Modular Abelian Varieties

Release 10.6

The Sage Development Team

Apr 01, 2025

CONTENTS

CHAPTER

ONE

CONSTRUCTORS FOR CERTAIN MODULAR ABELIAN VARIETIES

AUTHORS:

- William Stein (2007-03)

sage.modular.abvar.constructor.**AbelianVariety**(*X*)

Create the abelian variety corresponding to the given defining data.

INPUT:

- X – integer, string, newform, modsym space, congruence subgroup or tuple of congruence subgroups

OUTPUT: a modular abelian variety

EXAMPLES:

```
sage: AbelianVariety(Gamma0(37))
Abelian variety J0(37) of dimension 2
sage: AbelianVariety('37a')
Newform abelian subvariety 37a of dimension 1 of J0(37)
sage: AbelianVariety(Newform('37a'))
Newform abelian subvariety 37a of dimension 1 of J0(37)
sage: AbelianVariety(ModularSymbols(37).cuspidal_submodule())
Abelian variety J0(37) of dimension 2
sage: AbelianVariety((Gamma0(37), Gamma0(11)))
Abelian variety J0(37) x J0(11) of dimension 3
sage: AbelianVariety(37)
Abelian variety J0(37) of dimension 2
sage: AbelianVariety([1,2,3])
Traceback (most recent call last):
...
TypeError: X must be an integer, string, newform, modsym space, congruence␣
↪subgroup or tuple of congruence subgroups
```

```
>>> from sage.all import *
>>> AbelianVariety(Gamma0(Integer(37)))
Abelian variety J0(37) of dimension 2
>>> AbelianVariety('37a')
Newform abelian subvariety 37a of dimension 1 of J0(37)
>>> AbelianVariety(Newform('37a'))
Newform abelian subvariety 37a of dimension 1 of J0(37)
>>> AbelianVariety(ModularSymbols(Integer(37)).cuspidal_submodule())
Abelian variety J0(37) of dimension 2
>>> AbelianVariety((Gamma0(Integer(37)), Gamma0(Integer(11))))
```

(continues on next page)

(continued from previous page)

```
Abelian variety J0(37) x J0(11) of dimension 3
>>> AbelianVariety(Integer(37))
Abelian variety J0(37) of dimension 2
>>> AbelianVariety([Integer(1),Integer(2),Integer(3)])
Traceback (most recent call last):
...
TypeError: X must be an integer, string, newform, modsym space, congruence␣
↪subgroup or tuple of congruence subgroups
```

sage.modular.abvar.constructor.**J0**(N)

Return the Jacobian $J_0(N)$ of the modular curve $X_0(N)$.

EXAMPLES:

```
sage: J0(389)
Abelian variety J0(389) of dimension 32
```

```
>>> from sage.all import *
>>> J0(Integer(389))
Abelian variety J0(389) of dimension 32
```

The result is cached:

```
sage: J0(33) is J0(33)
True
```

```
>>> from sage.all import *
>>> J0(Integer(33)) is J0(Integer(33))
True
```

sage.modular.abvar.constructor.**J1**(N)

Return the Jacobian $J_1(N)$ of the modular curve $X_1(N)$.

EXAMPLES:

```
sage: J1(389)
Abelian variety J1(389) of dimension 6112
```

```
>>> from sage.all import *
>>> J1(Integer(389))
Abelian variety J1(389) of dimension 6112
```

sage.modular.abvar.constructor.**JH**(N, H)

Return the Jacobian $J_H(N)$ of the modular curve $X_H(N)$.

EXAMPLES:

```
sage: JH(389,[16])
Abelian variety JH(389,[16]) of dimension 64
```

```
>>> from sage.all import *
>>> JH(Integer(389),[Integer(16)])
Abelian variety JH(389,[16]) of dimension 64
```

CHAPTER

TWO

BASE CLASS FOR MODULAR ABELIAN VARIETIES

AUTHORS:

- William Stein (2007-03)

class sage.modular.abvar.abvar.**ModularAbelianVariety**(*groups*, *lattice=None*, *base_field=Rational Field*, *is_simple=None*, *newform_level=None*, *isogeny_number=None*, *number=None*, *check=True*)

Bases: *ModularAbelianVariety_abstract*

Create a modular abelian variety with given level and base field.

INPUT:

- `groups` – tuple of congruence subgroups
- `lattice` – (default: $\mathbf{Z}^n$) a full lattice in $\mathbf{Z}^n$, where n is the sum of the dimensions of the spaces of cuspidal modular symbols corresponding to each $\Gamma \in$ groups
- `base_field` – a field (default: $\mathbf{Q}$)

EXAMPLES:

```
sage: J0(23)
Abelian variety J0(23) of dimension 2
```

```
>>> from sage.all import *
>>> J0(Integer(23))
Abelian variety J0(23) of dimension 2
```

lattice()

Return the lattice that defines this abelian variety.

OUTPUT:

- `lattice` – a lattice embedded in the rational homology of the ambient product Jacobian

EXAMPLES:

```
sage: A = (J0(11) * J0(37))[1]; A
Simple abelian subvariety 37a(1,37) of dimension 1 of J0(11) x J0(37)
sage: type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_with_category'>
sage: A.lattice()
Free module of degree 6 and rank 2 over Integer Ring
```

(continues on next page)

(continued from previous page)

```
Echelon basis matrix:
[ 0  0  1 -1  1  0]
[ 0  0  0  0  2 -1]
```

```
>>> from sage.all import *
>>> A = (J0(Integer(11)) * J0(Integer(37)))[Integer(1)]; A
Simple abelian subvariety 37a(1,37) of dimension 1 of J0(11) x J0(37)
>>> type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_with_category'>
>>> A.lattice()
Free module of degree 6 and rank 2 over Integer Ring
Echelon basis matrix:
[ 0  0  1 -1  1  0]
[ 0  0  0  0  2 -1]
```

class sage.modular.abvar.abvar.**ModularAbelianVariety_abstract**(*groups*, *base_field*, *is_simple=None*, *newform_level=None*, *isogeny_number=None*, *number=None*, *check=True*)

Bases: Parent

Abstract base class for modular abelian varieties.

INPUT:

- groups – tuple of congruence subgroups
- base_field – a field
- is_simple – boolean; whether or not self is simple
- newform_level – if self is isogenous to a newform abelian variety, returns the level of that abelian variety
- isogeny_number – which isogeny class the corresponding newform is in; this corresponds to the Cremona letter code
- number – the t number of the degeneracy map that this abelian variety is the image under
- check – whether to do some type checking on the defining data

EXAMPLES: One should not create an instance of this class, but we do so anyways here as an example:

```
sage: A = sage.modular.abvar.abvar.ModularAbelianVariety_abstract((Gamma0(37),),␣
↪QQ)
sage: type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_abstract_with_category'>
```

```
>>> from sage.all import *
>>> A = sage.modular.abvar.abvar.ModularAbelianVariety_
↪abstract((Gamma0(Integer(37)),), QQ)
>>> type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_abstract_with_category'>
```

All hell breaks loose if you try to do anything with A:

```
sage: A
<repr(<sage.modular.abvar.abvar.ModularAbelianVariety_abstract_with_category at␣
↪0x...>) failed: NotImplementedError: BUG -- lattice method must be defined in␣
↪derived class>
```

```
>>> from sage.all import *
>>> A
<repr(<sage.modular.abvar.abvar.ModularAbelianVariety_abstract_with_category at␣
↪0x...>) failed: NotImplementedError: BUG -- lattice method must be defined in␣
↪derived class>
```

All instances of this class are in the category of modular abelian varieties:

```
sage: A.category()
Category of modular abelian varieties over Rational Field
sage: J0(23).category()
Category of modular abelian varieties over Rational Field
```

```
>>> from sage.all import *
>>> A.category()
Category of modular abelian varieties over Rational Field
>>> J0(Integer(23)).category()
Category of modular abelian varieties over Rational Field
```

ambient_morphism()

Return the morphism from `self` to the ambient variety. This is injective if `self` is natural a subvariety of the ambient product Jacobian.

OUTPUT: morphism

The output is cached.

EXAMPLES: We compute the ambient structure morphism for an abelian subvariety of $J_0(33)$:

```
sage: A,B,C = J0(33)
sage: phi = A.ambient_morphism()
sage: phi.domain()
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
sage: phi.codomain()
Abelian variety J0(33) of dimension 3
sage: phi.matrix()
[ 1  1 -2  0  2 -1]
[ 0  3 -2 -1  2  0]
```

```
>>> from sage.all import *
>>> A,B,C = J0(Integer(33))
>>> phi = A.ambient_morphism()
>>> phi.domain()
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> phi.codomain()
Abelian variety J0(33) of dimension 3
>>> phi.matrix()
[ 1  1 -2  0  2 -1]
[ 0  3 -2 -1  2  0]
```

`phi` is of course injective:

```
sage: phi.kernel()
(Finite subgroup with invariants [] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33),
 Abelian subvariety of dimension 0 of J0(33))
```

```
>>> from sage.all import *
>>> phi.kernel()
(Finite subgroup with invariants [] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33),
 Abelian subvariety of dimension 0 of J0(33))
```

This is the same as the basis matrix for the lattice corresponding to self:

```
sage: A.lattice()
Free module of degree 6 and rank 2 over Integer Ring
Echelon basis matrix:
[ 1  1 -2  0  2 -1]
[ 0  3 -2 -1  2  0]
```

```
>>> from sage.all import *
>>> A.lattice()
Free module of degree 6 and rank 2 over Integer Ring
Echelon basis matrix:
[ 1  1 -2  0  2 -1]
[ 0  3 -2 -1  2  0]
```

We compute a non-injective map to an ambient space:

```
sage: Q,pi = J0(33)/A
sage: phi = Q.ambient_morphism()
sage: phi.matrix()
[  1   4   1   9  -1  -1]
[  0  15   0   0  30 -75]
[  0   0   5  10  -5  15]
[  0   0   0  15 -15  30]
sage: phi.kernel()[0]
Finite subgroup with invariants [5, 15, 15] over QQ of Abelian
 variety factor of dimension 2 of J0(33)
```

```
>>> from sage.all import *
>>> Q,pi = J0(Integer(33))/A
>>> phi = Q.ambient_morphism()
>>> phi.matrix()
[  1   4   1   9  -1  -1]
[  0  15   0   0  30 -75]
[  0   0   5  10  -5  15]
[  0   0   0  15 -15  30]
>>> phi.kernel()[Integer(0)]
Finite subgroup with invariants [5, 15, 15] over QQ of Abelian
 variety factor of dimension 2 of J0(33)
```

ambient_variety()

Return the ambient modular abelian variety that contains this abelian variety. The ambient variety is always a product of Jacobians of modular curves.

OUTPUT: abelian variety

EXAMPLES:

```
sage: A = J0(33)[0]; A
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
sage: A.ambient_variety()
Abelian variety J0(33) of dimension 3
```

```
>>> from sage.all import *
>>> A = J0(Integer(33))[Integer(0)]; A
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> A.ambient_variety()
Abelian variety J0(33) of dimension 3
```

base_extend(*K*)

EXAMPLES:

```
sage: A = J0(37); A
Abelian variety J0(37) of dimension 2
sage: A.base_extend(QQbar)                                                  #␣
↪needs sage.rings.number_field
Abelian variety J0(37) over Algebraic Field of dimension 2
sage: A.base_extend(GF(7))
Abelian variety J0(37) over Finite Field of size 7 of dimension 2
```

```
>>> from sage.all import *
>>> A = J0(Integer(37)); A
Abelian variety J0(37) of dimension 2
>>> A.base_extend(QQbar)                                                    #␣
↪needs sage.rings.number_field
Abelian variety J0(37) over Algebraic Field of dimension 2
>>> A.base_extend(GF(Integer(7)))
Abelian variety J0(37) over Finite Field of size 7 of dimension 2
```

base_field()

Synonym for `self.base_ring()`.

EXAMPLES:

```
sage: J0(11).base_field()
Rational Field
```

```
>>> from sage.all import *
>>> J0(Integer(11)).base_field()
Rational Field
```

change_ring(*R*)

Change the base ring of this modular abelian variety.

EXAMPLES:

```
sage: A = J0(23)
sage: A.change_ring(QQ)
Abelian variety J0(23) of dimension 2
```

```
>>> from sage.all import *
>>> A = J0(Integer(23))
>>> A.change_ring(QQ)
Abelian variety J0(23) of dimension 2
```

complement(*A=None*)

Return a complement of this abelian variety.

INPUT:

- `A` – (default: `None`) if given, `A` must be an abelian variety that contains `self`, in which case the complement of `self` is taken inside `A`. Otherwise the complement is taken in the ambient product Jacobian.

OUTPUT: abelian variety

EXAMPLES:

```
sage: a,b,c = J0(33)
sage: (a+b).complement()
Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)
sage: (a+b).complement() == c
True
sage: a.complement(a+b)
Abelian subvariety of dimension 1 of J0(33)
```

```
>>> from sage.all import *
>>> a,b,c = J0(Integer(33))
>>> (a+b).complement()
Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)
>>> (a+b).complement() == c
True
>>> a.complement(a+b)
Abelian subvariety of dimension 1 of J0(33)
```

conductor()

Return the conductor of this abelian variety.

EXAMPLES:

```
sage: A = J0(23)
sage: A.conductor().factor()
23^2

sage: A = J1(25)
sage: A.conductor().factor()
5^24

sage: A = J0(11^2); A.decomposition()
[Simple abelian subvariety 11a(1,121) of dimension 1 of J0(121),
 Simple abelian subvariety 11a(11,121) of dimension 1 of J0(121),
```

(continues on next page)

(continued from previous page)

```
 Simple abelian subvariety 121a(1,121) of dimension 1 of J0(121),
 Simple abelian subvariety 121b(1,121) of dimension 1 of J0(121),
 Simple abelian subvariety 121c(1,121) of dimension 1 of J0(121),
 Simple abelian subvariety 121d(1,121) of dimension 1 of J0(121)]
sage: A.conductor().factor()
11^10

sage: A = J0(33)[0]; A
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
sage: A.conductor()
11
sage: A.elliptic_curve().conductor()
11
```

```
>>> from sage.all import *
>>> A = J0(Integer(23))
>>> A.conductor().factor()
23^2

>>> A = J1(Integer(25))
>>> A.conductor().factor()
5^24

>>> A = J0(Integer(11)**Integer(2)); A.decomposition()
[Simple abelian subvariety 11a(1,121) of dimension 1 of J0(121),
 Simple abelian subvariety 11a(11,121) of dimension 1 of J0(121),
 Simple abelian subvariety 121a(1,121) of dimension 1 of J0(121),
 Simple abelian subvariety 121b(1,121) of dimension 1 of J0(121),
 Simple abelian subvariety 121c(1,121) of dimension 1 of J0(121),
 Simple abelian subvariety 121d(1,121) of dimension 1 of J0(121)]
>>> A.conductor().factor()
11^10

>>> A = J0(Integer(33))[Integer(0)]; A
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> A.conductor()
11
>>> A.elliptic_curve().conductor()
11
```

cuspidal_subgroup()

Return the cuspidal subgroup of this modular abelian variety. This is the subgroup generated by rational cusps.

EXAMPLES:

```
sage: J = J0(54)
sage: C = J.cuspidal_subgroup()
sage: C.gens()
[[(1/3, 0, 0, 0, 0, 1/3, 0, 2/3)], [(0, 1/3, 0, 0, 0, 2/3, 0, 1/3)], [(0, 0,␣
↪1/9, 1/9, 1/9, 1/9, 1/9, 2/9)], [(0, 0, 0, 1/3, 0, 1/3, 0, 0)], [(0, 0, 0,␣
↪0, 1/3, 1/3, 0, 1/3)], [(0, 0, 0, 0, 0, 0, 1/3, 2/3)]]
```

(continues on next page)

(continued from previous page)

```
sage: C.invariants()
[3, 3, 3, 3, 3, 9]
sage: J1(13).cuspidal_subgroup()
Finite subgroup with invariants [19, 19] over QQ of Abelian variety J1(13) of␣
↪dimension 2
sage: A = J0(33)[0]
sage: A.cuspidal_subgroup()
Finite subgroup with invariants [5] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33)
```

```
>>> from sage.all import *
>>> J = J0(Integer(54))
>>> C = J.cuspidal_subgroup()
>>> C.gens()
[[(1/3, 0, 0, 0, 0, 1/3, 0, 2/3)], [(0, 1/3, 0, 0, 0, 2/3, 0, 1/3)], [(0, 0,␣
↪1/9, 1/9, 1/9, 1/9, 1/9, 2/9)], [(0, 0, 0, 1/3, 0, 1/3, 0, 0)], [(0, 0, 0,␣
↪0, 1/3, 1/3, 0, 1/3)], [(0, 0, 0, 0, 0, 0, 1/3, 2/3)]]
>>> C.invariants()
[3, 3, 3, 3, 3, 9]
>>> J1(Integer(13)).cuspidal_subgroup()
Finite subgroup with invariants [19, 19] over QQ of Abelian variety J1(13) of␣
↪dimension 2
>>> A = J0(Integer(33))[Integer(0)]
>>> A.cuspidal_subgroup()
Finite subgroup with invariants [5] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33)
```

decomposition(*simple=True*, *bound=None*)

Return a sequence of abelian subvarieties of `self` that are all simple, have finite intersection and sum to `self`.

INPUT:

- `simple` – boolean (default: `True`); if `True`, all factors are simple. If `False`, each factor returned is isogenous to a power of a simple and the simples in each factor are distinct.
- `bound` – integer (default: `None`); if given, only use Hecke operators up to this bound when decomposing. This can give wrong answers, so use with caution!

EXAMPLES:

```
sage: m = ModularSymbols(11).cuspidal_submodule()
sage: d1 = m.degeneracy_map(33,1).matrix(); d3=m.degeneracy_map(33,3).matrix()
sage: w = ModularSymbols(33).submodule((d1 + d3).image(), check=False)
sage: A = w.abelian_variety(); A
Abelian subvariety of dimension 1 of J0(33)
sage: D = A.decomposition(); D
[Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)]
sage: D[0] == A
True
sage: B = A + J0(33)[0]; B
Abelian subvariety of dimension 2 of J0(33)
sage: dd = B.decomposition(simple=False); dd
[Abelian subvariety of dimension 2 of J0(33)]
```

(continues on next page)

(continued from previous page)

```
sage: dd[0] == B
True
sage: dd = B.decomposition(); dd
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)]
sage: sum(dd) == B
True
```

```
>>> from sage.all import *
>>> m = ModularSymbols(Integer(11)).cuspidal_submodule()
>>> d1 = m.degeneracy_map(Integer(33),Integer(1)).matrix(); d3=m.degeneracy_
↪map(Integer(33),Integer(3)).matrix()
>>> w = ModularSymbols(Integer(33)).submodule((d1 + d3).image(), check=False)
>>> A = w.abelian_variety(); A
Abelian subvariety of dimension 1 of J0(33)
>>> D = A.decomposition(); D
[Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)]
>>> D[Integer(0)] == A
True
>>> B = A + J0(Integer(33))[Integer(0)]; B
Abelian subvariety of dimension 2 of J0(33)
>>> dd = B.decomposition(simple=False); dd
[Abelian subvariety of dimension 2 of J0(33)]
>>> dd[Integer(0)] == B
True
>>> dd = B.decomposition(); dd
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)]
>>> sum(dd) == B
True
```

We decompose a product of two Jacobians:

```
sage: (J0(33) * J0(11)).decomposition()
[Simple abelian subvariety 11a(1,11) of dimension 1 of J0(33) x J0(11),
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33) x J0(11),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33) x J0(11),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33) x J0(11)]
```

```
>>> from sage.all import *
>>> (J0(Integer(33)) * J0(Integer(11))).decomposition()
[Simple abelian subvariety 11a(1,11) of dimension 1 of J0(33) x J0(11),
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33) x J0(11),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33) x J0(11),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33) x J0(11)]
```

degen_t(*none_if_not_known=False*)

If this abelian variety is obtained via decomposition then it gets labeled with the newform label along with some information about degeneracy maps. In particular, the label ends in a pair (t, N), where N is the ambient level and t is an integer that divides the quotient of N by the newform level. This function returns the tuple (t, N), or raises a `ValueError` if `self` is not simple.

