fields.

degree()
    Return the degree of the place.

EXAMPLES:
```python
gaps()
gaps() Return the gap sequence for the place.

EXAMPLES:
```}

```python
is_infinite_place()
is_infinite_place() Return True if the place is above the unique infinite place of the underlying rational function field.

EXAMPLES:
```}

```python
local_uniformizer()
local_uniformizer() Return an element of the function field that has a simple zero at the place.

EXAMPLES:
```}

```python
place_below()
place_below() Return the place lying below the place.

EXAMPLES:
```}

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

**relative_degree()**

Return the relative degree of the place.

**EXAMPLES:**

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

**residue_field**(name=None)

Return the residue field of the place.

**INPUT:**

- name - string; name of the generator of the residue field

**OUTPUT:**

- a field isomorphic to the residue field
- a ring homomorphism from the valuation ring to the field
- a ring homomorphism from the 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: k, fr_k, to_k = p.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(y)
```

(continues on next page)
Traceback (most recent call last):
...  
TypeError: y fails to convert into the map's domain  
Valuation ring at Place (x, x*y)...

to_k(1/y)  
0

to_k(y/(1+y))  
1

valuation_ring()  
Return the valuation ring 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: p.valuation_ring()  
Valuation ring at Place (x, x*y)

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

Places of rational function fields.

degree()  
Return the degree of the place.

EXAMPLES:

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:

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:

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:

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

valuation_ring()
Return the valuation ring at the place.

EXAMPLES:

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

sage: L.<y> = K.extension(Y^3 + x + x^3*Y)
sage: p = L.places_finite()[0]
sage: L.place_set()
Set of places of Function field in y defined by y^3 + x^3*y + x

class sage.rings.function_field.place.PlaceSet(field)
Bases: sage.structure.unique_representation.UniqueRepresentation, sage.structure.parent.Parent
Sets of Places of function fields.

INPUT:

• field – function field

EXAMPLES:

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

Element
alias of FunctionFieldPlace

function_field()
Return the function field to which this place set belongs.
EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(2)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x + x^3*Y)
sage: PS = L.place_set()
sage: PS.function_field() == L
True
```
Sage allows extensive computations with divisors on 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: 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: 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: 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

**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:**
sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[

sage: F.<y> = K.extension(Y^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:

sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[

sage: F.<y> = K.extension(Y^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)
Bases: sage.structure.element.ModuleElement

Divisors of function fields.

INPUT:

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

EXAMPLES:

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

sage: F.<y> = K.extension(Y^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:

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

sage: l = len(d.basis_function_space())
sage: i = len(d.basis_differential_space())
sage: l == d.degree() + 1 - L.genus() + i
True

basis_function_space()
Return a basis of the Riemann-Roch space of the divisor.
EXAMPLES:

```python
sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[]
sage: F.<y> = K.extension(Y^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:

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

**dict()**

Return the dictionary representing the divisor.

EXAMPLES:

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

```python
sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[]
sage: F.<y> = K.extension(Y^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
```

(continues on next page)
### dimension()

Return the dimension of the Riemann-Roch space of the divisor.

**EXAMPLES:**

```
sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[]
sage: F.<y> = K.extension(Y^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:**

```
sage: K.<x> = FunctionField(GF(5)); _.<Y> = K[]
sage: F.<y> = K.extension(Y^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:**

```
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
sage: f = x/(y + 1)
sage: D = f.divisor()
sage: D.list()
[(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 – place of a function field

EXAMPlES:

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

**support()**

Return the support of the divisor.

EXAMPlES:

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

**valuation(place)**

Return the multiplicity of the divisor at the place.

INPut:

• place – place of a function field

EXAMPlES:

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

**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[]
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: from sage.rings.function_field.divisor import divisor
sage: p, q, r = F.places()
sage: divisor(F, {p: 1, q: 2, r: 3})
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[]
sage: F.<y> = K.extension(t^3 - x^2*(x^2 + x + 1)^2)
sage: p = F.places()[0]
sage: from sage.rings.function_field.divisor import prime_divisor
sage: d = prime_divisor(F, p)
sage: 3 * d == prime_divisor(F, p, 3)
True
Sage provides arithmetic with differentials of function fields.