> **Note**
>
> It need not be the case that `self` is literally equal to the image of the newform abelian variety under the t-th degeneracy map. See the documentation for the label method for more details.

INPUT:

- `none_if_not_known` – (default: `False`) if `True`, return `None` instead of attempting to compute the degen map's t, if it isn't known. This `None` result is not cached.

OUTPUT: a pair (integer, integer)

EXAMPLES:

```
sage: D = J0(33).decomposition(); D
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)]
sage: D[0].degen_t()
(1, 33)
sage: D[1].degen_t()
(3, 33)
sage: D[2].degen_t()
(1, 33)
sage: J0(33).degen_t()
Traceback (most recent call last):
...
ValueError: self must be simple
```

```
>>> from sage.all import *
>>> D = J0(Integer(33)).decomposition(); D
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)]
>>> D[Integer(0)].degen_t()
(1, 33)
>>> D[Integer(1)].degen_t()
(3, 33)
>>> D[Integer(2)].degen_t()
(1, 33)
>>> J0(Integer(33)).degen_t()
Traceback (most recent call last):
...
ValueError: self must be simple
```

degeneracy_map(*M_ls*, *t_ls*)

Return the degeneracy map with domain `self` and given level/parameter. If `self.ambient_variety()` is a product of Jacobians (as opposed to a single Jacobian), then one can provide a list of new levels and parameters, corresponding to the ambient Jacobians in order. (See the examples below.)

INPUT:

- `M`, `t` – integers level and t, or
- `Mlist, tlist` – if `self` is in a nontrivial product ambient Jacobian, input consists of a list of levels and corresponding list of t's.

OUTPUT: a degeneracy map

EXAMPLES: We make several degeneracy maps related to $J_0(11)$ and $J_0(33)$ and compute their matrices.

```
sage: d1 = J0(11).degeneracy_map(33, 1); d1
Degeneracy map from Abelian variety J0(11) of dimension 1 to Abelian variety␣
↪J0(33) of dimension 3 defined by [1]
sage: d1.matrix()
[ 0 -3  2  1 -2  0]
[ 1 -2  0  1  0 -1]
sage: d2 = J0(11).degeneracy_map(33, 3); d2
Degeneracy map from Abelian variety J0(11) of dimension 1 to Abelian variety␣
↪J0(33) of dimension 3 defined by [3]
sage: d2.matrix()
[-1  0  0  0  1 -2]
[-1 -1  1 -1  1  0]
sage: d3 = J0(33).degeneracy_map(11, 1); d3
Degeneracy map from Abelian variety J0(33) of dimension 3 to Abelian variety␣
↪J0(11) of dimension 1 defined by [1]
```

```
>>> from sage.all import *
>>> d1 = J0(Integer(11)).degeneracy_map(Integer(33), Integer(1)); d1
Degeneracy map from Abelian variety J0(11) of dimension 1 to Abelian variety␣
↪J0(33) of dimension 3 defined by [1]
>>> d1.matrix()
[ 0 -3  2  1 -2  0]
[ 1 -2  0  1  0 -1]
>>> d2 = J0(Integer(11)).degeneracy_map(Integer(33), Integer(3)); d2
Degeneracy map from Abelian variety J0(11) of dimension 1 to Abelian variety␣
↪J0(33) of dimension 3 defined by [3]
>>> d2.matrix()
[-1  0  0  0  1 -2]
[-1 -1  1 -1  1  0]
>>> d3 = J0(Integer(33)).degeneracy_map(Integer(11), Integer(1)); d3
Degeneracy map from Abelian variety J0(33) of dimension 3 to Abelian variety␣
↪J0(11) of dimension 1 defined by [1]
```

He we verify that first mapping from level 11 to level 33, then back is multiplication by 4:

```
sage: d1.matrix() * d3.matrix()
[4 0]
[0 4]
```

```
>>> from sage.all import *
>>> d1.matrix() * d3.matrix()
[4 0]
[0 4]
```

We compute a more complicated degeneracy map involving nontrivial product ambient Jacobians; note that this is just the block direct sum of the two matrices at the beginning of this example:

```
sage: d = (J0(11)*J0(11)).degeneracy_map([33,33], [1,3]); d
Degeneracy map from Abelian variety J0(11) x J0(11) of dimension 2 to Abelian␣
↪variety J0(33) x J0(33) of dimension 6 defined by [1, 3]
```

(continues on next page)

(continued from previous page)

```
sage: d.matrix()
[ 0 -3  2  1 -2  0  0  0  0  0  0  0]
[ 1 -2  0  1  0 -1  0  0  0  0  0  0]
[ 0  0  0  0  0  0 -1  0  0  0  1 -2]
[ 0  0  0  0  0  0 -1 -1  1 -1  1  0]
```

```
>>> from sage.all import *
>>> d = (J0(Integer(11))*J0(Integer(11))).degeneracy_map([Integer(33),
↪Integer(33)], [Integer(1),Integer(3)]); d
Degeneracy map from Abelian variety J0(11) x J0(11) of dimension 2 to Abelian␣
↪variety J0(33) x J0(33) of dimension 6 defined by [1, 3]
>>> d.matrix()
[ 0 -3  2  1 -2  0  0  0  0  0  0  0]
[ 1 -2  0  1  0 -1  0  0  0  0  0  0]
[ 0  0  0  0  0  0 -1  0  0  0  1 -2]
[ 0  0  0  0  0  0 -1 -1  1 -1  1  0]
```

degree()

Return the degree of this abelian variety, which is the dimension of the ambient Jacobian product.

EXAMPLES:

```
sage: A = J0(23)
sage: A.dimension()
2
```

```
>>> from sage.all import *
>>> A = J0(Integer(23))
>>> A.dimension()
2
```

dimension()

Return the dimension of this abelian variety.

EXAMPLES:

```
sage: A = J0(23)
sage: A.dimension()
2
```

```
>>> from sage.all import *
>>> A = J0(Integer(23))
>>> A.dimension()
2
```

direct_product(*other*)

Compute the direct product of `self` and `other`.

INPUT:

- `self`, `other` – modular abelian varieties

OUTPUT: abelian variety

EXAMPLES:

```
sage: J0(11).direct_product(J1(13))
Abelian variety J0(11) x J1(13) of dimension 3
sage: A = J0(33)[0].direct_product(J0(33)[1]); A
Abelian subvariety of dimension 2 of J0(33) x J0(33)
sage: A.lattice()
Free module of degree 12 and rank 4 over Integer Ring
Echelon basis matrix:
[ 1  1 -2  0  2 -1  0  0  0  0  0  0]
[ 0  3 -2 -1  2  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  1  0  0  0 -1  2]
[ 0  0  0  0  0  0  0  1 -1  1  0 -2]
```

```
>>> from sage.all import *
>>> J0(Integer(11)).direct_product(J1(Integer(13)))
Abelian variety J0(11) x J1(13) of dimension 3
>>> A = J0(Integer(33))[Integer(0)].direct_
↪product(J0(Integer(33))[Integer(1)]); A
Abelian subvariety of dimension 2 of J0(33) x J0(33)
>>> A.lattice()
Free module of degree 12 and rank 4 over Integer Ring
Echelon basis matrix:
[ 1  1 -2  0  2 -1  0  0  0  0  0  0]
[ 0  3 -2 -1  2  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  1  0  0  0 -1  2]
[ 0  0  0  0  0  0  0  1 -1  1  0 -2]
```

dual()

Return the dual of this abelian variety.

OUTPUT:

- dual abelian variety
- morphism from `self` to dual
- covering morphism from J to dual

Warning

This is currently only implemented when `self` is an abelian subvariety of the ambient Jacobian product, and the complement of `self` in the ambient product Jacobian share no common factors. A more general implementation will require implementing computation of the intersection pairing on integral homology and the resulting Weil pairing on torsion.

EXAMPLES: We compute the dual of the elliptic curve newform abelian variety of level 33, and find the kernel of the modular map, which has structure $(\mathbf{Z}/3)^2$.

```
sage: A,B,C = J0(33)
sage: C
Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)
sage: Cd, f, pi = C.dual()
sage: f.matrix()
[3 0]
```

(continues on next page)

(continued from previous page)

```
[0 3]
sage: f.kernel()[0]
Finite subgroup with invariants [3, 3] over QQ of Simple abelian subvariety␣
↪33a(1,33) of dimension 1 of J0(33)
```

```
>>> from sage.all import *
>>> A,B,C = J0(Integer(33))
>>> C
Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)
>>> Cd, f, pi = C.dual()
>>> f.matrix()
[3 0]
[0 3]
>>> f.kernel()[Integer(0)]
Finite subgroup with invariants [3, 3] over QQ of Simple abelian subvariety␣
↪33a(1,33) of dimension 1 of J0(33)
```

By a theorem the modular degree must thus be 3:

```
sage: E = EllipticCurve('33a')
sage: E.modular_degree()
3
```

```
>>> from sage.all import *
>>> E = EllipticCurve('33a')
>>> E.modular_degree()
3
```

Next we compute the dual of a 2-dimensional new simple abelian subvariety of $J_0(43)$.

```
sage: A = AbelianVariety('43b'); A
Newform abelian subvariety 43b of dimension 2 of J0(43)
sage: Ad, f, pi = A.dual()
```

```
>>> from sage.all import *
>>> A = AbelianVariety('43b'); A
Newform abelian subvariety 43b of dimension 2 of J0(43)
>>> Ad, f, pi = A.dual()
```

The kernel shows that the modular degree is 2:

```
sage: f.kernel()[0]
Finite subgroup with invariants [2, 2] over QQ of Newform abelian subvariety␣
↪43b of dimension 2 of J0(43)
```

```
>>> from sage.all import *
>>> f.kernel()[Integer(0)]
Finite subgroup with invariants [2, 2] over QQ of Newform abelian subvariety␣
↪43b of dimension 2 of J0(43)
```

Unfortunately, the dual is not implemented in general:

```
sage: A = J0(22)[0]; A
Simple abelian subvariety 11a(1,22) of dimension 1 of J0(22)
sage: A.dual()
Traceback (most recent call last):
...
NotImplementedError: dual not implemented unless complement shares no simple␣
↪factors with self.
```

```
>>> from sage.all import *
>>> A = J0(Integer(22))[Integer(0)]; A
Simple abelian subvariety 11a(1,22) of dimension 1 of J0(22)
>>> A.dual()
Traceback (most recent call last):
...
NotImplementedError: dual not implemented unless complement shares no simple␣
↪factors with self.
```

elliptic_curve()

Return an elliptic curve isogenous to `self`.

If `self` is not dimension 1 with rational base ring, this raises a `ValueError`.

The elliptic curve is found by looking it up in the CremonaDatabase. The CremonaDatabase contains all curves up to some large conductor. If a curve is not found in the CremonaDatabase, a `RuntimeError` will be raised. In practice, only the most committed users will see this `RuntimeError`.

OUTPUT: an elliptic curve isogenous to `self`

EXAMPLES:

```
sage: J = J0(11)
sage: J.elliptic_curve()
Elliptic Curve defined by y^2 + y = x^3 - x^2 - 10*x - 20 over Rational Field

sage: J = J0(49)
sage: J.elliptic_curve()
Elliptic Curve defined by y^2 + x*y = x^3 - x^2 - 2*x - 1 over Rational Field

sage: A = J0(37)[1]
sage: E = A.elliptic_curve()
sage: A.lseries()(1)
0.725681061936153
sage: E.lseries()(1)
0.725681061936153
```

```
>>> from sage.all import *
>>> J = J0(Integer(11))
>>> J.elliptic_curve()
Elliptic Curve defined by y^2 + y = x^3 - x^2 - 10*x - 20 over Rational Field

>>> J = J0(Integer(49))
>>> J.elliptic_curve()
Elliptic Curve defined by y^2 + x*y = x^3 - x^2 - 2*x - 1 over Rational Field
```

(continues on next page)

(continued from previous page)

```
>>> A = J0(Integer(37))[Integer(1)]
>>> E = A.elliptic_curve()
>>> A.lseries()(Integer(1))
0.725681061936153
>>> E.lseries()(Integer(1))
0.725681061936153
```

Elliptic curves are of dimension 1.

```
sage: J = J0(23)
sage: J.elliptic_curve()
Traceback (most recent call last):
...
ValueError: self must be of dimension 1
```

```
>>> from sage.all import *
>>> J = J0(Integer(23))
>>> J.elliptic_curve()
Traceback (most recent call last):
...
ValueError: self must be of dimension 1
```

This is only implemented for curves over QQ.

```
sage: J = J0(11).change_ring(CC)
sage: J.elliptic_curve()
Traceback (most recent call last):
...
ValueError: base ring must be QQ
```

```
>>> from sage.all import *
>>> J = J0(Integer(11)).change_ring(CC)
>>> J.elliptic_curve()
Traceback (most recent call last):
...
ValueError: base ring must be QQ
```

endomorphism_ring(*category=None*)

Return the endomorphism ring of `self`.

OUTPUT: b = self.sturm_bound()

EXAMPLES: We compute a few endomorphism rings:

```
sage: J0(11).endomorphism_ring()
Endomorphism ring of Abelian variety J0(11) of dimension 1
sage: J0(37).endomorphism_ring()
Endomorphism ring of Abelian variety J0(37) of dimension 2
sage: J0(33)[2].endomorphism_ring()
Endomorphism ring of Simple abelian subvariety 33a(1,33) of dimension 1 of␣
↪J0(33)
```

```
>>> from sage.all import *
>>> J0(Integer(11)).endomorphism_ring()
Endomorphism ring of Abelian variety J0(11) of dimension 1
>>> J0(Integer(37)).endomorphism_ring()
Endomorphism ring of Abelian variety J0(37) of dimension 2
>>> J0(Integer(33))[Integer(2)].endomorphism_ring()
Endomorphism ring of Simple abelian subvariety 33a(1,33) of dimension 1 of␣
↪J0(33)
```

No real computation is done:

```
sage: J1(123456).endomorphism_ring()
Endomorphism ring of Abelian variety J1(123456) of dimension 423185857
```

```
>>> from sage.all import *
>>> J1(Integer(123456)).endomorphism_ring()
Endomorphism ring of Abelian variety J1(123456) of dimension 423185857
```

finite_subgroup(*X*, *field_of_definition=None*, *check=True*)

Return a finite subgroup of this modular abelian variety.

INPUT:

- `X` – list of elements of other finite subgroups of this modular abelian variety or elements that coerce into the rational homology (viewed as a rational vector space); also X could be a finite subgroup itself that is contained in this abelian variety.
- `field_of_definition` – (default: `None`) field over which this group is defined. If `None` try to figure out the best base field.

OUTPUT: a finite subgroup of a modular abelian variety

EXAMPLES:

```
sage: J = J0(11)
sage: J.finite_subgroup([[1/5,0], [0,1/3]])
Finite subgroup with invariants [15] over QQbar of Abelian variety J0(11) of␣
↪dimension 1
```

```
>>> from sage.all import *
>>> J = J0(Integer(11))
>>> J.finite_subgroup([[Integer(1)/Integer(5),Integer(0)], [Integer(0),
↪Integer(1)/Integer(3)]])
Finite subgroup with invariants [15] over QQbar of Abelian variety J0(11) of␣
↪dimension 1
```

```
sage: J = J0(33); C = J[0].cuspidal_subgroup(); C
Finite subgroup with invariants [5] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33)
sage: J.finite_subgroup([[0,0,0,0,0,1/6]])
Finite subgroup with invariants [6] over QQbar of Abelian variety J0(33) of␣
↪dimension 3
sage: J.finite_subgroup(C)
Finite subgroup with invariants [5] over QQ of Abelian variety J0(33) of␣
↪dimension 3
```

```
>>> from sage.all import *
>>> J = J0(Integer(33)); C = J[Integer(0)].cuspidal_subgroup(); C
Finite subgroup with invariants [5] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33)
>>> J.finite_subgroup([[Integer(0),Integer(0),Integer(0),Integer(0),
↪Integer(0),Integer(1)/Integer(6)]])
Finite subgroup with invariants [6] over QQbar of Abelian variety J0(33) of␣
↪dimension 3
>>> J.finite_subgroup(C)
Finite subgroup with invariants [5] over QQ of Abelian variety J0(33) of␣
↪dimension 3
```

This method gives a way of changing the ambient abelian variety of a finite subgroup. This caused an issue in Issue #6392 but should be fixed now.

```
sage: A, B = J0(43)
sage: G, _ = A.intersection(B)
sage: G
Finite subgroup with invariants [2, 2] over QQ of Simple abelian subvariety␣
↪43a(1,43) of dimension 1 of J0(43)
sage: B.finite_subgroup(G)
Finite subgroup with invariants [2, 2] over QQ of Simple abelian subvariety␣
↪43b(1,43) of dimension 2 of J0(43)
```

```
>>> from sage.all import *
>>> A, B = J0(Integer(43))
>>> G, _ = A.intersection(B)
>>> G
Finite subgroup with invariants [2, 2] over QQ of Simple abelian subvariety␣
↪43a(1,43) of dimension 1 of J0(43)
>>> B.finite_subgroup(G)
Finite subgroup with invariants [2, 2] over QQ of Simple abelian subvariety␣
↪43b(1,43) of dimension 2 of J0(43)
```

free_module()

Synonym for `self.lattice()`.

OUTPUT: a free module over $\mathbf{Z}$

EXAMPLES:

```
sage: J0(37).free_module()
Ambient free module of rank 4 over the principal ideal domain Integer Ring
sage: J0(37)[0].free_module()
Free module of degree 4 and rank 2 over Integer Ring
Echelon basis matrix:
[ 1 -1  1  0]
[ 0  0  2 -1]
```

```
>>> from sage.all import *
>>> J0(Integer(37)).free_module()
Ambient free module of rank 4 over the principal ideal domain Integer Ring
>>> J0(Integer(37))[Integer(0)].free_module()
```

(continues on next page)

(continued from previous page)

```
Free module of degree 4 and rank 2 over Integer Ring
Echelon basis matrix:
[ 1 -1  1  0]
[ 0  0  2 -1]
```

frobenius_polynomial(*p*, *var='x'*)

Compute the frobenius polynomial at p.

INPUT:

- p – prime number

OUTPUT: a monic integral polynomial

EXAMPLES:

```
sage: f = Newform('39b','a')
sage: A = AbelianVariety(f)
sage: A.frobenius_polynomial(5)
x^4 + 2*x^2 + 25

sage: J = J0(23)
sage: J.frobenius_polynomial(997)
x^4 + 20*x^3 + 1374*x^2 + 19940*x + 994009

sage: J = J0(33)
sage: J.frobenius_polynomial(7)
x^6 + 9*x^4 - 16*x^3 + 63*x^2 + 343

sage: J = J0(19)
sage: J.frobenius_polynomial(3, var='y')
y^2 + 2*y + 3

sage: J = J0(3); J
Abelian variety J0(3) of dimension 0
sage: J.frobenius_polynomial(11)
1

sage: A = J1(27)[1]; A
Simple abelian subvariety 27bG1(1,27) of dimension 12 of J1(27)
sage: A.frobenius_polynomial(11)
x^24 - 3*x^23 - 15*x^22 + 126*x^21 - 201*x^20 - 1488*x^19 + 7145*x^18 -␣
↪1530*x^17 - 61974*x^16 + 202716*x^15 - 19692*x^14 - 1304451*x^13 +␣
↪4526883*x^12 - 14348961*x^11 - 2382732*x^10 + 269814996*x^9 - 907361334*x^8␣
↪- 246408030*x^7 + 12657803345*x^6 - 28996910448*x^5 - 43086135081*x^4 +␣
↪297101409066*x^3 - 389061369015*x^2 - 855935011833*x + 3138428376721

sage: J = J1(33)
sage: J.frobenius_polynomial(11)
Traceback (most recent call last):
...
ValueError: p must not divide the level of self
sage: J.frobenius_polynomial(4)
Traceback (most recent call last):
```

(continues on next page)

(continued from previous page)

```
...
ValueError: p must be prime
```

```
>>> from sage.all import *
>>> f = Newform('39b','a')
>>> A = AbelianVariety(f)
>>> A.frobenius_polynomial(Integer(5))
x^4 + 2*x^2 + 25

>>> J = J0(Integer(23))
>>> J.frobenius_polynomial(Integer(997))
x^4 + 20*x^3 + 1374*x^2 + 19940*x + 994009

>>> J = J0(Integer(33))
>>> J.frobenius_polynomial(Integer(7))
x^6 + 9*x^4 - 16*x^3 + 63*x^2 + 343

>>> J = J0(Integer(19))
>>> J.frobenius_polynomial(Integer(3), var='y')
y^2 + 2*y + 3

>>> J = J0(Integer(3)); J
Abelian variety J0(3) of dimension 0
>>> J.frobenius_polynomial(Integer(11))
1

>>> A = J1(Integer(27))[Integer(1)]; A
Simple abelian subvariety 27bG1(1,27) of dimension 12 of J1(27)
>>> A.frobenius_polynomial(Integer(11))
x^24 - 3*x^23 - 15*x^22 + 126*x^21 - 201*x^20 - 1488*x^19 + 7145*x^18 -␣
↪1530*x^17 - 61974*x^16 + 202716*x^15 - 19692*x^14 - 1304451*x^13 +␣
↪4526883*x^12 - 14348961*x^11 - 2382732*x^10 + 269814996*x^9 - 907361334*x^8␣
↪- 246408030*x^7 + 12657803345*x^6 - 28996910448*x^5 - 43086135081*x^4 +␣
↪297101409066*x^3 - 389061369015*x^2 - 855935011833*x + 3138428376721

>>> J = J1(Integer(33))
>>> J.frobenius_polynomial(Integer(11))
Traceback (most recent call last):
...
ValueError: p must not divide the level of self
>>> J.frobenius_polynomial(Integer(4))
Traceback (most recent call last):
...
ValueError: p must be prime
```

groups()

Return an ordered tuple of the congruence subgroups that the ambient product Jacobian is attached to.

Every modular abelian variety is a finite quotient of an abelian subvariety of a product of modular Jacobians J_Γ. This function returns a tuple containing the groups Γ.

EXAMPLES:

```
sage: A = (J0(37) * J1(13))[0]; A
Simple abelian subvariety 13aG1(1,13) of dimension 2 of J0(37) x J1(13)
sage: A.groups()
(Congruence Subgroup Gamma0(37), Congruence Subgroup Gamma1(13))
```

```
>>> from sage.all import *
>>> A = (J0(Integer(37)) * J1(Integer(13)))[Integer(0)]; A
Simple abelian subvariety 13aG1(1,13) of dimension 2 of J0(37) x J1(13)
>>> A.groups()
(Congruence Subgroup Gamma0(37), Congruence Subgroup Gamma1(13))
```

hecke_operator(*n*)

Return the n-th Hecke operator on the modular abelian variety, if this makes sense [[elaborate]]. Otherwise raise a `ValueError`.

EXAMPLES: We compute T_2 on $J_0(37)$.

```
sage: t2 = J0(37).hecke_operator(2); t2
Hecke operator T_2 on Abelian variety J0(37) of dimension 2
sage: t2.charpoly().factor()
x * (x + 2)
sage: t2.index()
2
```

```
>>> from sage.all import *
>>> t2 = J0(Integer(37)).hecke_operator(Integer(2)); t2
Hecke operator T_2 on Abelian variety J0(37) of dimension 2
>>> t2.charpoly().factor()
x * (x + 2)
>>> t2.index()
2
```

Note that there is no matrix associated to Hecke operators on modular abelian varieties. For a matrix, instead consider, e.g., the Hecke operator on integral or rational homology.

```
sage: t2.action_on_homology().matrix()
[-1  1  1 -1]
[ 1 -1  1  0]
[ 0  0 -2  1]
[ 0  0  0  0]
```

```
>>> from sage.all import *
>>> t2.action_on_homology().matrix()
[-1  1  1 -1]
[ 1 -1  1  0]
[ 0  0 -2  1]
[ 0  0  0  0]
```

hecke_polynomial(*n*, *var='x'*)

Return the characteristic polynomial of the n-th Hecke operator T_n acting on `self`. Raises an `ArithmeticError` if `self` is not Hecke equivariant.

INPUT:

- `n` – integer ≥ 1
- `var` – string (default: `'x'`); valid variable name

EXAMPLES:

```
sage: J0(33).hecke_polynomial(2)
x^3 + 3*x^2 - 4
sage: f = J0(33).hecke_polynomial(2, 'y'); f
y^3 + 3*y^2 - 4
sage: f.parent()
Univariate Polynomial Ring in y over Rational Field
sage: J0(33)[2].hecke_polynomial(3)
x + 1
sage: J0(33)[0].hecke_polynomial(5)
x - 1
sage: J0(33)[0].hecke_polynomial(11)
x - 1
sage: J0(33)[0].hecke_polynomial(3)
Traceback (most recent call last):
...
ArithmeticError: subspace is not invariant under matrix
```

```
>>> from sage.all import *
>>> J0(Integer(33)).hecke_polynomial(Integer(2))
x^3 + 3*x^2 - 4
>>> f = J0(Integer(33)).hecke_polynomial(Integer(2), 'y'); f
y^3 + 3*y^2 - 4
>>> f.parent()
Univariate Polynomial Ring in y over Rational Field
>>> J0(Integer(33))[Integer(2)].hecke_polynomial(Integer(3))
x + 1
>>> J0(Integer(33))[Integer(0)].hecke_polynomial(Integer(5))
x - 1
>>> J0(Integer(33))[Integer(0)].hecke_polynomial(Integer(11))
x - 1
>>> J0(Integer(33))[Integer(0)].hecke_polynomial(Integer(3))
Traceback (most recent call last):
...
ArithmeticError: subspace is not invariant under matrix
```

homology(*base_ring=Integer Ring*)

Return the homology of this modular abelian variety.

> **Warning**
>
> For efficiency reasons the basis of the integral homology need not be the same as the basis for the rational homology.

EXAMPLES:

```
sage: J0(389).homology(GF(7))
Homology with coefficients in Finite Field of size 7 of Abelian variety␣
```

(continues on next page)

(continued from previous page)

```
↪J0(389) of dimension 32
sage: J0(389).homology(QQ)
Rational Homology of Abelian variety J0(389) of dimension 32
sage: J0(389).homology(ZZ)
Integral Homology of Abelian variety J0(389) of dimension 32
```

```
>>> from sage.all import *
>>> J0(Integer(389)).homology(GF(Integer(7)))
Homology with coefficients in Finite Field of size 7 of Abelian variety␣
↪J0(389) of dimension 32
>>> J0(Integer(389)).homology(QQ)
Rational Homology of Abelian variety J0(389) of dimension 32
>>> J0(Integer(389)).homology(ZZ)
Integral Homology of Abelian variety J0(389) of dimension 32
```

in_same_ambient_variety(*other*)

Return `True` if `self` and `other` are abelian subvarieties of the same ambient product Jacobian.

EXAMPLES:

```
sage: A,B,C = J0(33)
sage: A.in_same_ambient_variety(B)
True
sage: A.in_same_ambient_variety(J0(11))
False
```

```
>>> from sage.all import *
>>> A,B,C = J0(Integer(33))
>>> A.in_same_ambient_variety(B)
True
>>> A.in_same_ambient_variety(J0(Integer(11)))
False
```

integral_homology()

Return the integral homology of this modular abelian variety.

EXAMPLES:

```
sage: H = J0(43).integral_homology(); H
Integral Homology of Abelian variety J0(43) of dimension 3
sage: H.rank()
6
sage: H = J1(17).integral_homology(); H
Integral Homology of Abelian variety J1(17) of dimension 5
sage: H.rank()
10
```

```
>>> from sage.all import *
>>> H = J0(Integer(43)).integral_homology(); H
Integral Homology of Abelian variety J0(43) of dimension 3
>>> H.rank()
6
```

(continues on next page)

(continued from previous page)

```
>>> H = J1(Integer(17)).integral_homology(); H
Integral Homology of Abelian variety J1(17) of dimension 5
>>> H.rank()
10
```

If you just ask for the rank of the homology, no serious calculations are done, so the following is fast:

```
sage: H = J0(50000).integral_homology(); H
Integral Homology of Abelian variety J0(50000) of dimension 7351
sage: H.rank()
14702
```

```
>>> from sage.all import *
>>> H = J0(Integer(50000)).integral_homology(); H
Integral Homology of Abelian variety J0(50000) of dimension 7351
>>> H.rank()
14702
```

A product:

```
sage: H = (J0(11) * J1(13)).integral_homology()
sage: H.hecke_operator(2)
Hecke operator T_2 on Integral Homology of Abelian variety J0(11) x J1(13) of␣
↪dimension 3
sage: H.hecke_operator(2).matrix()
[-2  0  0  0  0  0]
[ 0 -2  0  0  0  0]
[ 0  0 -1 -1 -1  1]
[ 0  0  1 -2 -1  0]
[ 0  0  0  0 -2  1]
[ 0  0  0  0 -1 -1]
```

```
>>> from sage.all import *
>>> H = (J0(Integer(11)) * J1(Integer(13))).integral_homology()
>>> H.hecke_operator(Integer(2))
Hecke operator T_2 on Integral Homology of Abelian variety J0(11) x J1(13) of␣
↪dimension 3
>>> H.hecke_operator(Integer(2)).matrix()
[-2  0  0  0  0  0]
[ 0 -2  0  0  0  0]
[ 0  0 -1 -1 -1  1]
[ 0  0  1 -2 -1  0]
[ 0  0  0  0 -2  1]
[ 0  0  0  0 -1 -1]
```

intersection(*other*)

Return the intersection of `self` and `other` inside a common ambient Jacobian product.

When `other` is a modular abelian variety, the output will be a tuple `(G, A)`, where `G` is a finite subgroup that surjects onto the component group and `A` is the identity component. So in particular, the intersection is the variety `G+A`. Note that `G` is not chosen in any canonical way. When `other` is a finite group, the intersection will be returned as a finite group.

INPUT:

- `other` – a modular abelian variety or a finite group

OUTPUT: if other is a modular abelian variety:

- `G` – finite subgroup of self
- `A` – abelian variety (identity component of intersection)

If other is a finite group:

- `G` – a finite group

EXAMPLES: We intersect some abelian varieties with finite intersection.

```
sage: J = J0(37)
sage: J[0].intersection(J[1])
(Finite subgroup with invariants [2, 2] over QQ of Simple abelian
 subvariety 37a(1,37) of dimension 1 of J0(37), Simple abelian
 subvariety of dimension 0 of J0(37))
```

```
>>> from sage.all import *
>>> J = J0(Integer(37))
>>> J[Integer(0)].intersection(J[Integer(1)])
(Finite subgroup with invariants [2, 2] over QQ of Simple abelian
 subvariety 37a(1,37) of dimension 1 of J0(37), Simple abelian
 subvariety of dimension 0 of J0(37))
```

```
sage: D = list(J0(65)); D
[Simple abelian subvariety 65a(1,65) of dimension 1 of J0(65),
 Simple abelian subvariety 65b(1,65) of dimension 2 of J0(65),
 Simple abelian subvariety 65c(1,65) of dimension 2 of J0(65)]
sage: D[0].intersection(D[1])
(Finite subgroup with invariants [2] over QQ of Simple abelian
 subvariety 65a(1,65) of dimension 1 of J0(65), Simple abelian
 subvariety of dimension 0 of J0(65))
sage: (D[0]+D[1]).intersection(D[1]+D[2])
(Finite subgroup with invariants [2] over QQbar of Abelian
 subvariety of dimension 3 of J0(65), Abelian subvariety of
 dimension 2 of J0(65))
```

```
>>> from sage.all import *
>>> D = list(J0(Integer(65))); D
[Simple abelian subvariety 65a(1,65) of dimension 1 of J0(65),
 Simple abelian subvariety 65b(1,65) of dimension 2 of J0(65),
 Simple abelian subvariety 65c(1,65) of dimension 2 of J0(65)]
>>> D[Integer(0)].intersection(D[Integer(1)])
(Finite subgroup with invariants [2] over QQ of Simple abelian
 subvariety 65a(1,65) of dimension 1 of J0(65), Simple abelian
 subvariety of dimension 0 of J0(65))
>>> (D[Integer(0)]+D[Integer(1)]).intersection(D[Integer(1)]+D[Integer(2)])
(Finite subgroup with invariants [2] over QQbar of Abelian
 subvariety of dimension 3 of J0(65), Abelian subvariety of
 dimension 2 of J0(65))
```

```
sage: J = J0(33)
sage: J[0].intersection(J[1])
(Finite subgroup with invariants [5] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33), Simple abelian subvariety of dimension␣
↪0 of J0(33))
```

```
>>> from sage.all import *
>>> J = J0(Integer(33))
>>> J[Integer(0)].intersection(J[Integer(1)])
(Finite subgroup with invariants [5] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33), Simple abelian subvariety of dimension␣
↪0 of J0(33))
```

Next we intersect two abelian varieties with non-finite intersection:

```
sage: J = J0(67); D = J.decomposition(); D
[Simple abelian subvariety 67a(1,67) of dimension 1 of J0(67),
 Simple abelian subvariety 67b(1,67) of dimension 2 of J0(67),
 Simple abelian subvariety 67c(1,67) of dimension 2 of J0(67)]
sage: (D[0] + D[1]).intersection(D[1] + D[2])
(Finite subgroup with invariants [5, 10] over QQbar of Abelian subvariety of␣
↪dimension 3 of J0(67), Abelian subvariety of dimension 2 of J0(67))
```

```
>>> from sage.all import *
>>> J = J0(Integer(67)); D = J.decomposition(); D
[Simple abelian subvariety 67a(1,67) of dimension 1 of J0(67),
 Simple abelian subvariety 67b(1,67) of dimension 2 of J0(67),
 Simple abelian subvariety 67c(1,67) of dimension 2 of J0(67)]
>>> (D[Integer(0)] + D[Integer(1)]).intersection(D[Integer(1)] +␣
↪D[Integer(2)])
(Finite subgroup with invariants [5, 10] over QQbar of Abelian subvariety of␣
↪dimension 3 of J0(67), Abelian subvariety of dimension 2 of J0(67))
```

When the intersection is infinite, the output is `(G, A)`, where `G` surjects onto the component group. This choice of `G` is not canonical (see Issue #26189). In this following example, `B` is a subvariety of `J`:

```
sage: d1 = J0(11).degeneracy_map(22, 1)
sage: d2 = J0(11).degeneracy_map(22, 2)
sage: B = (d1-d2).image()
sage: J = J0(22)
sage: J.intersection(B)
(Finite subgroup with invariants [] over QQbar of Abelian variety J0(22) of␣
↪dimension 2,
 Abelian subvariety of dimension 1 of J0(22))
sage: G, B = B.intersection(J); G, B
(Finite subgroup with invariants [2] over QQbar of Abelian subvariety of␣
↪dimension 1 of J0(22),
 Abelian subvariety of dimension 1 of J0(22))
sage: G.is_subgroup(B)
True
```

```
>>> from sage.all import *
```

(continues on next page)

(continued from previous page)

```
>>> d1 = J0(Integer(11)).degeneracy_map(Integer(22), Integer(1))
>>> d2 = J0(Integer(11)).degeneracy_map(Integer(22), Integer(2))
>>> B = (d1-d2).image()
>>> J = J0(Integer(22))
>>> J.intersection(B)
(Finite subgroup with invariants [] over QQbar of Abelian variety J0(22) of␣
↪dimension 2,
 Abelian subvariety of dimension 1 of J0(22))
>>> G, B = B.intersection(J); G, B
(Finite subgroup with invariants [2] over QQbar of Abelian subvariety of␣
↪dimension 1 of J0(22),
 Abelian subvariety of dimension 1 of J0(22))
>>> G.is_subgroup(B)
True
```

is_J0()

Return whether or not `self` is of the form J0(N).

OUTPUT: boolean

EXAMPLES:

```
sage: J0(23).is_J0()
True
sage: J1(11).is_J0()
False
sage: (J0(23) * J1(11)).is_J0()
False
sage: J0(37)[0].is_J0()
False
sage: (J0(23) * J0(21)).is_J0()
False
```

```
>>> from sage.all import *
>>> J0(Integer(23)).is_J0()
True
>>> J1(Integer(11)).is_J0()
False
>>> (J0(Integer(23)) * J1(Integer(11))).is_J0()
False
>>> J0(Integer(37))[Integer(0)].is_J0()
False
>>> (J0(Integer(23)) * J0(Integer(21))).is_J0()
False
```

is_J1()

Return whether or not `self` is of the form J1(N).