EXAMPLES:

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

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

We can compute a canonical divisor:

\[
\text{sage: } k = \text{df.divisor()} \\
\text{sage: } k.degree() \\
4 \\
\text{sage: } k\text{.degree()} == 2 \ast \text{L.genus()} - 2 \\
\text{True} \\
\]

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

\[
\text{sage: } \text{df.cartier()} \\
\text{0} \\
\text{sage: } w = 1/f \ast \text{df} \\
\text{sage: } w\text{.cartier()} == w \\
\text{True} \\
\]

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:

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

The space of differentials is a one-dimensional module over the function field. So a base differential is chosen to represent differentials. Usually the generator of the base rational function field is a separating element and used to generate the base differential. Otherwise a separating element is automatically found and used to generate the base differential relative to which other differentials are denoted:

```
sage: K.<x> = FunctionField(GF(5))
sage: R.<y> = K[
```
```
sage: L.<y> = K.extension(y^5 - 1/x)
sage: L(x).differential()
0
sage: y.differential()
d(y)
sage: (y^2).differential()
(2*y) d(y)
```

```
Element
alias of FunctionFieldDifferential
```

```
basis()
Return a basis.

EXAMPLES:
```
```
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[
```
```
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
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:
```
```
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[
```
```
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
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.DifferentialsSpaceInclusion
Bases: sage.categories.morphism.Morphism
```

Inclusion morphisms for extensions of function fields.

EXAMPLES:
```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: OK = K.space_of_differentials()
sage: OL = L.space_of_differentials()
sage: OL.coerce_map_from(OK)
Inclusion morphism:
  From: Space of differentials of Rational function field in x over Rational Field
  To:   Space of differentials of Function field in y defined by y^2 - x*y + 4*x^3

is_injective()
Return True, since the inclusion morphism is injective.

EXAMPLES:
```n
```
```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: OK = K.space_of_differentials()
sage: OL = L.space_of_differentials()
sage: OL.coerce_map_from(OK).is_injective()
True

is_surjective()
Return True if the inclusion morphism is surjective.

EXAMPLES:
```n
```
```python
sage: K.<x> = FunctionField(QQ); R.<y> = K[]
sage: L.<y> = K.extension(y^2 - x*y + 4*x^3)
sage: OK = K.space_of_differentials()
sage: OL = L.space_of_differentials()
sage: OL.coerce_map_from(OK).is_surjective()
False
sage: S.<z> = L[]
sage: M.<z> = L.extension(z - 1)
sage: OM = M.space_of_differentials()
sage: OM.coerce_map_from(OL).is_surjective()
True
```

```python
class sage.rings.function_field.differential.DifferentialsSpace_global(field, category=None)

Bases: sage.rings.function_field.differential.DifferentialsSpace

Space of differentials of a global function field.

INPUT:

- `field` – function field

EXAMPLES:
```n
```
```python
sage: K.<x> = FunctionField(GF(4)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
sage: L.space_of_differentials()
Space of differentials of Function field in y defined by y^3 + x^3*y + x
```

```python
Element

alias of FunctionFieldDifferential_global
```
```
class sage.rings.function_field.differential.FunctionFieldDifferential(parent, f, t=None)

Bases: sage.structure.element.ModuleElement

Base class for differentials on function fields.

INPUT:

• \( f \) – element of the function field

• \( t \) – element of the function field; if \( t \) is not specified, the generator of the base differential is assumed

EXAMPLES:

\[
\begin{align*}
\text{sage: } & F.<x>=\text{FunctionField}(\mathbb{Q}) \\
\text{sage: } & f = x/(x^2 + x + 1) \\
\text{sage: } & f.differential() \\
& ((-x^2 + 1)/(x^4 + 2*x^3 + 3*x^2 + 2*x + 1)) \, d(x)
\end{align*}
\]

\[
\begin{align*}
\text{sage: } & K.<x> = \text{FunctionField}(\mathbb{Q}); _.<Y> = K[] \\
\text{sage: } & L.<y> = K.extension(Y^3 + x + x^3*Y) \\
\text{sage: } & L(x).differential() \\
& d(x) \\
\text{sage: } & y.differential() \\
& ((21/4*x/(x^7 + 27/4))*y^2 + ((3/2*x^7 + 9/4)/(x^8 + 27/4*x))*y + 7/2*x^4/(x^7 + 27/4)) \, d(x)
\end{align*}
\]

divisor()

Return the divisor of the differential.