OUTPUT: boolean

EXAMPLES:

```
sage: J1(23).is_J1()
True
```

(continues on next page)

(continued from previous page)

```
sage: J0(23).is_J1()
False
sage: (J1(11) * J1(13)).is_J1()
False
sage: (J1(11) * J0(13)).is_J1()
False
sage: J1(23)[0].is_J1()
False
```

```
>>> from sage.all import *
>>> J1(Integer(23)).is_J1()
True
>>> J0(Integer(23)).is_J1()
False
>>> (J1(Integer(11)) * J1(Integer(13))).is_J1()
False
>>> (J1(Integer(11)) * J0(Integer(13))).is_J1()
False
>>> J1(Integer(23))[Integer(0)].is_J1()
False
```

is_ambient()

Return `True` if `self` equals the ambient product Jacobian.

OUTPUT: boolean

EXAMPLES:

```
sage: A,B,C = J0(33)
sage: A.is_ambient()
False
sage: J0(33).is_ambient()
True
sage: (A+B).is_ambient()
False
sage: (A+B+C).is_ambient()
True
```

```
>>> from sage.all import *
>>> A,B,C = J0(Integer(33))
>>> A.is_ambient()
False
>>> J0(Integer(33)).is_ambient()
True
>>> (A+B).is_ambient()
False
>>> (A+B+C).is_ambient()
True
```

is_hecke_stable()

Return `True` if `self` is stable under the Hecke operators of its ambient Jacobian.

OUTPUT: boolean

EXAMPLES:

```
sage: J0(11).is_hecke_stable()
True
sage: J0(33)[2].is_hecke_stable()
True
sage: J0(33)[0].is_hecke_stable()
False
sage: (J0(33)[0] + J0(33)[1]).is_hecke_stable()
True
```

```
>>> from sage.all import *
>>> J0(Integer(11)).is_hecke_stable()
True
>>> J0(Integer(33))[Integer(2)].is_hecke_stable()
True
>>> J0(Integer(33))[Integer(0)].is_hecke_stable()
False
>>> (J0(Integer(33))[Integer(0)] + J0(Integer(33))[Integer(1)]).is_hecke_
↪stable()
True
```

is_simple(*none_if_not_known=False*)

Return whether or not this modular abelian variety is simple, i.e., has no proper nonzero abelian subvarieties.

INPUT:

- `none_if_not_known` – boolean (default: `False`); if `True` then this function may return `None` instead of `True` of `False` if we don't already know whether or not `self` is simple.

EXAMPLES:

```
sage: J0(5).is_simple(none_if_not_known=True) is None  # this may fail if␣
↪J0(5) comes up elsewhere...
True
sage: J0(33).is_simple()
False
sage: J0(33).is_simple(none_if_not_known=True)
False
sage: J0(33)[1].is_simple()
True
sage: J1(17).is_simple()
False
```

```
>>> from sage.all import *
>>> J0(Integer(5)).is_simple(none_if_not_known=True) is None  # this may fail␣
↪if J0(5) comes up elsewhere...
True
>>> J0(Integer(33)).is_simple()
False
>>> J0(Integer(33)).is_simple(none_if_not_known=True)
False
>>> J0(Integer(33))[Integer(1)].is_simple()
True
```

(continues on next page)

(continued from previous page)

```
>>> J1(Integer(17)).is_simple()
False
```

is_subvariety(*other*)

Return `True` if `self` is a subvariety of other as they sit in a common ambient modular Jacobian. In particular, this function will only return `True` if `self` and `other` have exactly the same ambient Jacobians.

EXAMPLES:

```
sage: J = J0(37); J
Abelian variety J0(37) of dimension 2
sage: A = J[0]; A
Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37)
sage: A.is_subvariety(A)
True
sage: A.is_subvariety(J)
True
```

```
>>> from sage.all import *
>>> J = J0(Integer(37)); J
Abelian variety J0(37) of dimension 2
>>> A = J[Integer(0)]; A
Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37)
>>> A.is_subvariety(A)
True
>>> A.is_subvariety(J)
True
```

is_subvariety_of_ambient_jacobian()

Return `True` if `self` is (presented as) a subvariety of the ambient product Jacobian.

Every abelian variety in Sage is a quotient of a subvariety of an ambient Jacobian product by a finite subgroup.

EXAMPLES:

```
sage: J0(33).is_subvariety_of_ambient_jacobian()
True
sage: A = J0(33)[0]; A
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
sage: A.is_subvariety_of_ambient_jacobian()
True
sage: B, phi = A / A.torsion_subgroup(2)
sage: B
Abelian variety factor of dimension 1 of J0(33)
sage: phi.matrix()
[2 0]
[0 2]
sage: B.is_subvariety_of_ambient_jacobian()
False
```

```
>>> from sage.all import *
>>> J0(Integer(33)).is_subvariety_of_ambient_jacobian()
True
```

(continues on next page)

(continued from previous page)

```
>>> A = J0(Integer(33))[Integer(0)]; A
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> A.is_subvariety_of_ambient_jacobian()
True
>>> B, phi = A / A.torsion_subgroup(Integer(2))
>>> B
Abelian variety factor of dimension 1 of J0(33)
>>> phi.matrix()
[2 0]
[0 2]
>>> B.is_subvariety_of_ambient_jacobian()
False
```

isogeny_number(*none_if_not_known=False*)

Return the number (starting at 0) of the isogeny class of new simple abelian varieties that `self` is in.

If `self` is not simple, this raises a `ValueError` exception.

INPUT:

- `none_if_not_known` – boolean (default: `False`); if `True` then this function may return `None` instead of `True` or `False` if we do not already know the isogeny number of `self`.

EXAMPLES: We test the `none_if_not_known` flag first:

```
sage: J0(33).isogeny_number(none_if_not_known=True) is None
True
```

```
>>> from sage.all import *
>>> J0(Integer(33)).isogeny_number(none_if_not_known=True) is None
True
```

Of course, $J_0(33)$ is not simple, so this function raises a `ValueError`:

```
sage: J0(33).isogeny_number()
Traceback (most recent call last):
...
ValueError: self must be simple
```

```
>>> from sage.all import *
>>> J0(Integer(33)).isogeny_number()
Traceback (most recent call last):
...
ValueError: self must be simple
```

Each simple factor has isogeny number 1, since that's the number at which the factor is new.

```
sage: J0(33)[1].isogeny_number()
0
sage: J0(33)[2].isogeny_number()
0
```

```
>>> from sage.all import *
>>> J0(Integer(33))[Integer(1)].isogeny_number()
```

(continues on next page)

(continued from previous page)

```
0
>>> J0(Integer(33))[Integer(2)].isogeny_number()
0
```

Next consider $J_0(37)$ where there are two distinct newform factors:

```
sage: J0(37)[1].isogeny_number()
1
```

```
>>> from sage.all import *
>>> J0(Integer(37))[Integer(1)].isogeny_number()
1
```

label()

Return the label associated to this modular abelian variety.

The format of the label is [level][isogeny class][group](t, ambient level)

If this abelian variety B has the above label, this implies only that B is isogenous to the newform abelian variety A_f associated to the newform with label [level][isogeny class][group]. The [group] is empty for $\Gamma_0(N)$, is G1 for $\Gamma_1(N)$ and is GH[...] for $\Gamma_H(N)$.

Warning

The sum of $\delta_s(A_f)$ for all $s \mid t$ contains A, but no sum for a proper divisor of t contains A. It need *not* be the case that B is equal to $\delta_t(A_f)$!!!

OUTPUT: string

EXAMPLES:

```
sage: J0(11).label()
'11a(1,11)'
sage: J0(11)[0].label()
'11a(1,11)'
sage: J0(33)[2].label()
'33a(1,33)'
sage: J0(22).label()
Traceback (most recent call last):
...
ValueError: self must be simple
```

```
>>> from sage.all import *
>>> J0(Integer(11)).label()
'11a(1,11)'
>>> J0(Integer(11))[Integer(0)].label()
'11a(1,11)'
>>> J0(Integer(33))[Integer(2)].label()
'33a(1,33)'
>>> J0(Integer(22)).label()
Traceback (most recent call last):
```

(continues on next page)

(continued from previous page)

```
...
ValueError: self must be simple
```

We illustrate that `self` need not equal $\delta_t(A_f)$:

```
sage: J = J0(11); phi = J.degeneracy_map(33, 1) + J.degeneracy_map(33,3)
sage: B = phi.image(); B
Abelian subvariety of dimension 1 of J0(33)
sage: B.decomposition()
[Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)]
sage: C = J.degeneracy_map(33,3).image(); C
Abelian subvariety of dimension 1 of J0(33)
sage: C == B
False
```

```
>>> from sage.all import *
>>> J = J0(Integer(11)); phi = J.degeneracy_map(Integer(33), Integer(1)) + J.
↪degeneracy_map(Integer(33),Integer(3))
>>> B = phi.image(); B
Abelian subvariety of dimension 1 of J0(33)
>>> B.decomposition()
[Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)]
>>> C = J.degeneracy_map(Integer(33),Integer(3)).image(); C
Abelian subvariety of dimension 1 of J0(33)
>>> C == B
False
```

lattice()

Return lattice in ambient cuspidal modular symbols product that defines this modular abelian variety.

This must be defined in each derived class.

OUTPUT: a free module over $\mathbf{Z}$

EXAMPLES:

```
sage: A = sage.modular.abvar.abvar.ModularAbelianVariety_abstract((Gamma0(37),
↪), QQ)
sage: A
<repr(<sage.modular.abvar.abvar.ModularAbelianVariety_abstract_with_category␣
↪at 0x...>) failed: NotImplementedError: BUG -- lattice method must be␣
↪defined in derived class>
```

```
>>> from sage.all import *
>>> A = sage.modular.abvar.abvar.ModularAbelianVariety_
↪abstract((Gamma0(Integer(37)),), QQ)
>>> A
<repr(<sage.modular.abvar.abvar.ModularAbelianVariety_abstract_with_category␣
↪at 0x...>) failed: NotImplementedError: BUG -- lattice method must be␣
↪defined in derived class>
```

level()

Return the level of this modular abelian variety, which is an integer N (usually minimal) such that this modular

abelian variety is a quotient of $J_1(N)$. In the case that the ambient variety of `self` is a product of Jacobians, return the LCM of their levels.

EXAMPLES:

```
sage: J1(5077).level()
5077
sage: JH(389,[4]).level()
389
sage: (J0(11)*J0(17)).level()
187
```

```
>>> from sage.all import *
>>> J1(Integer(5077)).level()
5077
>>> JH(Integer(389),[Integer(4)]).level()
389
>>> (J0(Integer(11))*J0(Integer(17))).level()
187
```

lseries()

Return the complex L-series of this modular abelian variety.

EXAMPLES:

```
sage: A = J0(37)
sage: A.lseries()
Complex L-series attached to Abelian variety J0(37) of dimension 2
```

```
>>> from sage.all import *
>>> A = J0(Integer(37))
>>> A.lseries()
Complex L-series attached to Abelian variety J0(37) of dimension 2
```

modular_degree()

Return the modular degree of this abelian variety, which is the square root of the degree of the modular kernel.

EXAMPLES:

```
sage: A = AbelianVariety('37a')
sage: A.modular_degree()
2
```

```
>>> from sage.all import *
>>> A = AbelianVariety('37a')
>>> A.modular_degree()
2
```

modular_kernel()

Return the modular kernel of this abelian variety, which is the kernel of the canonical polarization of `self`.

EXAMPLES:

```
sage: A = AbelianVariety('33a'); A
Newform abelian subvariety 33a of dimension 1 of J0(33)
sage: A.modular_kernel()
Finite subgroup with invariants [3, 3] over QQ of Newform abelian subvariety␣
↪33a of dimension 1 of J0(33)
```

```
>>> from sage.all import *
>>> A = AbelianVariety('33a'); A
Newform abelian subvariety 33a of dimension 1 of J0(33)
>>> A.modular_kernel()
Finite subgroup with invariants [3, 3] over QQ of Newform abelian subvariety␣
↪33a of dimension 1 of J0(33)
```

newform(*names=None*)

Return the newform f such that this abelian variety is isogenous to the newform abelian variety A_f.

If this abelian variety is not simple, this raises a `ValueError`.

INPUT:

- `names` – (default: `None`) if the newform has coefficients in a number field, then a generator name must be specified

OUTPUT: a newform f so that `self` is isogenous to A_f

EXAMPLES:

```
sage: J0(11).newform()
q - 2*q^2 - q^3 + 2*q^4 + q^5 + O(q^6)

sage: f = J0(23).newform(names='a')
sage: AbelianVariety(f) == J0(23)
True

sage: J = J0(33)
sage: [s.newform('a') for s in J.decomposition()]
[q - 2*q^2 - q^3 + 2*q^4 + q^5 + O(q^6),
 q - 2*q^2 - q^3 + 2*q^4 + q^5 + O(q^6),
 q + q^2 - q^3 - q^4 - 2*q^5 + O(q^6)]
```

```
>>> from sage.all import *
>>> J0(Integer(11)).newform()
q - 2*q^2 - q^3 + 2*q^4 + q^5 + O(q^6)

>>> f = J0(Integer(23)).newform(names='a')
>>> AbelianVariety(f) == J0(Integer(23))
True

>>> J = J0(Integer(33))
>>> [s.newform('a') for s in J.decomposition()]
[q - 2*q^2 - q^3 + 2*q^4 + q^5 + O(q^6),
 q - 2*q^2 - q^3 + 2*q^4 + q^5 + O(q^6),
 q + q^2 - q^3 - q^4 - 2*q^5 + O(q^6)]
```

The following fails since $J_0(33)$ is not simple:

```
sage: J0(33).newform()
Traceback (most recent call last):
...
ValueError: self must be simple
```

```
>>> from sage.all import *
>>> J0(Integer(33)).newform()
Traceback (most recent call last):
...
ValueError: self must be simple
```

newform_decomposition(*names=None*)

Return the newforms of the simple subvarieties in the decomposition of `self` as a product of simple subvarieties, up to isogeny.

OUTPUT: an array of newforms

EXAMPLES:

```
sage: J = J1(11) * J0(23)
sage: J.newform_decomposition('a')
[q - 2*q^2 - q^3 + 2*q^4 + q^5 + O(q^6),
q + a0*q^2 + (-2*a0 - 1)*q^3 + (-a0 - 1)*q^4 + 2*a0*q^5 + O(q^6)]
```

```
>>> from sage.all import *
>>> J = J1(Integer(11)) * J0(Integer(23))
>>> J.newform_decomposition('a')
[q - 2*q^2 - q^3 + 2*q^4 + q^5 + O(q^6),
q + a0*q^2 + (-2*a0 - 1)*q^3 + (-a0 - 1)*q^4 + 2*a0*q^5 + O(q^6)]
```

newform_label()

Return the label [level][isogeny class][group] of the newform f such that this abelian variety is isogenous to the newform abelian variety A_f.

If this abelian variety is not simple, this raises a `ValueError`.

OUTPUT: string

EXAMPLES:

```
sage: J0(11).newform_label()
'11a'
sage: J0(33)[2].newform_label()
'33a'
```

```
>>> from sage.all import *
>>> J0(Integer(11)).newform_label()
'11a'
>>> J0(Integer(33))[Integer(2)].newform_label()
'33a'
```

The following fails since $J_0(33)$ is not simple:

```
sage: J0(33).newform_label()
Traceback (most recent call last):
...
ValueError: self must be simple
```

```
>>> from sage.all import *
>>> J0(Integer(33)).newform_label()
Traceback (most recent call last):
...
ValueError: self must be simple
```

newform_level(*none_if_not_known=False*)

Write `self` as a product (up to isogeny) of newform abelian varieties A_f. Then this function return the least common multiple of the levels of the newforms f, along with the corresponding group or list of groups (the groups do not appear with multiplicity).

INPUT:

- `none_if_not_known` – boolean (default: `False`); if `True`, return `None` instead of attempting to compute the newform level, if it isn't already known. This None result is not cached.

OUTPUT: integer group or list of distinct groups

EXAMPLES:

```
sage: J0(33)[0].newform_level()
(11, Congruence Subgroup Gamma0(33))
sage: J0(33)[0].newform_level(none_if_not_known=True)
(11, Congruence Subgroup Gamma0(33))
```

```
>>> from sage.all import *
>>> J0(Integer(33))[Integer(0)].newform_level()
(11, Congruence Subgroup Gamma0(33))
>>> J0(Integer(33))[Integer(0)].newform_level(none_if_not_known=True)
(11, Congruence Subgroup Gamma0(33))
```

Here there are multiple groups since there are in fact multiple newforms:

```
sage: (J0(11) * J1(13)).newform_level()
(143, [Congruence Subgroup Gamma0(11), Congruence Subgroup Gamma1(13)])
```

```
>>> from sage.all import *
>>> (J0(Integer(11)) * J1(Integer(13))).newform_level()
(143, [Congruence Subgroup Gamma0(11), Congruence Subgroup Gamma1(13)])
```

number_of_rational_points()

Return the number of rational points of this modular abelian variety.

It is not always possible to compute the order of the torsion subgroup. The BSD conjecture is assumed to compute the algebraic rank.

OUTPUT: a positive integer or infinity

EXAMPLES:

```
sage: J0(23).number_of_rational_points()
11
sage: J0(29).number_of_rational_points()
7
sage: J0(37).number_of_rational_points()
+Infinity

sage: J0(12); J0(12).number_of_rational_points()
Abelian variety J0(12) of dimension 0
1

sage: J1(17).number_of_rational_points()
584

sage: J1(16).number_of_rational_points()
Traceback (most recent call last):
...
RuntimeError: Unable to compute order of torsion subgroup (it is in [1, 2, 4,␣
↪5, 10, 20])
```

```
>>> from sage.all import *
>>> J0(Integer(23)).number_of_rational_points()
11
>>> J0(Integer(29)).number_of_rational_points()
7
>>> J0(Integer(37)).number_of_rational_points()
+Infinity

>>> J0(Integer(12)); J0(Integer(12)).number_of_rational_points()
Abelian variety J0(12) of dimension 0
1

>>> J1(Integer(17)).number_of_rational_points()
584

>>> J1(Integer(16)).number_of_rational_points()
Traceback (most recent call last):
...
RuntimeError: Unable to compute order of torsion subgroup (it is in [1, 2, 4,␣
↪5, 10, 20])
```

padic_lseries(*p*)

Return the p-adic L-series of this modular abelian variety.

EXAMPLES:

```
sage: A = J0(37)
sage: A.padic_lseries(7)
7-adic L-series attached to Abelian variety J0(37) of dimension 2
```

```
>>> from sage.all import *
>>> A = J0(Integer(37))
>>> A.padic_lseries(Integer(7))
```

(continues on next page)

(continued from previous page)

```
7-adic L-series attached to Abelian variety J0(37) of dimension 2
```

project_to_factor(*n*)

If `self` is an ambient product of Jacobians, return a projection from `self` to the n-th such Jacobian.