EXAMPLES:

\[
\begin{align*}
\text{sage: } & K.<x> = \text{FunctionField}(\mathbb{F}(5)); _.<Y> = K[] \\
\text{sage: } & L.<y> = K.extension(Y^3+x+x^3*Y) \\
\text{sage: } & w = (1/y) * y.differential() \\
\text{sage: } & w.discriminant() \\
& - \text{Place } (1/x, 1/x^3*y^2 + 1/x) \\
& - \text{Place } (1/x, 1/x^3*y^2 + 1/x^2*y + 1) \\
& - \text{Place } (x, y) \\
& + \text{Place } (x + 2, y + 3) \\
& + \text{Place } (x^6 + 3*x^5 + 4*x^4 + 2*x^3 + x^2 + 3*x + 4, y + x^5)
\end{align*}
\]

\[
\begin{align*}
\text{sage: } & F.<x> = \text{FunctionField}(\mathbb{Q}) \\
\text{sage: } & w = (1/x).differential() \\
\text{sage: } & w.discriminant() \\
& -2*\text{Place } (x)
\end{align*}
\]

monomial_coefficients(copy=True)

Return a dictionary whose keys are indices of basis elements in the support of \( \text{self} \) and whose values are the corresponding coefficients.

EXAMPLES:

\[
\begin{align*}
\text{sage: } & K.<x> = \text{FunctionField}(\mathbb{F}(5)); _.<Y> = K[] \\
\text{sage: } & L.<y> = K.extension(Y^3+x+x^3*Y) \\
\text{sage: } & d = y.differential() \\
\text{sage: } & d \\
& ((4*x/(x^7 + 3))*y^2 + (((4*x^7 + 1)/(x^8 + 3*x))*y + x^4/(x^7 + 3)) \, d(x)
\end{align*}
\]

(continues on next page)
sage: d.monomial_coefficients()
{0: (4*x/(x^7 + 3))*y^2 + ((4*x^7 + 1)/(x^8 + 3*x))*y + x^4/(x^7 + 3)}

**residue(place)**
Return the residue of the differential at the place.

**INPUT:**
- place – a 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:

```python
sage: F.<x> = FunctionField(GF(4))
sage: f = 0
sage: while f == 0:
    ....:    f = F.random_element()
sage: w = 1/f * f.differential()
sage: d = f.divisor()
sage: s = d.support()
sage: sum([w.residue(p).trace() for p in s])
0
```

and in an extension field:

```python
sage: K.<x> = FunctionField(GF(7)); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x + x^3*Y)
sage: f = 0
sage: while f == 0:
    ....:    f = L.random_element()
sage: w = 1/f * f.differential()
sage: d = f.divisor()
sage: s = d.support()
sage: sum([w.residue(p).trace() for p in s])
0
```

and also in a function field of characteristic zero:

```python
sage: R.<x> = FunctionField(QQ)
sage: L.<y> = R[]
sage: F.<y> = R.extension(Y^2 - x^4 - 4*x^3 - 2*x^2 - 1)
sage: a = 6*x^2 + 5*x + 7
sage: b = 2*x^6 + 8*x^5 + 3*x^4 - 4*x^3 -1
sage: w = y*a/b*x.differential()
sage: d = w.divisor()
sage: sum([QQ(w.residue(p)) for p in d.support()])
0
```

**valuation(place)**
Return the valuation of the differential at the place.

**INPUT:**
• place – a place of the function field

**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 valuation(p) for p in L.places()]
[-1, -1, -1, 0, 1, 0]
```

```python
class sage.rings.function_field.differential.FunctionFieldDifferential_global(parent, f, t=None)

Bases: sage.rings.function_field.differential.FunctionFieldDifferential

Differentials on global function fields.

**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)
```

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

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 in prime-power representation \( \omega = (f_p^0 + f_p^1 x + \cdots + f_p^{p-1} x^{p-1})dx \), then the Cartier operator maps \( \omega \) to \( f_p^{-1} dx \). 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
```

```python
sage: F.<x> = FunctionField(GF(4))
sage: f = x/(x^2 + x + 1)
sage: w = 1/f*f.differential()
sage: w.cartier() == w
True
```
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:**

```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
Place (x, x*y)
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:

```python
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 valuation ring modulo its unique maximal ideal. The method `residue_field()` of the valuation ring returns an extension field of the constant base field, isomorphic to the residue field, along with lifting and evaluation homomorphisms:

```python
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)
```

(continues on next page)
AUTHORS:

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

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: R = p.valuation_ring()
sage: p == R.place()
True
```

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 field

OUTPUT:

• a field isomorphic to the residue field
• a ring homomorphism from the valuation ring to the field
• a ring homomorphism from the 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
```
Maps and morphisms useful for computations with function 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
```

For global function fields, which have positive characteristics, the higher derivation is available:

```
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(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

```
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:

```python
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:

```python
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(field, place, name=None, prec=None, gen_name=None)

Bases: sage.categories.map.Map

Completions on 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:

```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: 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)
s^2 + s^3 + s^4 + s^5 + s^6 + s^7 + s^8 + s^9 + s^10 + s^11 + s^12 + s^13 + s^14 + s^15 + s^16 + s^17 + s^18 + O(s^22)
sage: m(y)
s^-1 + 1 + s^-3 + s^-5 + s^-7 + s^-9 + s^-13 + s^-15 + s^-17 + O(s^-19)
```
sage: m(x*y) == m(x) * m(y)
True
sage: m(x+y) == m(x) + m(y)
True

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:

```python
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:**

```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: m = L.completion(p)
sage: m.default_precision()
20
```

**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:**

```python
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`

Base class for derivations on function fields.

A derivation on $R$ is a map $R \rightarrow R$ 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:
sage: K.<x> = FunctionField(QQ)
sage: d = K.derivation()
sage: d.is_injective()
False

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:
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:
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

class sage.rings.function_field.maps.FunctionFieldDerivation_rational(K, u)
Bases: sage.rings.function_field.maps.FunctionFieldDerivation
Derivations on rational function fields.
EXAMPLES:
sage: K.<x> = FunctionField(QQ)
sage: K.derivation()
Derivation map:
(continues on next page)
class sage.rings.function_field.maps.FunctionFieldDerivation_separable(L, d)
Bases: sage.rings.function_field.maps.FunctionFieldDerivation

Derivations of separable extensions.

EXAMPLES:

```python
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:

```python
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_char_zero(field)
Bases: sage.rings.function_field.maps.FunctionFieldHigherDerivation

Higher derivations of function fields of characteristic zero.

INPUT:

- field – function field on which the derivation operates

EXAMPLES:

```python
sage: K.<x> = FunctionField(QQ); _.<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,1) == -(3*x^2*y+1)/(3*y^2+x^3)
```

(continues on next page)
True
sage: h(y^2, 1) == -2*y*(3*x^2*y+1)/(3*y^2+x^3)
True
sage: e = L.random_element()
sage: h(h(e, 1), 1) == 2*h(e, 2)
True
sage: h(h(h(e, 1), 1), 1) == 3*2*h(e, 3)
True

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

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.function_field.maps.FunctionFieldMorphism
    Base class for morphisms between function fields.
    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.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:
```python
sage: K.<x> = FunctionField(GF(7)); R.<y> = K[]
sage: L.<y> = K.extension(y^3 + 6*x^3 + x)
sage: f = L.hom(y*2); f
Function Field endomorphism of Function field in y defined by y^3 + 6*x^3 + x
   Defn: y |--> 2*y
sage: factor(L.polynomial())
y^3 + 6*x^3 + x
sage: f(y).charpoly('y')
y^3 + 6*x^3 + x
```

```
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.