EXAMPLES:

```
sage: J = J0(33)
sage: J.project_to_factor(0)
Abelian variety endomorphism of Abelian variety J0(33) of dimension 3
```

```
>>> from sage.all import *
>>> J = J0(Integer(33))
>>> J.project_to_factor(Integer(0))
Abelian variety endomorphism of Abelian variety J0(33) of dimension 3
```

```
sage: J = J0(33) * J0(37) * J0(11)
sage: J.project_to_factor(2)
Abelian variety morphism:
  From: Abelian variety J0(33) x J0(37) x J0(11) of dimension 6
  To:   Abelian variety J0(11) of dimension 1
sage: J.project_to_factor(2).matrix()
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 1]
```

```
>>> from sage.all import *
>>> J = J0(Integer(33)) * J0(Integer(37)) * J0(Integer(11))
>>> J.project_to_factor(Integer(2))
Abelian variety morphism:
  From: Abelian variety J0(33) x J0(37) x J0(11) of dimension 6
  To:   Abelian variety J0(11) of dimension 1
>>> J.project_to_factor(Integer(2)).matrix()
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
```

(continues on next page)

(continued from previous page)

```
[1 0]
[0 1]
```

projection(*A*, *check=True*)

Given an abelian subvariety A of self, return a projection morphism from `self` to A. Note that this morphism need not be unique.

INPUT:

- `A` – an abelian variety

OUTPUT: a morphism

EXAMPLES:

```
sage: a,b,c = J0(33)
sage: pi = J0(33).projection(a); pi.matrix()
[ 3 -2]
[-5  5]
[-4  1]
[ 3 -2]
[ 5  0]
[ 1  1]
sage: pi = (a+b).projection(a); pi.matrix()
[ 0  0]
[-3  2]
[-4  1]
[-1 -1]
sage: pi = a.projection(a); pi.matrix()
[1 0]
[0 1]
```

```
>>> from sage.all import *
>>> a,b,c = J0(Integer(33))
>>> pi = J0(Integer(33)).projection(a); pi.matrix()
[ 3 -2]
[-5  5]
[-4  1]
[ 3 -2]
[ 5  0]
[ 1  1]
>>> pi = (a+b).projection(a); pi.matrix()
[ 0  0]
[-3  2]
[-4  1]
[-1 -1]
>>> pi = a.projection(a); pi.matrix()
[1 0]
[0 1]
```

We project onto a factor in a product of two Jacobians:

```
sage: A = J0(11)*J0(11); A
Abelian variety J0(11) x J0(11) of dimension 2
```

(continues on next page)

(continued from previous page)

```
sage: A[0]
Simple abelian subvariety 11a(1,11) of dimension 1 of J0(11) x J0(11)
sage: A.projection(A[0])
Abelian variety morphism:
  From: Abelian variety J0(11) x J0(11) of dimension 2
  To:   Simple abelian subvariety 11a(1,11) of dimension 1 of J0(11) x J0(11)
sage: A.projection(A[0]).matrix()
[0 0]
[0 0]
[1 0]
[0 1]
sage: A.projection(A[1]).matrix()
[1 0]
[0 1]
[0 0]
[0 0]
```

```
>>> from sage.all import *
>>> A = J0(Integer(11))*J0(Integer(11)); A
Abelian variety J0(11) x J0(11) of dimension 2
>>> A[Integer(0)]
Simple abelian subvariety 11a(1,11) of dimension 1 of J0(11) x J0(11)
>>> A.projection(A[Integer(0)])
Abelian variety morphism:
  From: Abelian variety J0(11) x J0(11) of dimension 2
  To:   Simple abelian subvariety 11a(1,11) of dimension 1 of J0(11) x J0(11)
>>> A.projection(A[Integer(0)]).matrix()
[0 0]
[0 0]
[1 0]
[0 1]
>>> A.projection(A[Integer(1)]).matrix()
[1 0]
[0 1]
[0 0]
[0 0]
```

qbar_torsion_subgroup()

Return the group of all points of finite order in the algebraic closure of this abelian variety.

EXAMPLES:

```
sage: T = J0(33).qbar_torsion_subgroup(); T
Group of all torsion points in QQbar on Abelian variety J0(33) of dimension 3
```

```
>>> from sage.all import *
>>> T = J0(Integer(33)).qbar_torsion_subgroup(); T
Group of all torsion points in QQbar on Abelian variety J0(33) of dimension 3
```

The field of definition is the same as the base field of the abelian variety.

```
sage: T.field_of_definition()
Rational Field
```

```
>>> from sage.all import *
>>> T.field_of_definition()
Rational Field
```

On the other hand, T is a module over **Z**.

```
sage: T.base_ring()
Integer Ring
```

```
>>> from sage.all import *
>>> T.base_ring()
Integer Ring
```

quotient(*other*, ***kwds*)

Compute the quotient of `self` and `other`, where other is either an abelian subvariety of `self` or a finite subgroup of `self`.

INPUT:

- `other` – a finite subgroup or subvariety
- further named arguments, that are currently ignored

OUTPUT: a pair (A, phi) with phi the quotient map from `self` to A

EXAMPLES: We quotient $J_0(33)$ out by an abelian subvariety:

```
sage: Q, f = J0(33).quotient(J0(33)[0])
sage: Q
Abelian variety factor of dimension 2 of J0(33)
sage: f
Abelian variety morphism:
  From: Abelian variety J0(33) of dimension 3
  To:   Abelian variety factor of dimension 2 of J0(33)
```

```
>>> from sage.all import *
>>> Q, f = J0(Integer(33)).quotient(J0(Integer(33))[Integer(0)])
>>> Q
Abelian variety factor of dimension 2 of J0(33)
>>> f
Abelian variety morphism:
  From: Abelian variety J0(33) of dimension 3
  To:   Abelian variety factor of dimension 2 of J0(33)
```

We quotient $J_0(33)$ by the cuspidal subgroup:

```
sage: C = J0(33).cuspidal_subgroup()
sage: Q, f = J0(33).quotient(C)
sage: Q
Abelian variety factor of dimension 3 of J0(33)
sage: f.kernel()[0]
Finite subgroup with invariants [10, 10] over QQ of Abelian variety J0(33) of␣
```

(continues on next page)

(continued from previous page)

```
↪dimension 3
sage: C
Finite subgroup with invariants [10, 10] over QQ of Abelian variety J0(33) of␣
↪dimension 3
sage: J0(11).direct_product(J1(13))
Abelian variety J0(11) x J1(13) of dimension 3
```

```
>>> from sage.all import *
>>> C = J0(Integer(33)).cuspidal_subgroup()
>>> Q, f = J0(Integer(33)).quotient(C)
>>> Q
Abelian variety factor of dimension 3 of J0(33)
>>> f.kernel()[Integer(0)]
Finite subgroup with invariants [10, 10] over QQ of Abelian variety J0(33) of␣
↪dimension 3
>>> C
Finite subgroup with invariants [10, 10] over QQ of Abelian variety J0(33) of␣
↪dimension 3
>>> J0(Integer(11)).direct_product(J1(Integer(13)))
Abelian variety J0(11) x J1(13) of dimension 3
```

rank()

Return the rank of the underlying lattice of `self`.

EXAMPLES:

```
sage: J = J0(33)
sage: J.rank()
6
sage: J[1]
Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)
sage: (J[1] * J[1]).rank()
4
```

```
>>> from sage.all import *
>>> J = J0(Integer(33))
>>> J.rank()
6
>>> J[Integer(1)]
Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)
>>> (J[Integer(1)] * J[Integer(1)]).rank()
4
```

rational_cusp_subgroup()

Return the subgroup of this modular abelian variety generated by rational cusps.

This is a subgroup of the group of rational points in the cuspidal subgroup.

Warning

This is only currently implemented for $\Gamma_0(N)$.

EXAMPLES:

```
sage: J = J0(54)
sage: CQ = J.rational_cusp_subgroup(); CQ
Finite subgroup with invariants [3, 3, 9] over QQ of Abelian variety J0(54)␣
↪of dimension 4
sage: CQ.gens()
[[(1/3, 0, 0, 1/3, 2/3, 1/3, 0, 1/3)], [(0, 0, 1/9, 1/9, 7/9, 7/9, 1/9, 8/9)],
↪ [(0, 0, 0, 0, 0, 0, 1/3, 2/3)]]
sage: factor(CQ.order())
3^4
sage: CQ.invariants()
[3, 3, 9]
```

```
>>> from sage.all import *
>>> J = J0(Integer(54))
>>> CQ = J.rational_cusp_subgroup(); CQ
Finite subgroup with invariants [3, 3, 9] over QQ of Abelian variety J0(54)␣
↪of dimension 4
>>> CQ.gens()
[[(1/3, 0, 0, 1/3, 2/3, 1/3, 0, 1/3)], [(0, 0, 1/9, 1/9, 7/9, 7/9, 1/9, 8/9)],
↪ [(0, 0, 0, 0, 0, 0, 1/3, 2/3)]]
>>> factor(CQ.order())
3^4
>>> CQ.invariants()
[3, 3, 9]
```

In this example the rational cuspidal subgroup and the cuspidal subgroup differ by a lot.

```
sage: J = J0(49)
sage: J.cuspidal_subgroup()
Finite subgroup with invariants [2, 14] over QQ of Abelian variety J0(49) of␣
↪dimension 1
sage: J.rational_cusp_subgroup()
Finite subgroup with invariants [2] over QQ of Abelian variety J0(49) of␣
↪dimension 1
```

```
>>> from sage.all import *
>>> J = J0(Integer(49))
>>> J.cuspidal_subgroup()
Finite subgroup with invariants [2, 14] over QQ of Abelian variety J0(49) of␣
↪dimension 1
>>> J.rational_cusp_subgroup()
Finite subgroup with invariants [2] over QQ of Abelian variety J0(49) of␣
↪dimension 1
```

Note that computation of the rational cusp subgroup isn't implemented for Γ_1.

```
sage: J = J1(13)
sage: J.cuspidal_subgroup()
Finite subgroup with invariants [19, 19] over QQ of Abelian variety J1(13) of␣
↪dimension 2
sage: J.rational_cusp_subgroup()
```

(continues on next page)

(continued from previous page)

```
Traceback (most recent call last):
...
NotImplementedError: computation of rational cusps only implemented in Gamma0␣
↪case.
```

```
>>> from sage.all import *
>>> J = J1(Integer(13))
>>> J.cuspidal_subgroup()
Finite subgroup with invariants [19, 19] over QQ of Abelian variety J1(13) of␣
↪dimension 2
>>> J.rational_cusp_subgroup()
Traceback (most recent call last):
...
NotImplementedError: computation of rational cusps only implemented in Gamma0␣
↪case.
```

rational_cuspidal_subgroup()

Return the rational subgroup of the cuspidal subgroup of this modular abelian variety.

This is a subgroup of the group of rational points in the cuspidal subgroup.

Warning

This is only currently implemented for $\Gamma_0(N)$.

EXAMPLES:

```
sage: J = J0(54)
sage: CQ = J.rational_cuspidal_subgroup(); CQ
Finite subgroup with invariants [3, 3, 9] over QQ of Abelian variety J0(54)␣
↪of dimension 4
sage: CQ.gens()
[[(1/3, 0, 0, 1/3, 2/3, 1/3, 0, 1/3)], [(0, 0, 1/9, 1/9, 7/9, 7/9, 1/9, 8/9)],
↪ [(0, 0, 0, 0, 0, 0, 1/3, 2/3)]]
sage: factor(CQ.order())
3^4
sage: CQ.invariants()
[3, 3, 9]
```

```
>>> from sage.all import *
>>> J = J0(Integer(54))
>>> CQ = J.rational_cuspidal_subgroup(); CQ
Finite subgroup with invariants [3, 3, 9] over QQ of Abelian variety J0(54)␣
↪of dimension 4
>>> CQ.gens()
[[(1/3, 0, 0, 1/3, 2/3, 1/3, 0, 1/3)], [(0, 0, 1/9, 1/9, 7/9, 7/9, 1/9, 8/9)],
↪ [(0, 0, 0, 0, 0, 0, 1/3, 2/3)]]
>>> factor(CQ.order())
3^4
>>> CQ.invariants()
[3, 3, 9]
```

In this example the rational cuspidal subgroup and the cuspidal subgroup differ by a lot.

```
sage: J = J0(49)
sage: J.cuspidal_subgroup()
Finite subgroup with invariants [2, 14] over QQ of Abelian variety J0(49) of␣
↪dimension 1
sage: J.rational_cuspidal_subgroup()
Finite subgroup with invariants [2] over QQ of Abelian variety J0(49) of␣
↪dimension 1
```

```
>>> from sage.all import *
>>> J = J0(Integer(49))
>>> J.cuspidal_subgroup()
Finite subgroup with invariants [2, 14] over QQ of Abelian variety J0(49) of␣
↪dimension 1
>>> J.rational_cuspidal_subgroup()
Finite subgroup with invariants [2] over QQ of Abelian variety J0(49) of␣
↪dimension 1
```

Note that computation of the rational cusp subgroup isn't implemented for Γ_1.

```
sage: J = J1(13)
sage: J.cuspidal_subgroup()
Finite subgroup with invariants [19, 19] over QQ of Abelian variety J1(13) of␣
↪dimension 2
sage: J.rational_cuspidal_subgroup()
Traceback (most recent call last):
...
NotImplementedError: only implemented when group is Gamma0
```

```
>>> from sage.all import *
>>> J = J1(Integer(13))
>>> J.cuspidal_subgroup()
Finite subgroup with invariants [19, 19] over QQ of Abelian variety J1(13) of␣
↪dimension 2
>>> J.rational_cuspidal_subgroup()
Traceback (most recent call last):
...
NotImplementedError: only implemented when group is Gamma0
```

rational_homology()

Return the rational homology of this modular abelian variety.

EXAMPLES:

```
sage: H = J0(37).rational_homology(); H
Rational Homology of Abelian variety J0(37) of dimension 2
sage: H.rank()
4
sage: H.base_ring()
Rational Field
sage: H = J1(17).rational_homology(); H
Rational Homology of Abelian variety J1(17) of dimension 5
sage: H.rank()
```

(continues on next page)

(continued from previous page)

```
10
sage: H.base_ring()
Rational Field
```

```
>>> from sage.all import *
>>> H = J0(Integer(37)).rational_homology(); H
Rational Homology of Abelian variety J0(37) of dimension 2
>>> H.rank()
4
>>> H.base_ring()
Rational Field
>>> H = J1(Integer(17)).rational_homology(); H
Rational Homology of Abelian variety J1(17) of dimension 5
>>> H.rank()
10
>>> H.base_ring()
Rational Field
```

rational_torsion_order(*proof=True*)

Return the order of the rational torsion subgroup of this modular abelian variety.

This function is really an alias for *order()* See the docstring there for a more in-depth reference and more interesting examples.

INPUT:

- `proof` – boolean (default: `True`)

OUTPUT:

The order of the rational torsion subgroup of this modular abelian variety.

EXAMPLES:

```
sage: J0(11).rational_torsion_subgroup().order()
5
sage: J0(11).rational_torsion_order()
5
```

```
>>> from sage.all import *
>>> J0(Integer(11)).rational_torsion_subgroup().order()
5
>>> J0(Integer(11)).rational_torsion_order()
5
```

rational_torsion_subgroup()

Return the maximal torsion subgroup of `self` defined over $\mathbf{Q}$.

EXAMPLES:

```
sage: J = J0(33)
sage: A = J.new_subvariety()
sage: A
Abelian subvariety of dimension 1 of J0(33)
sage: t = A.rational_torsion_subgroup(); t
```

(continues on next page)

(continued from previous page)

```
Torsion subgroup of Abelian subvariety of dimension 1 of J0(33)
sage: t.multiple_of_order()
4
sage: t.divisor_of_order()
4
sage: t.order()
4
sage: t.gens()
[[(1/2, 0, 0, -1/2, 0, 0)], [(0, 0, 1/2, 0, 1/2, -1/2)]]
```

```
>>> from sage.all import *
>>> J = J0(Integer(33))
>>> A = J.new_subvariety()
>>> A
Abelian subvariety of dimension 1 of J0(33)
>>> t = A.rational_torsion_subgroup(); t
Torsion subgroup of Abelian subvariety of dimension 1 of J0(33)
>>> t.multiple_of_order()
4
>>> t.divisor_of_order()
4
>>> t.order()
4
>>> t.gens()
[[(1/2, 0, 0, -1/2, 0, 0)], [(0, 0, 1/2, 0, 1/2, -1/2)]]
```

shimura_subgroup()

Return the Shimura subgroup of this modular abelian variety.

This is the kernel of $J_0(N) \to J_1(N)$ under the natural map.

Here we compute the Shimura subgroup as the kernel of $J_0(N) \to J_0(Np)$ where the map is the difference between the two degeneracy maps.

EXAMPLES:

```
sage: J = J0(11)
sage: J.shimura_subgroup()
Finite subgroup with invariants [5] over QQ of Abelian variety J0(11) of␣
↪dimension 1

sage: J = J0(17)
sage: G = J.cuspidal_subgroup(); G
Finite subgroup with invariants [4] over QQ of Abelian variety J0(17) of␣
↪dimension 1
sage: S = J.shimura_subgroup(); S
Finite subgroup with invariants [4] over QQ of Abelian variety J0(17) of␣
↪dimension 1
sage: G.intersection(S)
Finite subgroup with invariants [2] over QQ of Abelian variety J0(17) of␣
↪dimension 1

sage: J = J0(33)
```

(continues on next page)

(continued from previous page)

```
sage: A = J.decomposition()[0]
sage: A.shimura_subgroup()
Finite subgroup with invariants [5] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33)
sage: J.shimura_subgroup()
Finite subgroup with invariants [10] over QQ of Abelian variety J0(33) of␣
↪dimension 3
```

```
>>> from sage.all import *
>>> J = J0(Integer(11))
>>> J.shimura_subgroup()
Finite subgroup with invariants [5] over QQ of Abelian variety J0(11) of␣
↪dimension 1

>>> J = J0(Integer(17))
>>> G = J.cuspidal_subgroup(); G
Finite subgroup with invariants [4] over QQ of Abelian variety J0(17) of␣
↪dimension 1
>>> S = J.shimura_subgroup(); S
Finite subgroup with invariants [4] over QQ of Abelian variety J0(17) of␣
↪dimension 1
>>> G.intersection(S)
Finite subgroup with invariants [2] over QQ of Abelian variety J0(17) of␣
↪dimension 1

>>> J = J0(Integer(33))
>>> A = J.decomposition()[Integer(0)]
>>> A.shimura_subgroup()
Finite subgroup with invariants [5] over QQ of Simple abelian subvariety␣
↪11a(1,33) of dimension 1 of J0(33)
>>> J.shimura_subgroup()
Finite subgroup with invariants [10] over QQ of Abelian variety J0(33) of␣
↪dimension 3
```

sturm_bound()

Return a bound B such that all Hecke operators T_n for $n \leq B$ generate the Hecke algebra.

OUTPUT: integer

EXAMPLES:

```
sage: J0(11).sturm_bound()
2
sage: J0(33).sturm_bound()
8
sage: J1(17).sturm_bound()
48
sage: J1(123456).sturm_bound()
1693483008
sage: JH(37,[2,3]).sturm_bound()
7
sage: J1(37).sturm_bound()
228
```

```
>>> from sage.all import *
>>> J0(Integer(11)).sturm_bound()
2
>>> J0(Integer(33)).sturm_bound()
8
>>> J1(Integer(17)).sturm_bound()
48
>>> J1(Integer(123456)).sturm_bound()
1693483008
>>> JH(Integer(37),[Integer(2),Integer(3)]).sturm_bound()
7
>>> J1(Integer(37)).sturm_bound()
228
```

torsion_subgroup(n)

If n is an integer, return the subgroup of points of order n. Return the n-torsion subgroup of elements of order dividing n of this modular abelian variety A, i.e., the group $A[n]$.

EXAMPLES:

```
sage: J1(13).torsion_subgroup(19)
Finite subgroup with invariants [19, 19, 19, 19] over QQ of Abelian variety␣
↪J1(13) of dimension 2
```

```
>>> from sage.all import *
>>> J1(Integer(13)).torsion_subgroup(Integer(19))
Finite subgroup with invariants [19, 19, 19, 19] over QQ of Abelian variety␣
↪J1(13) of dimension 2
```

```
sage: A = J0(23)
sage: G = A.torsion_subgroup(5); G
Finite subgroup with invariants [5, 5, 5, 5] over QQ of Abelian variety␣
↪J0(23) of dimension 2
sage: G.order()
625
sage: G.gens()
[[(1/5, 0, 0, 0)], [(0, 1/5, 0, 0)], [(0, 0, 1/5, 0)], [(0, 0, 0, 1/5)]]
sage: A = J0(23)
sage: A.torsion_subgroup(2).order()
16
```

```
>>> from sage.all import *
>>> A = J0(Integer(23))
>>> G = A.torsion_subgroup(Integer(5)); G
Finite subgroup with invariants [5, 5, 5, 5] over QQ of Abelian variety␣
↪J0(23) of dimension 2
>>> G.order()
625
>>> G.gens()
[[(1/5, 0, 0, 0)], [(0, 1/5, 0, 0)], [(0, 0, 1/5, 0)], [(0, 0, 0, 1/5)]]
>>> A = J0(Integer(23))
>>> A.torsion_subgroup(Integer(2)).order()
16
```

vector_space()

Return vector space corresponding to the modular abelian variety.

This is the lattice tensored with $\mathbf{Q}$.

EXAMPLES:

```
sage: J0(37).vector_space()
Vector space of dimension 4 over Rational Field
sage: J0(37)[0].vector_space()
Vector space of degree 4 and dimension 2 over Rational Field
Basis matrix:
[   1   -1    0  1/2]
[   0    0    1 -1/2]
```

```
>>> from sage.all import *
>>> J0(Integer(37)).vector_space()
Vector space of dimension 4 over Rational Field
>>> J0(Integer(37))[Integer(0)].vector_space()
Vector space of degree 4 and dimension 2 over Rational Field
Basis matrix:
[   1   -1    0  1/2]
[   0    0    1 -1/2]
```

zero_subgroup()

Return the zero subgroup of this modular abelian variety, as a finite group.

EXAMPLES:

```
sage: A =J0(54); G = A.zero_subgroup(); G
Finite subgroup with invariants [] over QQ of Abelian variety J0(54) of␣
↪dimension 4
sage: G.is_subgroup(A)
True
```

```
>>> from sage.all import *
>>> A =J0(Integer(54)); G = A.zero_subgroup(); G
Finite subgroup with invariants [] over QQ of Abelian variety J0(54) of␣
↪dimension 4
>>> G.is_subgroup(A)
True
```

zero_subvariety()

Return the zero subvariety of `self`.

EXAMPLES:

```
sage: J = J0(37)
sage: J.zero_subvariety()
Simple abelian subvariety of dimension 0 of J0(37)
sage: J.zero_subvariety().level()
37
sage: J.zero_subvariety().newform_level()
(1, [])
```

```
sage: from sage.all import *
sage: J = J0(Integer(37))
sage: J.zero_subvariety()
Simple abelian subvariety of dimension 0 of J0(37)
sage: J.zero_subvariety().level()
37
sage: J.zero_subvariety().newform_level()
(1, [])
```

class sage.modular.abvar.abvar.**ModularAbelianVariety_modsym**(*modsym*, *lattice=None*, *newform_level=None*, *is_simple=None*, *isogeny_number=None*, *number=None*, *check=True*)

Bases: *ModularAbelianVariety_modsym_abstract*

Modular abelian variety that corresponds to a Hecke stable space of cuspidal modular symbols.

EXAMPLES: We create a modular abelian variety attached to a space of modular symbols.

```
sage: M = ModularSymbols(23).cuspidal_submodule()
sage: A = M.abelian_variety(); A
Abelian variety J0(23) of dimension 2
```

```
sage: from sage.all import *
sage: M = ModularSymbols(Integer(23)).cuspidal_submodule()
sage: A = M.abelian_variety(); A
Abelian variety J0(23) of dimension 2
```

brandt_module(*p*)

Return the Brandt module at p that corresponds to `self`. This is the factor of the vector space on the ideal class set in an order of level N in the quaternion algebra ramified at p and infinity.

INPUT:

- `p` – prime that exactly divides the level

OUTPUT: Brandt module space that corresponds to self

EXAMPLES:

```
sage: J0(43)[1].brandt_module(43)
Subspace of dimension 2 of Brandt module of dimension 4 of level 43 of weight␣
↪2 over Rational Field
sage: J0(43)[1].brandt_module(43).basis()
((1, 0, -1/2, -1/2), (0, 1, -1/2, -1/2))
sage: J0(43)[0].brandt_module(43).basis()
((0, 0, 1, -1),)
sage: J0(35)[0].brandt_module(5).basis()
((1, 0, -1, 0),)
sage: J0(35)[0].brandt_module(7).basis()
((1, -1, 1, -1),)
```

```
sage: from sage.all import *
sage: J0(Integer(43))[Integer(1)].brandt_module(Integer(43))
```

(continues on next page)

(continued from previous page)

```
Subspace of dimension 2 of Brandt module of dimension 4 of level 43 of weight␣
↪2 over Rational Field
>>> J0(Integer(43))[Integer(1)].brandt_module(Integer(43)).basis()
((1, 0, -1/2, -1/2), (0, 1, -1/2, -1/2))
>>> J0(Integer(43))[Integer(0)].brandt_module(Integer(43)).basis()
((0, 0, 1, -1),)
>>> J0(Integer(35))[Integer(0)].brandt_module(Integer(5)).basis()
((1, 0, -1, 0),)
>>> J0(Integer(35))[Integer(0)].brandt_module(Integer(7)).basis()
((1, -1, 1, -1),)
```

component_group_order(*p*)

Return the order of the component group of the special fiber at p of the Neron model of `self`.

Note

For bad primes, this is only implemented when the group is Γ_0 and p exactly divides the level.

Note

The input abelian variety must be simple.

ALGORITHM: See "Component Groups of Quotients of J0(N)" by Kohel and Stein. That paper is about optimal quotients; however, section 4.1 of Conrad-Stein "Component Groups of Purely Toric Quotients", one sees that the component group of an optimal quotient is the same as the component group of its dual (which is the subvariety).

INPUT:

- `p` – a prime number

OUTPUT: integer

EXAMPLES:

```
sage: A = J0(37)[1]
sage: A.component_group_order(37)
3
sage: A = J0(43)[1]
sage: A.component_group_order(37)
1
sage: A.component_group_order(43)
7
sage: A = J0(23)[0]
sage: A.component_group_order(23)
11
```

```
>>> from sage.all import *
>>> A = J0(Integer(37))[Integer(1)]
>>> A.component_group_order(Integer(37))
3
```

(continues on next page)

(continued from previous page)

```
>>> A = J0(Integer(43))[Integer(1)]
>>> A.component_group_order(Integer(37))
1
>>> A.component_group_order(Integer(43))
7
>>> A = J0(Integer(23))[Integer(0)]
>>> A.component_group_order(Integer(23))
11
```

tamagawa_number(*p*)

Return the Tamagawa number of this abelian variety at p.

NOTE: For bad primes, this is only implemented when the group if Gamma0 and p exactly divides the level and Atkin-Lehner acts diagonally on this abelian variety (e.g., if this variety is new and simple). See the self.component_group command for more information.

NOTE: the input abelian variety must be simple

In cases where this function doesn't work, consider using the self.tamagawa_number_bounds functions.

INPUT:

- `p` – a prime number

OUTPUT: integer

EXAMPLES:

```
sage: A = J0(37)[1]
sage: A.tamagawa_number(37)
3
sage: A = J0(43)[1]
sage: A.tamagawa_number(37)
1
sage: A.tamagawa_number(43)
7
sage: A = J0(23)[0]
sage: A.tamagawa_number(23)
11
```

```
>>> from sage.all import *
>>> A = J0(Integer(37))[Integer(1)]
>>> A.tamagawa_number(Integer(37))
3
>>> A = J0(Integer(43))[Integer(1)]
>>> A.tamagawa_number(Integer(37))
1
>>> A.tamagawa_number(Integer(43))
7
>>> A = J0(Integer(23))[Integer(0)]
>>> A.tamagawa_number(Integer(23))
11
```

tamagawa_number_bounds(*p*)

Return a divisor and multiple of the Tamagawa number of `self` at *p*.

NOTE: the input abelian variety must be simple.

INPUT:

- `p` – a prime number

OUTPUT:

- `div` – integer; divisor of Tamagawa number at p
- `mul` – integer; multiple of Tamagawa number at p
- `mul_primes` – tuple; in case `mul==0`, a list of all primes that can possibly divide the Tamagawa number at p

EXAMPLES:

```
sage: A = J0(63).new_subvariety()[1]; A
Simple abelian subvariety 63b(1,63) of dimension 2 of J0(63)
sage: A.tamagawa_number_bounds(7)
(3, 3, ())
sage: A.tamagawa_number_bounds(3)
(1, 0, (2, 3, 5))
```

```
>>> from sage.all import *
>>> A = J0(Integer(63)).new_subvariety()[Integer(1)]; A
Simple abelian subvariety 63b(1,63) of dimension 2 of J0(63)
>>> A.tamagawa_number_bounds(Integer(7))
(3, 3, ())
>>> A.tamagawa_number_bounds(Integer(3))
(1, 0, (2, 3, 5))
```

class sage.modular.abvar.abvar.**ModularAbelianVariety_modsym_abstract**(*groups*, *base_field*, *is_simple=None*, *newform_level=None*, *isogeny_number=None*, *number=None*, *check=True*)

Bases: *ModularAbelianVariety_abstract*

decomposition(*simple=True*, *bound=None*)

Decompose this modular abelian variety as a product of abelian subvarieties, up to isogeny.

INPUT:

- `simple` – boolean (default: `True`); if `True`, all factors are simple. If `False`, each factor returned is isogenous to a power of a simple and the simples in each factor are distinct.
- `bound` – integer (default: `None`); if given, only use Hecke operators up to this bound when decomposing. This can give wrong answers, so use with caution!

EXAMPLES:

```
sage: J = J0(33)
sage: J.decomposition()
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)]
sage: J1(17).decomposition()
```

(continues on next page)

(continued from previous page)

```
[Simple abelian subvariety 17aG1(1,17) of dimension 1 of J1(17),
 Simple abelian subvariety 17bG1(1,17) of dimension 4 of J1(17)]
```

```
>>> from sage.all import *
>>> J = J0(Integer(33))
>>> J.decomposition()
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)]
>>> J1(Integer(17)).decomposition()
[Simple abelian subvariety 17aG1(1,17) of dimension 1 of J1(17),
 Simple abelian subvariety 17bG1(1,17) of dimension 4 of J1(17)]
```

dimension()

Return the dimension of this modular abelian variety.

EXAMPLES:

```
sage: J0(37)[0].dimension()
1
sage: J0(43)[1].dimension()
2
sage: J1(17)[1].dimension()
4
```

```
>>> from sage.all import *
>>> J0(Integer(37))[Integer(0)].dimension()
1
>>> J0(Integer(43))[Integer(1)].dimension()
2
>>> J1(Integer(17))[Integer(1)].dimension()
4
```

group()

Return the congruence subgroup associated that this modular abelian variety is associated to.

EXAMPLES:

```
sage: J0(13).group()
Congruence Subgroup Gamma0(13)
sage: J1(997).group()
Congruence Subgroup Gamma1(997)
sage: JH(37,[3]).group()
Congruence Subgroup Gamma_H(37) with H generated by [3]
sage: J0(37)[1].groups()
(Congruence Subgroup Gamma0(37),)
```

```
>>> from sage.all import *
>>> J0(Integer(13)).group()
Congruence Subgroup Gamma0(13)
>>> J1(Integer(997)).group()
Congruence Subgroup Gamma1(997)
```

(continues on next page)

(continued from previous page)

```
>>> JH(Integer(37),[Integer(3)]).group()
Congruence Subgroup Gamma_H(37) with H generated by [3]
>>> J0(Integer(37))[Integer(1)].groups()
(Congruence Subgroup Gamma0(37),)
```

groups()

Return the tuple of groups associated to the modular symbols abelian variety. This is always a 1-tuple.

OUTPUT: tuple

EXAMPLES:

```
sage: A = ModularSymbols(33).cuspidal_submodule().abelian_variety(); A
Abelian variety J0(33) of dimension 3
sage: A.groups()
(Congruence Subgroup Gamma0(33),)
sage: type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_modsym_with_category'>
```

```
>>> from sage.all import *
>>> A = ModularSymbols(Integer(33)).cuspidal_submodule().abelian_variety(); A
Abelian variety J0(33) of dimension 3
>>> A.groups()
(Congruence Subgroup Gamma0(33),)
>>> type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_modsym_with_category'>
```

is_ambient()

Return `True` if this abelian variety attached to a modular symbols space is attached to the cuspidal subspace of the ambient modular symbols space.

OUTPUT: boolean

EXAMPLES:

```
sage: A = ModularSymbols(43).cuspidal_subspace().abelian_variety(); A
Abelian variety J0(43) of dimension 3
sage: A.is_ambient()
True
sage: type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_modsym_with_category'>
sage: A = ModularSymbols(43).cuspidal_subspace()[1].abelian_variety(); A
Abelian subvariety of dimension 2 of J0(43)
sage: A.is_ambient()
False
```

```
>>> from sage.all import *
>>> A = ModularSymbols(Integer(43)).cuspidal_subspace().abelian_variety(); A
Abelian variety J0(43) of dimension 3
>>> A.is_ambient()
True
>>> type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_modsym_with_category'>
```

(continues on next page)

(continued from previous page)

```
>>> A = ModularSymbols(Integer(43)).cuspidal_subspace()[Integer(1)].abelian_
↪variety(); A
Abelian subvariety of dimension 2 of J0(43)
>>> A.is_ambient()
False
```

is_subvariety(*other*)

Return `True` if `self` is a subvariety of `other`.

EXAMPLES:

```
sage: J = J0(37); J
Abelian variety J0(37) of dimension 2
sage: A = J[0]; A
Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37)
sage: A.is_subvariety(J)
True
sage: A.is_subvariety(J0(11))
False
```

```
>>> from sage.all import *
>>> J = J0(Integer(37)); J
Abelian variety J0(37) of dimension 2
>>> A = J[Integer(0)]; A
Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37)
>>> A.is_subvariety(J)
True
>>> A.is_subvariety(J0(Integer(11)))
False
```

There may be a way to map A into $J_0(74)$, but A is not equipped with any special structure of an embedding.

```
sage: A.is_subvariety(J0(74))
False
```

```
>>> from sage.all import *
>>> A.is_subvariety(J0(Integer(74)))
False
```

Some ambient examples:

```
sage: J = J0(37)
sage: J.is_subvariety(J)
True
sage: J.is_subvariety(25)
False
```

```
>>> from sage.all import *
>>> J = J0(Integer(37))
>>> J.is_subvariety(J)
True
```

(continues on next page)

(continued from previous page)

```
>>> J.is_subvariety(Integer(25))
False
```

More examples:

```
sage: A = J0(42); D = A.decomposition(); D
[Simple abelian subvariety 14a(1,42) of dimension 1 of J0(42),
 Simple abelian subvariety 14a(3,42) of dimension 1 of J0(42),
 Simple abelian subvariety 21a(1,42) of dimension 1 of J0(42),
 Simple abelian subvariety 21a(2,42) of dimension 1 of J0(42),
 Simple abelian subvariety 42a(1,42) of dimension 1 of J0(42)]
sage: D[0].is_subvariety(A)
True
sage: D[1].is_subvariety(D[0] + D[1])
True
sage: D[2].is_subvariety(D[0] + D[1])
False
```

```
>>> from sage.all import *
>>> A = J0(Integer(42)); D = A.decomposition(); D
[Simple abelian subvariety 14a(1,42) of dimension 1 of J0(42),
 Simple abelian subvariety 14a(3,42) of dimension 1 of J0(42),
 Simple abelian subvariety 21a(1,42) of dimension 1 of J0(42),
 Simple abelian subvariety 21a(2,42) of dimension 1 of J0(42),
 Simple abelian subvariety 42a(1,42) of dimension 1 of J0(42)]
>>> D[Integer(0)].is_subvariety(A)
True
>>> D[Integer(1)].is_subvariety(D[Integer(0)] + D[Integer(1)])
True
>>> D[Integer(2)].is_subvariety(D[Integer(0)] + D[Integer(1)])
False
```

lattice()

Return the lattice defining this modular abelian variety.

OUTPUT:

A free **Z**-module embedded in an ambient **Q**-vector space.

EXAMPLES:

```
sage: A = ModularSymbols(33).cuspidal_submodule()[0].abelian_variety(); A
Abelian subvariety of dimension 1 of J0(33)
sage: A.lattice()
Free module of degree 6 and rank 2 over Integer Ring
User basis matrix:
[ 1  0  0 -1  0  0]
[ 0  0  1  0  1 -1]
sage: type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_modsym_with_category'>
```

```
>>> from sage.all import *
>>> A = ModularSymbols(Integer(33)).cuspidal_submodule()[Integer(0)].abelian_
```

(continues on next page)

(continued from previous page)

```
↪variety(); A
Abelian subvariety of dimension 1 of J0(33)
>>> A.lattice()
Free module of degree 6 and rank 2 over Integer Ring
User basis matrix:
[ 1  0  0 -1  0  0]
[ 0  0  1  0  1 -1]
>>> type(A)
<class 'sage.modular.abvar.abvar.ModularAbelianVariety_modsym_with_category'>
```

modular_symbols(*sign=0*)

Return space of modular symbols (with given sign) associated to this modular abelian variety, if it can be found by cutting down using Hecke operators. Otherwise raise a `RuntimeError` exception.