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:

```
```
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()

is_injective()  
Return True, since the 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 the 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.MapFunctionFieldToVectorSpace(K, V)  
Bases: sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism

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:
  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 RationalField
```

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:

```
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:
```

(continues on next page)
From: Vector space of dimension 2 over Rational function field in x over Rational Field
To: Function field in y defined by \( y^2 - x^4 + 4x^3 \)

codomain()

Return the function field which is the codomain of the isomorphism.

EXAMPLES:

```sage
code: 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 \( y^2 - x^4 + 4x^3 \)
```

domain()

Return the vector space which is the domain of the isomorphism.

EXAMPLES:

```sage
code: 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
```

class sage.rings.function_field.maps.RationalFunctionFieldHigherDerivation_global(field)

Bases: sage.rings.function_field.maps.FunctionFieldHigherDerivation

Higher derivations of rational function fields over finite fields.

INPUT:

- field – function field on which the derivation operates

EXAMPLES:

```sage
code: sage: F.<x> = FunctionField(GF(2))
sage: h = F.higher_derivation()
sage: 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
sage: h(x^2,2)
1
```
CHAPTER
TEN

FACTORIES TO CONSTRUCT FUNCTION FIELDS

This module provides factories to construct function fields. These factories are only for internal use.

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
```

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`

```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: 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\(\textit{polynomial, names}\)  
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\(\textit{version, key, **extra_args}\)  
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 \texttt{sage.rings.function_field.constructor.FunctionFieldFactory}  
Bases: \texttt{sage.structure.factory.UniqueFactory}  

Return the function field in one variable with constant field \textit{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:  
• \textit{F} – field  
• \textit{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\(\textit{F, names}\)  
Given the arguments and keywords, create a key that uniquely determines this object.  

EXAMPLES:  

```
sage: K.<x> = FunctionField(QQ)  # indirect doctest
```

create_object\(\textit{version, key, **extra_args}\)  
Create the object from the key and extra arguments. This is only called if the object was not found in the cache.  

EXAMPLES:
A basic reference for the theory of algebraic function fields is [Stich2009].
11.1 Hermite form computation for function fields

This module provides an optimized implementation of the algorithm computing Hermite forms of matrices over polynomials. This is the workhorse of the function field machinery of Sage.

EXAMPLES:

```
sage: P.<x> = PolynomialRing(QQ)
sage: A = matrix(P,3,[-(x-1)^(i-j+1) % 3 for i in range(3) for j in range(3)])
sage: A
[-x + 1  -x^2 + 2*x - 1]
[-x^2 + 2*x - 1  -x + 1  -1]
[-1  -x^2 + 2*x - 1  -x + 1]
sage: from sage.rings.function_field.hermite_form_polynomial import reversed_hermite_form
sage: B = copy(A)
sage: U = reversed_hermite_form(B, transformation=True)
sage: U * A == B
True
sage: B
[x^3 - 3*x^2 + 3*x - 2  0  0]
[ 0  x^3 - 3*x^2 + 3*x - 2  0]
[x^2 - 2*x + 1  x - 1  1]
```

The function `reversed_hermite_form()` computes the reversed hermite form, which is reversed both row-wise and column-wise from the usual hermite form. Let us check it:

```
sage: A.reverse_rows_and_columns()
sage: C = copy(A.hermite_form())
sage: C.reverse_rows_and_columns()
sage: C
[x^3 - 3*x^2 + 3*x - 2  0  0]
[ 0  x^3 - 3*x^2 + 3*x - 2  0]
[x^2 - 2*x + 1  x - 1  1]
sage: C == B
True
```

AUTHORS:

- Kwankyu Lee (2021-05-21): initial version
sage.rings.function_field.hermite_form_polynomial.reversed_hermite_form(mat, transformation=False)

Transform the matrix in place to reversed hermite normal form and optionally return the transformation matrix.

INPUT:

- transformation – boolean (default: False); if True, return the transformation matrix

EXAMPLES:

```python
sage: from sage.rings.function_field.hermite_form_polynomial import reversed_hermite_form
sage: P.<x> = PolynomialRing(QQ)
sage: A = matrix(P,3,[-(x-1)^((i-2*j) % 4) for i in range(3) for j in range(3)])
sage: A
[-1 -x^2 + 2*x - 1 -1]
[-x + 1 -x^3 + 3*x^2 - 3*x + 1 -x + 1]
[-x^2 + 2*x - 1 -1 -x^2 + 2*x - 1]
sage: B = copy(A)
sage: U = reversed_hermite_form(B, transformation=True)
sage: U * A == B
True
sage: U
[0 0 0]
[0 x^4 - 4*x^3 + 6*x^2 - 4*x 0]
[1 x^2 - 2*x + 1 1]
```
INDICES AND TABLES

• Index
• Module Index
• Search Page
sage.rings.function_field.constructor, 147
sage.rings.function_field.differential, 127
sage.rings.function_field.divisor, 121
sage.rings.function_field.element, 51
sage.rings.function_field.function_field, 3
sage.rings.function_field.hermite_form_polynomial, 151
sage.rings.function_field.ideal, 89
sage.rings.function_field.maps, 137
sage.rings.function_field.order, 67
sage.rings.function_field.place, 113
sage.rings.function_field.valuation_ring, 133
 gens_two() (sage.rings.function_field.ideal.FunctionFieldIdealPolynomial (in module sage.rings.function_field.function_field), 97)

 genus() (sage.rings.function_field.function_field.FunctionFieldGlobalPolynomial (in module sage.rings.function_field.function_field), 28)

 higher_derivative() (sage.rings.function_field.function_field.RationalFunctionField (in module sage.rings.function_field.function_field), 80)

 ideal_below() (sage.rings.function_field.idea.FunctionFieldIdealGlobalPolynomial (in module sage.rings.function_field.function_field), 65)

 ideal_below() (sage.rings.function_field.ideal.FunctionFieldIdealGlobalPolynomial (in module sage.rings.function_field.function_field), 97)

 ideal() (sage.rings.function_field.order.FunctionFieldMaximalOrderPolynomial (in module sage.rings.function_field.function_field), 77)

 ideal() (sage.rings.function_field.order.FunctionFieldMaximalOrderRational (in module sage.rings.function_field.function_field), 79)

 ideal() (sage.rings.function_field.order.FunctionFieldMaximalOrderPolynomial (in module sage.rings.function_field.function_field), 71)

 ideal() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinitePolynomial (in module sage.rings.function_field.function_field), 72)

 ideal() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfiniteRational (in module sage.rings.function_field.function_field), 71)

 ideal() (sage.rings.function_field.order.FunctionFieldMaximalOrderPolynomial (in module sage.rings.function_field.function_field), 86)

 ideal() (sage.rings.function_field.order.FunctionFieldMaximalOrderInfinitePolynomial (in module sage.rings.function_field.function_field), 82)

 ideal_below() (sage.rings.function_field.ideal.FunctionFieldIdealGlobalPolynomial (in module sage.rings.function_field.function_field), 55)

 ideal_below() (sage.rings.function_field.ideal.FunctionFieldIdealGlobalPolynomial (in module sage.rings.function_field.function_field), 105)

 ideal_below() (sage.rings.function_field.ideal.FunctionFieldIdealGlobalPolynomial (in module sage.rings.function_field.function_field), 107)

 ideal_below() (sage.rings.function_field.ideal.FunctionFieldIdealGlobalPolynomial (in module sage.rings.function_field.function_field), 84)

 is_field() (sage.rings.function_field.order.FunctionFieldOrderBase (in module sage.rings.function_field.function_field), 84)

 is_field() (sage.rings.function_field.order.FunctionFieldOrderInfinitePolynomial (in module sage.rings.function_field.function_field), 82)

 is_field() (sage.rings.function_field.order.FunctionFieldOrderInfiniteRational (in module sage.rings.function_field.function_field), 80)

 is_field() (sage.rings.function_field.order.FunctionFieldOrderBasis (in module sage.rings.function_field.function_field), 86)

 is_injctive() (sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism (in module sage.rings.function_field.function_field), 117)