EXAMPLES:

```
sage: A = J0(37)
sage: A.modular_symbols()
Modular Symbols subspace of dimension 4 of Modular Symbols space of dimension␣
↪5 for Gamma_0(37) of weight 2 with sign 0 over Rational Field
sage: A.modular_symbols(1)
Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension␣
↪3 for Gamma_0(37) of weight 2 with sign 1 over Rational Field
```

```
>>> from sage.all import *
>>> A = J0(Integer(37))
>>> A.modular_symbols()
Modular Symbols subspace of dimension 4 of Modular Symbols space of dimension␣
↪5 for Gamma_0(37) of weight 2 with sign 0 over Rational Field
>>> A.modular_symbols(Integer(1))
Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension␣
↪3 for Gamma_0(37) of weight 2 with sign 1 over Rational Field
```

More examples:

```
sage: J0(11).modular_symbols()
Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension␣
↪3 for Gamma_0(11) of weight 2 with sign 0 over Rational Field
sage: J0(11).modular_symbols(sign=1)
Modular Symbols subspace of dimension 1 of Modular Symbols space of dimension␣
↪2 for Gamma_0(11) of weight 2 with sign 1 over Rational Field
sage: J0(11).modular_symbols(sign=0)
Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension␣
↪3 for Gamma_0(11) of weight 2 with sign 0 over Rational Field
sage: J0(11).modular_symbols(sign=-1)
Modular Symbols space of dimension 1 for Gamma_0(11) of weight 2 with sign -1␣
↪over Rational Field
```

```
>>> from sage.all import *
>>> J0(Integer(11)).modular_symbols()
Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension␣
↪3 for Gamma_0(11) of weight 2 with sign 0 over Rational Field
```

(continues on next page)

(continued from previous page)

```
>>> J0(Integer(11)).modular_symbols(sign=Integer(1))
Modular Symbols subspace of dimension 1 of Modular Symbols space of dimension␣
↪2 for Gamma_0(11) of weight 2 with sign 1 over Rational Field
>>> J0(Integer(11)).modular_symbols(sign=Integer(0))
Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension␣
↪3 for Gamma_0(11) of weight 2 with sign 0 over Rational Field
>>> J0(Integer(11)).modular_symbols(sign=-Integer(1))
Modular Symbols space of dimension 1 for Gamma_0(11) of weight 2 with sign -1␣
↪over Rational Field
```

Even more examples:

```
sage: A = J0(33)[1]; A
Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)
sage: A.modular_symbols()
Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension␣
↪9 for Gamma_0(33) of weight 2 with sign 0 over Rational Field
```

```
>>> from sage.all import *
>>> A = J0(Integer(33))[Integer(1)]; A
Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33)
>>> A.modular_symbols()
Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension␣
↪9 for Gamma_0(33) of weight 2 with sign 0 over Rational Field
```

It is not always possible to determine the sign subspaces:

```
sage: A.modular_symbols(1)
Traceback (most recent call last):
...
RuntimeError: unable to determine sign (=1) space of modular symbols
```

```
>>> from sage.all import *
>>> A.modular_symbols(Integer(1))
Traceback (most recent call last):
...
RuntimeError: unable to determine sign (=1) space of modular symbols
```

```
sage: A.modular_symbols(-1)
Traceback (most recent call last):
...
RuntimeError: unable to determine sign (=-1) space of modular symbols
```

```
>>> from sage.all import *
>>> A.modular_symbols(-Integer(1))
Traceback (most recent call last):
...
RuntimeError: unable to determine sign (=-1) space of modular symbols
```

new_subvariety(*p=None*)

Return the new or p-new subvariety of `self`.

INPUT:

- `self` – a modular abelian variety
- `p` – prime number or None (default); if p is a prime, return the p-new subvariety. Otherwise return the full new subvariety.

EXAMPLES:

```
sage: J0(34).new_subvariety()
Abelian subvariety of dimension 1 of J0(34)
sage: J0(100).new_subvariety()
Abelian subvariety of dimension 1 of J0(100)
sage: J1(13).new_subvariety()
Abelian variety J1(13) of dimension 2
```

```
>>> from sage.all import *
>>> J0(Integer(34)).new_subvariety()
Abelian subvariety of dimension 1 of J0(34)
>>> J0(Integer(100)).new_subvariety()
Abelian subvariety of dimension 1 of J0(100)
>>> J1(Integer(13)).new_subvariety()
Abelian variety J1(13) of dimension 2
```

old_subvariety(*p=None*)

Return the old or p-old abelian variety of `self`.

INPUT:

- `self` – a modular abelian variety
- `p` – prime number or `None` (default); if p is a prime, return the p-old subvariety. Otherwise return the full old subvariety.

EXAMPLES:

```
sage: J0(33).old_subvariety()
Abelian subvariety of dimension 2 of J0(33)
sage: J0(100).old_subvariety()
Abelian subvariety of dimension 6 of J0(100)
sage: J1(13).old_subvariety()
Abelian subvariety of dimension 0 of J1(13)
```

```
>>> from sage.all import *
>>> J0(Integer(33)).old_subvariety()
Abelian subvariety of dimension 2 of J0(33)
>>> J0(Integer(100)).old_subvariety()
Abelian subvariety of dimension 6 of J0(100)
>>> J1(Integer(13)).old_subvariety()
Abelian subvariety of dimension 0 of J1(13)
```

sage.modular.abvar.abvar.**factor_modsym_space_new_factors**(M)

Return the factorizations of all the new subspaces of M.

INPUT:

- `M` – ambient modular symbols space

OUTPUT: list of decompositions corresponding to each new space

EXAMPLES:

```
sage: M = ModularSymbols(33)
sage: sage.modular.abvar.abvar.factor_modsym_space_new_factors(M)
[[Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 3␣
↪for Gamma_0(11) of weight 2 with sign 0 over Rational Field],
 [Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field]]
```

```
>>> from sage.all import *
>>> M = ModularSymbols(Integer(33))
>>> sage.modular.abvar.abvar.factor_modsym_space_new_factors(M)
[[Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 3␣
↪for Gamma_0(11) of weight 2 with sign 0 over Rational Field],
 [Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field]]
```

sage.modular.abvar.abvar.**factor_new_space**(*M*)

Given a new space M of modular symbols, return the decomposition into simple of M under the Hecke operators.

INPUT:

- `M` – modular symbols space

OUTPUT: list of factors

EXAMPLES:

```
sage: M = ModularSymbols(37).cuspidal_subspace()
sage: sage.modular.abvar.abvar.factor_new_space(M)
[Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 5␣
↪for Gamma_0(37) of weight 2 with sign 0 over Rational Field,
 Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 5␣
↪for Gamma_0(37) of weight 2 with sign 0 over Rational Field]
```

```
>>> from sage.all import *
>>> M = ModularSymbols(Integer(37)).cuspidal_subspace()
>>> sage.modular.abvar.abvar.factor_new_space(M)
[Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 5␣
↪for Gamma_0(37) of weight 2 with sign 0 over Rational Field,
 Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 5␣
↪for Gamma_0(37) of weight 2 with sign 0 over Rational Field]
```

sage.modular.abvar.abvar.**is_ModularAbelianVariety**(*x*)

Return `True` if x is a modular abelian variety.

INPUT:

- `x` – object

EXAMPLES:

```
sage: from sage.modular.abvar.abvar import is_ModularAbelianVariety
sage: is_ModularAbelianVariety(5)
doctest:warning...
DeprecationWarning: The function is_ModularAbelianVariety is deprecated; use
↪'isinstance(..., ModularAbelianVariety_abstract)' instead.
```

(continues on next page)

(continued from previous page)

```
See https://github.com/sagemath/sage/issues/38035 for details.
False
sage: is_ModularAbelianVariety(J0(37))
True
```

```
>>> from sage.all import *
>>> from sage.modular.abvar.abvar import is_ModularAbelianVariety
>>> is_ModularAbelianVariety(Integer(5))
doctest:warning...
DeprecationWarning: The function is_ModularAbelianVariety is deprecated; use
↪'isinstance(..., ModularAbelianVariety_abstract)' instead.
See https://github.com/sagemath/sage/issues/38035 for details.
False
>>> is_ModularAbelianVariety(J0(Integer(37)))
True
```

Returning `True` is a statement about the data type not whether or not some abelian variety is modular:

```
sage: is_ModularAbelianVariety(EllipticCurve('37a'))
False
```

```
>>> from sage.all import *
>>> is_ModularAbelianVariety(EllipticCurve('37a'))
False
```

sage.modular.abvar.abvar.**modsym_lattices**(*M*, *factors*)

Append lattice information to the output of simple_factorization_of_modsym_space.

INPUT:

- `M` – modular symbols spaces
- `factors` – sequence (simple_factorization_of_modsym_space)

OUTPUT: sequence with more information for each factor (the lattice)

EXAMPLES:

```
sage: M = ModularSymbols(33)
sage: factors = sage.modular.abvar.abvar.simple_factorization_of_modsym_space(M,␣
↪simple=False)
sage: sage.modular.abvar.abvar.modsym_lattices(M, factors)
[(11,
  0,
  None,
  Modular Symbols subspace of dimension 4 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field,
  Free module of degree 6 and rank 4 over Integer Ring
  Echelon basis matrix:
  [ 1  0  0  0 -1  2]
  [ 0  1  0  0 -1  1]
  [ 0  0  1  0 -2  2]
  [ 0  0  0  1 -1 -1]),
 (33,
```

(continues on next page)

(continued from previous page)

```
  0,
  None,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field,
  Free module of degree 6 and rank 2 over Integer Ring
  Echelon basis matrix:
  [ 1  0  0 -1  0  0]
  [ 0  0  1  0  1 -1])]
```

```
>>> from sage.all import *
>>> M = ModularSymbols(Integer(33))
>>> factors = sage.modular.abvar.abvar.simple_factorization_of_modsym_space(M,␣
↪simple=False)
>>> sage.modular.abvar.abvar.modsym_lattices(M, factors)
[(11,
  0,
  None,
  Modular Symbols subspace of dimension 4 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field,
  Free module of degree 6 and rank 4 over Integer Ring
  Echelon basis matrix:
  [ 1  0  0  0 -1  2]
  [ 0  1  0  0 -1  1]
  [ 0  0  1  0 -2  2]
  [ 0  0  0  1 -1 -1]),
 (33,
  0,
  None,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field,
  Free module of degree 6 and rank 2 over Integer Ring
  Echelon basis matrix:
  [ 1  0  0 -1  0  0]
  [ 0  0  1  0  1 -1])]
```

sage.modular.abvar.abvar.**random_hecke_operator**(*M*, *t=None*, *p=2*)

Return a random Hecke operator acting on M, got by adding to t a random multiple of T_p

INPUT:

- `M` – modular symbols space
- `t` – `None` or a Hecke operator
- `p` – a prime

OUTPUT: Hecke operator prime

EXAMPLES:

```
sage: M = ModularSymbols(11).cuspidal_subspace()
sage: t, p = sage.modular.abvar.abvar.random_hecke_operator(M)
sage: p
3
sage: t, p = sage.modular.abvar.abvar.random_hecke_operator(M, t, p)
```

(continues on next page)

(continued from previous page)

```
sage: p
5
```

```
>>> from sage.all import *
>>> M = ModularSymbols(Integer(11)).cuspidal_subspace()
>>> t, p = sage.modular.abvar.abvar.random_hecke_operator(M)
>>> p
3
>>> t, p = sage.modular.abvar.abvar.random_hecke_operator(M, t, p)
>>> p
5
```

sage.modular.abvar.abvar.**simple_factorization_of_modsym_space**(*M*, *simple=True*)

Return the canonical factorization of M into (simple) subspaces.

INPUT:

- `M` – ambient modular symbols space
- `simple` – boolean (default: `True`); if set to `False`, isogenous simple factors are grouped together

OUTPUT: sequence

EXAMPLES:

```
sage: M = ModularSymbols(33)
sage: sage.modular.abvar.abvar.simple_factorization_of_modsym_space(M)
[(11,
  0,
  1,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field),
 (11,
  0,
  3,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field),
 (33,
  0,
  1,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field)]
sage: sage.modular.abvar.abvar.simple_factorization_of_modsym_space(M,␣
↪simple=False)
[(11,
  0,
  None,
  Modular Symbols subspace of dimension 4 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field),
 (33,
  0,
  None,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field)]
```

```
>>> from sage.all import *
>>> M = ModularSymbols(Integer(33))
>>> sage.modular.abvar.abvar.simple_factorization_of_modsym_space(M)
[(11,
  0,
  1,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field),
 (11,
  0,
  3,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field),
 (33,
  0,
  1,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field)]
>>> sage.modular.abvar.abvar.simple_factorization_of_modsym_space(M, simple=False)
[(11,
  0,
  None,
  Modular Symbols subspace of dimension 4 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field),
 (33,
  0,
  None,
  Modular Symbols subspace of dimension 2 of Modular Symbols space of dimension 9␣
↪for Gamma_0(33) of weight 2 with sign 0 over Rational Field)]
```

sage.modular.abvar.abvar.**sqrt_poly**(f)

Return the square root of the polynomial f.

Note

At some point something like this should be a member of the polynomial class. For now this is just used internally by some charpoly functions above.

EXAMPLES:

```
sage: R.<x> = QQ[]
sage: f = (x-1)*(x+2)*(x^2 + 1/3*x + 5)
sage: f
x^4 + 4/3*x^3 + 10/3*x^2 + 13/3*x - 10
sage: sage.modular.abvar.abvar.sqrt_poly(f^2)
x^4 + 4/3*x^3 + 10/3*x^2 + 13/3*x - 10
sage: sage.modular.abvar.abvar.sqrt_poly(f)
Traceback (most recent call last):
...
ValueError: f must be a perfect square
sage: sage.modular.abvar.abvar.sqrt_poly(2*f^2)
Traceback (most recent call last):
```

(continues on next page)

(continued from previous page)

```
...
ValueError: f must be monic
```

```
>>> from sage.all import *
>>> R = QQ['x']; (x,) = R._first_ngens(1)
>>> f = (x-Integer(1))*(x+Integer(2))*(x**Integer(2) + Integer(1)/Integer(3)*x +␣
↪Integer(5))
>>> f
x^4 + 4/3*x^3 + 10/3*x^2 + 13/3*x - 10
>>> sage.modular.abvar.abvar.sqrt_poly(f**Integer(2))
x^4 + 4/3*x^3 + 10/3*x^2 + 13/3*x - 10
>>> sage.modular.abvar.abvar.sqrt_poly(f)
Traceback (most recent call last):
...
ValueError: f must be a perfect square
>>> sage.modular.abvar.abvar.sqrt_poly(Integer(2)*f**Integer(2))
Traceback (most recent call last):
...
ValueError: f must be monic
```

CHAPTER

THREE

AMBIENT JACOBIAN ABELIAN VARIETY

sage.modular.abvar.abvar_ambient_jacobian.**ModAbVar_ambient_jacobian**(*group*)

Return the ambient Jacobian attached to a given congruence subgroup.

The result is cached using a weakref. This function is called internally by modular abelian variety constructors.

INPUT:

- `group` – a congruence subgroup

OUTPUT: a modular abelian variety attached

EXAMPLES:

```
sage: import sage.modular.abvar.abvar_ambient_jacobian as abvar_ambient_jacobian
sage: A = abvar_ambient_jacobian.ModAbVar_ambient_jacobian(Gamma0(11))
sage: A
Abelian variety J0(11) of dimension 1
sage: B = abvar_ambient_jacobian.ModAbVar_ambient_jacobian(Gamma0(11))
sage: A is B
True
```

```
>>> from sage.all import *
>>> import sage.modular.abvar.abvar_ambient_jacobian as abvar_ambient_jacobian
>>> A = abvar_ambient_jacobian.ModAbVar_ambient_jacobian(Gamma0(Integer(11)))
>>> A
Abelian variety J0(11) of dimension 1
>>> B = abvar_ambient_jacobian.ModAbVar_ambient_jacobian(Gamma0(Integer(11)))
>>> A is B
True
```

You can get access to and/or clear the cache as follows:

```
sage: abvar_ambient_jacobian._cache = {}
sage: B = abvar_ambient_jacobian.ModAbVar_ambient_jacobian(Gamma0(11))
sage: A is B
False
```

```
>>> from sage.all import *
>>> abvar_ambient_jacobian._cache = {}
>>> B = abvar_ambient_jacobian.ModAbVar_ambient_jacobian(Gamma0(Integer(11)))
>>> A is B
False
```

class sage.modular.abvar.abvar_ambient_jacobian.**ModAbVar_ambient_jacobian_class**(*group*)

Bases: *ModularAbelianVariety_modsym_abstract*

An ambient Jacobian modular abelian variety attached to a congruence subgroup.

ambient_variety()

Return the ambient modular abelian variety that contains `self`. Since `self` is a Jacobian modular abelian variety, this is just `self`.

OUTPUT: abelian variety

EXAMPLES:

```
sage: A = J0(17)
sage: A.ambient_variety()
Abelian variety J0(17) of dimension 1
sage: A is A.ambient_variety()
True
```

```
>>> from sage.all import *
>>> A = J0(Integer(17))
>>> A.ambient_variety()
Abelian variety J0(17) of dimension 1
>>> A is A.ambient_variety()
True
```

decomposition(*simple=True*, *bound=None*)

Decompose this ambient Jacobian as a product of abelian subvarieties, up to isogeny.

EXAMPLES:

```
sage: J0(33).decomposition(simple=False)
[Abelian subvariety of dimension 2 of J0(33),
 Abelian subvariety of dimension 1 of J0(33)]
sage: J0(33).decomposition(simple=False)[1].is_simple()
True
sage: J0(33).decomposition(simple=False)[0].is_simple()
False
sage: J0(33).decomposition(simple=False)
[Abelian subvariety of dimension 2 of J0(33),
 Simple abelian subvariety 33a(None,33) of dimension 1 of J0(33)]
sage: J0(33).decomposition(simple=True)
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)]
```

```
>>> from sage.all import *
>>> J0(Integer(33)).decomposition(simple=False)
[Abelian subvariety of dimension 2 of J0(33),
 Abelian subvariety of dimension 1 of J0(33)]
>>> J0(Integer(33)).decomposition(simple=False)[Integer(1)].is_simple()
True
>>> J0(Integer(33)).decomposition(simple=False)[Integer(0)].is_simple()
False
>>> J0(Integer(33)).decomposition(simple=False)
```

(continues on next page)

(continued from previous page)

```
[Abelian subvariety of dimension 2 of J0(33),
 Simple abelian subvariety 33a(None,33) of dimension 1 of J0(33)]
>>> J0(Integer(33)).decomposition(simple=True)
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)]
```

degeneracy_map(*level*, *t=1*, *check=True*)

Return the t-th degeneracy map from `self` to `J(level)`. Here t must be a divisor of either `level/self.level()` or `self.level()/level`.