 is_injctive() (sage.rings.function_field.maps.FunctionFieldDerivation (in module sage.rings.function_field.function_field), 102)

 is_injctive() (sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism (in module sage.rings.function_field.function_field), 115)

 is_noetherian() (sage.rings.function_field.order.FunctionFieldOrderBasis (in module sage.rings.function_field.function_field), 84)
Index

is_nth_power() (sage.rings.function_field.element.FunctionFieldElement), 55

is_nth_power() (sage.rings.function_field.element.FunctionFieldElement_polymod, method), 60

is_nth_power() (sage.rings.function_field.element.FunctionFieldElement_rational, method), 60

is_nth_power() (sage.rings.function_field.function_field.FunctionField), 31

maximal_order() (sage.rings.function_field.function_field.FunctionField), 31

maximal_order() (sage.rings.function_field.function_field.FunctionField_polymod, module), 46

maximal_order() (sage.rings.function_field.function_field.RationalFunctionField, module), 107

maximal_order_infinite() (sage.rings.function_field.function_field.RationalFunctionField), 46

is_perfect() (sage.rings.function_field.function_field.FunctionField), 62

maximal_order_infinite() (sage.rings.function_field.function_field.RationalFunctionField), 46

is_prime() (sage.rings.function_field.function_field.RationalFunctionField, module), 100

is_prime() (sage.rings.function_field.function_field.RationalFunctionField, module), 103

is_prime() (sage.rings.function_field.function_field.RationalFunctionField, module), 107

is_prime() (sage.rings.function_field.ideal.FunctionFieldIdeal, module), 103

is_prime() (sage.rings.function_field.ideal.FunctionFieldIdealPolynomial, module), 107

is_prime() (sage.rings.function_field.ideal.FunctionFieldIdealPolynomial, module), 107

is_prime() (sage.rings.function_field.ideal.FunctionFieldIdealInfinite, module), 107

is_prime() (sage.rings.function_field.ideal.FunctionFieldIdealInfinitePolynomial, module), 107

minpoly() (sage.rings.function_field.function_field.RationalFunctionField), 107

minpoly() (sage.rings.function_field.function_field.RationalFunctionField, module), 107

is_separable() (sage.rings.function_field.function_field.FunctionField), 107

is_square() (sage.rings.function_field.function_field.FunctionField), 107

is_surjective() (sage.rings.function_field.function_field.FunctionField), 107

is_surjective() (sage.rings.function_field.maps.FunctionFieldVectorSpaceIsomorphism), 115

L

L_polynomial() (sage.rings.function_field.function_field.FunctionField), 17

list() (sage.rings.function_field.divisor.FunctionFieldDivisor, module), 124

list() (sage.rings.function_field.element.FunctionFieldElementPolynomial, module), 60

list() (sage.rings.function_field.element.FunctionFieldElementPolynomial, module), 60

local_uniformizer() (sage.rings.function_field.place.FunctionFieldPlace, module), 115

local_uniformizer() (sage.rings.function_field.place.FunctionFieldPlacePolynomial, module), 115

M

make_FunctionFieldElement() (in module sage.rings.function_field.element), 65

MapFunctionFieldToVectorSpace (class in sage.rings.function_field.maps), 144

MapVectorSpaceToFunctionField (class in sage.rings.function_field.maps), 144

matrix() (sage.rings.function_field.element.FunctionFieldElementPolynomial, method), 56

maximal_order() (sage.rings.function_field.function_field.FunctionField), 19

maximal_order() (sage.rings.function_field.function_field.RationalFunctionField, module), 46

multiplicity() (sage.rings.function_field.divisor.FunctionFieldDivisor, method), 124

monic_integral_model() (sage.rings.function_field.function_field.FunctionFieldPolynomial, module), 31

monomial_coefficients() (sage.rings.function_field.function_field.FunctionFieldPolynomial, module), 31

N

map() (sage.rings.function_field.function_field.FunctionFieldPolynomial, module), 66

map() (sage.rings.function_field.function_field.FunctionFieldPolynomial, module), 66
Index 163