INPUT:

- `level` – integer (multiple or divisor of level of `self`)
- `t` – divisor of quotient of level of `self` and level
- `check` – boolean (default: `True`); if `True` do some checks on the input

OUTPUT: a morphism

EXAMPLES:

```
sage: J0(11).degeneracy_map(33)
Degeneracy map from Abelian variety J0(11) of dimension 1 to Abelian variety␣
↪J0(33) of dimension 3 defined by [1]
sage: J0(11).degeneracy_map(33).matrix()
[ 0 -3  2  1 -2  0]
[ 1 -2  0  1  0 -1]
sage: J0(11).degeneracy_map(33,3).matrix()
[-1  0  0  0  1 -2]
[-1 -1  1 -1  1  0]
sage: J0(33).degeneracy_map(11,1).matrix()
[ 0  1]
[ 0 -1]
[ 1 -1]
[ 0  1]
[-1  1]
[ 0  0]
sage: J0(11).degeneracy_map(33,1).matrix() * J0(33).degeneracy_map(11,1).
↪matrix()
[4 0]
[0 4]
```

```
>>> from sage.all import *
>>> J0(Integer(11)).degeneracy_map(Integer(33))
Degeneracy map from Abelian variety J0(11) of dimension 1 to Abelian variety␣
↪J0(33) of dimension 3 defined by [1]
>>> J0(Integer(11)).degeneracy_map(Integer(33)).matrix()
[ 0 -3  2  1 -2  0]
[ 1 -2  0  1  0 -1]
>>> J0(Integer(11)).degeneracy_map(Integer(33),Integer(3)).matrix()
[-1  0  0  0  1 -2]
[-1 -1  1 -1  1  0]
>>> J0(Integer(33)).degeneracy_map(Integer(11),Integer(1)).matrix()
```

(continues on next page)

(continued from previous page)

```
[ 0  1]
[ 0 -1]
[ 1 -1]
[ 0  1]
[-1  1]
[ 0  0]
>>> J0(Integer(11)).degeneracy_map(Integer(33),Integer(1)).matrix() *␣
↪J0(Integer(33)).degeneracy_map(Integer(11),Integer(1)).matrix()
[4 0]
[0 4]
```

dimension()

Return the dimension of this modular abelian variety.

EXAMPLES:

```
sage: J0(2007).dimension()
221
sage: J1(13).dimension()
2
sage: J1(997).dimension()
40920
sage: J0(389).dimension()
32
sage: JH(389,[4]).dimension()
64
sage: J1(389).dimension()
6112
```

```
>>> from sage.all import *
>>> J0(Integer(2007)).dimension()
221
>>> J1(Integer(13)).dimension()
2
>>> J1(Integer(997)).dimension()
40920
>>> J0(Integer(389)).dimension()
32
>>> JH(Integer(389),[Integer(4)]).dimension()
64
>>> J1(Integer(389)).dimension()
6112
```

group()

Return the group that this Jacobian modular abelian variety is attached to.

EXAMPLES:

```
sage: J1(37).group()
Congruence Subgroup Gamma1(37)
sage: J0(5077).group()
Congruence Subgroup Gamma0(5077)
sage: J = GammaH(11,[3]).modular_abelian_variety(); J
```

(continues on next page)

(continued from previous page)

```
Abelian variety JH(11,[3]) of dimension 1
sage: J.group()
Congruence Subgroup Gamma_H(11) with H generated by [3]
```

```
>>> from sage.all import *
>>> J1(Integer(37)).group()
Congruence Subgroup Gamma1(37)
>>> J0(Integer(5077)).group()
Congruence Subgroup Gamma0(5077)
>>> J = GammaH(Integer(11),[Integer(3)]).modular_abelian_variety(); J
Abelian variety JH(11,[3]) of dimension 1
>>> J.group()
Congruence Subgroup Gamma_H(11) with H generated by [3]
```

groups()

Return the tuple of congruence subgroups attached to this ambient Jacobian. This is always a tuple of length 1.

OUTPUT: tuple

EXAMPLES:

```
sage: J0(37).groups()
(Congruence Subgroup Gamma0(37),)
```

```
>>> from sage.all import *
>>> J0(Integer(37)).groups()
(Congruence Subgroup Gamma0(37),)
```

newform_decomposition(*names=None*)

Return the newforms of the simple subvarieties in the decomposition of `self` as a product of simple subvarieties, up to isogeny.

OUTPUT: an array of newforms

EXAMPLES:

```
sage: J0(81).newform_decomposition('a')
[q - 2*q^4 + O(q^6), q - 2*q^4 + O(q^6), q + a0*q^2 + q^4 - a0*q^5 + O(q^6)]

sage: J1(19).newform_decomposition('a')
[q - 2*q^3 - 2*q^4 + 3*q^5 + O(q^6),
 q + a1*q^2 + (-1/9*a1^5 - 1/3*a1^4 - 1/3*a1^3 + 1/3*a1^2 - a1 - 1)*q^3 + (4/
↪9*a1^5 + 2*a1^4 + 14/3*a1^3 + 17/3*a1^2 + 6*a1 + 2)*q^4 + (-2/3*a1^5 - 11/
↪3*a1^4 - 10*a1^3 - 14*a1^2 - 15*a1 - 9)*q^5 + O(q^6)]
```

```
>>> from sage.all import *
>>> J0(Integer(81)).newform_decomposition('a')
[q - 2*q^4 + O(q^6), q - 2*q^4 + O(q^6), q + a0*q^2 + q^4 - a0*q^5 + O(q^6)]

>>> J1(Integer(19)).newform_decomposition('a')
[q - 2*q^3 - 2*q^4 + 3*q^5 + O(q^6),
 q + a1*q^2 + (-1/9*a1^5 - 1/3*a1^4 - 1/3*a1^3 + 1/3*a1^2 - a1 - 1)*q^3 + (4/
```

(continues on next page)

(continued from previous page)

```
↪9*a1^5 + 2*a1^4 + 14/3*a1^3 + 17/3*a1^2 + 6*a1 + 2)*q^4 + (-2/3*a1^5 - 11/
↪3*a1^4 - 10*a1^3 - 14*a1^2 - 15*a1 - 9)*q^5 + O(q^6)]
```

CHAPTER

FOUR

FINITE SUBGROUPS OF MODULAR ABELIAN VARIETIES

Sage can compute with fairly general finite subgroups of modular abelian varieties. Elements of finite order are represented by equivalence classes of elements in $H_1(A, \mathbf{Q})$ modulo $H_1(A, \mathbf{Z})$. A finite subgroup can be defined by giving generators and via various other constructions. Given a finite subgroup, one can compute generators, as well as the structure as an abstract group. Arithmetic on subgroups is also supported, including adding two subgroups together, checking inclusion, etc.

TODO: Intersection, action of Hecke operators.

AUTHORS:

- William Stein (2007-03)

EXAMPLES:

```
sage: J = J0(33)
sage: C = J.cuspidal_subgroup()
sage: C
Finite subgroup with invariants [10, 10] over QQ of Abelian variety J0(33) of␣
↪dimension 3
sage: C.order()
100
sage: C.gens()
[[(1/10, 0, 1/10, 1/10, 1/10, 3/10)], [(0, 1/5, 1/10, 0, 1/10, 9/10)], [(0, 0, 1/2, 0,
↪ 1/2, 1/2)]]
sage: C.0 + C.1
[(1/10, 1/5, 1/5, 1/10, 1/5, 6/5)]
sage: 10*(C.0 + C.1)
[(0, 0, 0, 0, 0, 0)]
sage: G = C.subgroup([C.0 + C.1]); G
Finite subgroup with invariants [10] over QQbar of Abelian variety J0(33) of␣
↪dimension 3
sage: G.gens()
[[(1/10, 1/5, 1/5, 1/10, 1/5, 1/5)]]
sage: G.order()
10
sage: G <= C
True
sage: G >= C
False
```

```
>>> from sage.all import *
>>> J = J0(Integer(33))
```

(continues on next page)

(continued from previous page)

```
>>> C = J.cuspidal_subgroup()
>>> C
Finite subgroup with invariants [10, 10] over QQ of Abelian variety J0(33) of␣
↪dimension 3
>>> C.order()
100
>>> C.gens()
[[(1/10, 0, 1/10, 1/10, 1/10, 3/10)], [(0, 1/5, 1/10, 0, 1/10, 9/10)], [(0, 0, 1/2, 0,
↪ 1/2, 1/2)]]
>>> C.gen(0) + C.gen(1)
[(1/10, 1/5, 1/5, 1/10, 1/5, 6/5)]
>>> Integer(10)*(C.gen(0) + C.gen(1))
[(0, 0, 0, 0, 0, 0)]
>>> G = C.subgroup([C.gen(0) + C.gen(1)]); G
Finite subgroup with invariants [10] over QQbar of Abelian variety J0(33) of␣
↪dimension 3
>>> G.gens()
[[(1/10, 1/5, 1/5, 1/10, 1/5, 1/5)]]
>>> G.order()
10
>>> G <= C
True
>>> G >= C
False
```

We make a table of the order of the cuspidal subgroup for the first few levels:

```
sage: for N in range(11,40):
....:     print("{} {}".format(N, J0(N).cuspidal_subgroup().order()))
11 5
12 1
13 1
14 6
15 8
16 1
17 4
18 1
19 3
20 6
21 8
22 25
23 11
24 8
25 1
26 21
27 9
28 36
29 7
30 192
31 5
32 8
33 100
```

(continues on next page)

(continued from previous page)

```
34 48
35 48
36 12
37 3
38 135
39 56
```

```
>>> from sage.all import *
>>> for N in range(Integer(11),Integer(40)):
...     print("{} {}".format(N, J0(N).cuspidal_subgroup().order()))
11 5
12 1
13 1
14 6
15 8
16 1
17 4
18 1
19 3
20 6
21 8
22 25
23 11
24 8
25 1
26 21
27 9
28 36
29 7
30 192
31 5
32 8
33 100
34 48
35 48
36 12
37 3
38 135
39 56
```

class sage.modular.abvar.finite_subgroup.**FiniteSubgroup**(*abvar*, *field_of_definition=Rational Field*)

Bases: Module

A finite subgroup of a modular abelian variety.

INPUT:

- abvar – a modular abelian variety
- field_of_definition – a field over which this group is defined

EXAMPLES:

This is an abstract base class, so there are no instances of this class itself:

```
sage: A = J0(37)
sage: G = A.torsion_subgroup(3); G
Finite subgroup with invariants [3, 3, 3, 3] over QQ of Abelian variety J0(37) of␣
↪dimension 2
sage: type(G)
<class 'sage.modular.abvar.finite_subgroup.FiniteSubgroup_lattice_with_category'>
sage: from sage.modular.abvar.finite_subgroup import FiniteSubgroup
sage: isinstance(G, FiniteSubgroup)
True
```

```
>>> from sage.all import *
>>> A = J0(Integer(37))
>>> G = A.torsion_subgroup(Integer(3)); G
Finite subgroup with invariants [3, 3, 3, 3] over QQ of Abelian variety J0(37) of␣
↪dimension 2
>>> type(G)
<class 'sage.modular.abvar.finite_subgroup.FiniteSubgroup_lattice_with_category'>
>>> from sage.modular.abvar.finite_subgroup import FiniteSubgroup
>>> isinstance(G, FiniteSubgroup)
True
```

Element

alias of `TorsionPoint`

abelian_variety()

Return the abelian variety that this is a finite subgroup of.

EXAMPLES:

```
sage: J = J0(42)
sage: G = J.rational_torsion_subgroup(); G
Torsion subgroup of Abelian variety J0(42) of dimension 5
sage: G.abelian_variety()
Abelian variety J0(42) of dimension 5
```

```
>>> from sage.all import *
>>> J = J0(Integer(42))
>>> G = J.rational_torsion_subgroup(); G
Torsion subgroup of Abelian variety J0(42) of dimension 5
>>> G.abelian_variety()
Abelian variety J0(42) of dimension 5
```

exponent()

Return the exponent of this finite abelian group.

OUTPUT: integer

EXAMPLES:

```
sage: t = J0(33).hecke_operator(7)
sage: G = t.kernel()[0]; G
Finite subgroup with invariants [2, 2, 2, 2, 4, 4] over QQ of
 Abelian variety J0(33) of dimension 3
```

(continues on next page)

(continued from previous page)

```
sage: G.exponent()
4
```

```
>>> from sage.all import *
>>> t = J0(Integer(33)).hecke_operator(Integer(7))
>>> G = t.kernel()[Integer(0)]; G
Finite subgroup with invariants [2, 2, 2, 2, 4, 4] over QQ of
 Abelian variety J0(33) of dimension 3
>>> G.exponent()
4
```

field_of_definition()

Return the field over which this finite modular abelian variety subgroup is defined. This is a field over which this subgroup is defined.

EXAMPLES:

```
sage: J = J0(42)
sage: G = J.rational_torsion_subgroup(); G
Torsion subgroup of Abelian variety J0(42) of dimension 5
sage: G.field_of_definition()
Rational Field
```

```
>>> from sage.all import *
>>> J = J0(Integer(42))
>>> G = J.rational_torsion_subgroup(); G
Torsion subgroup of Abelian variety J0(42) of dimension 5
>>> G.field_of_definition()
Rational Field
```

gen(n)

Return n-th generator of `self`.

EXAMPLES:

```
sage: J = J0(23)
sage: C = J.torsion_subgroup(3)
sage: C.gens()
[[(1/3, 0, 0, 0)], [(0, 1/3, 0, 0)], [(0, 0, 1/3, 0)], [(0, 0, 0, 1/3)]]
sage: C.gen(0)
[(1/3, 0, 0, 0)]
sage: C.gen(3)
[(0, 0, 0, 1/3)]
sage: C.gen(4)
Traceback (most recent call last):
...
IndexError: list index out of range
```

```
>>> from sage.all import *
>>> J = J0(Integer(23))
>>> C = J.torsion_subgroup(Integer(3))
>>> C.gens()
```

(continues on next page)

(continued from previous page)

```
[[(1/3, 0, 0, 0)], [(0, 1/3, 0, 0)], [(0, 0, 1/3, 0)], [(0, 0, 0, 1/3)]]
>>> C.gen(Integer(0))
[(1/3, 0, 0, 0)]
>>> C.gen(Integer(3))
[(0, 0, 0, 1/3)]
>>> C.gen(Integer(4))
Traceback (most recent call last):
...
IndexError: list index out of range
```

Negative indices wrap around:

```
sage: C.gen(-1)
[(0, 0, 0, 1/3)]
```

```
>>> from sage.all import *
>>> C.gen(-Integer(1))
[(0, 0, 0, 1/3)]
```

gens()

Return generators for this finite subgroup.

EXAMPLES:

We list generators for several cuspidal subgroups:

```
sage: J0(11).cuspidal_subgroup().gens()
[[(0, 1/5)]]
sage: J0(37).cuspidal_subgroup().gens()
[[(0, 0, 0, 1/3)]]
sage: J0(43).cuspidal_subgroup().gens()
[[(0, 1/7, 0, 6/7, 0, 5/7)]]
sage: J1(13).cuspidal_subgroup().gens()
[[(1/19, 0, 9/19, 9/19)], [(0, 1/19, 0, 9/19)]]
sage: J0(22).torsion_subgroup(6).gens()
[[(1/6, 0, 0, 0)], [(0, 1/6, 0, 0)], [(0, 0, 1/6, 0)], [(0, 0, 0, 1/6)]]
```

```
>>> from sage.all import *
>>> J0(Integer(11)).cuspidal_subgroup().gens()
[[(0, 1/5)]]
>>> J0(Integer(37)).cuspidal_subgroup().gens()
[[(0, 0, 0, 1/3)]]
>>> J0(Integer(43)).cuspidal_subgroup().gens()
[[(0, 1/7, 0, 6/7, 0, 5/7)]]
>>> J1(Integer(13)).cuspidal_subgroup().gens()
[[(1/19, 0, 9/19, 9/19)], [(0, 1/19, 0, 9/19)]]
>>> J0(Integer(22)).torsion_subgroup(Integer(6)).gens()
[[(1/6, 0, 0, 0)], [(0, 1/6, 0, 0)], [(0, 0, 1/6, 0)], [(0, 0, 0, 1/6)]]
```

intersection(*other*)

Return the intersection of the finite subgroups `self` and `other`.

INPUT:

- `other` – a finite group

OUTPUT: a finite group

EXAMPLES:

```
sage: E11a0, E11a1, B = J0(33)
sage: G = E11a0.torsion_subgroup(6); H = E11a0.torsion_subgroup(9)
sage: G.intersection(H)
Finite subgroup with invariants [3, 3] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
sage: W = E11a1.torsion_subgroup(15)
sage: G.intersection(W)
Finite subgroup with invariants [] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
sage: E11a0.intersection(E11a1)[0]
Finite subgroup with invariants [5] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
```

```
>>> from sage.all import *
>>> E11a0, E11a1, B = J0(Integer(33))
>>> G = E11a0.torsion_subgroup(Integer(6)); H = E11a0.torsion_
↪subgroup(Integer(9))
>>> G.intersection(H)
Finite subgroup with invariants [3, 3] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> W = E11a1.torsion_subgroup(Integer(15))
>>> G.intersection(W)
Finite subgroup with invariants [] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> E11a0.intersection(E11a1)[Integer(0)]
Finite subgroup with invariants [5] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
```

We intersect subgroups of different abelian varieties.

```
sage: E11a0, E11a1, B = J0(33)
sage: G = E11a0.torsion_subgroup(5); H = E11a1.torsion_subgroup(5)
sage: G.intersection(H)
Finite subgroup with invariants [5] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
sage: E11a0.intersection(E11a1)[0]
Finite subgroup with invariants [5] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
```

```
>>> from sage.all import *
>>> E11a0, E11a1, B = J0(Integer(33))
>>> G = E11a0.torsion_subgroup(Integer(5)); H = E11a1.torsion_
↪subgroup(Integer(5))
>>> G.intersection(H)
Finite subgroup with invariants [5] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> E11a0.intersection(E11a1)[Integer(0)]
Finite subgroup with invariants [5] over QQ of
```

(continues on next page)

(continued from previous page)

```
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
```

We intersect abelian varieties with subgroups:

```
sage: t = J0(33).hecke_operator(7)
sage: G = t.kernel()[0]; G
Finite subgroup with invariants [2, 2, 2, 2, 4, 4] over QQ of
 Abelian variety J0(33) of dimension 3
sage: A = J0(33).old_subvariety()
sage: A.intersection(G)
Finite subgroup with invariants [2, 2, 2, 2] over QQ of
 Abelian subvariety of dimension 2 of J0(33)
sage: A.hecke_operator(7).kernel()[0]
Finite subgroup with invariants [2, 2, 2, 2] over QQ of
 Abelian subvariety of dimension 2 of J0(33)
sage: B = J0(33).new_subvariety()
sage: B.intersection(G)
Finite subgroup with invariants [4, 4] over QQ of
 Abelian subvariety of dimension 1 of J0(33)
sage: B.hecke_operator(7).kernel()[0]
Finite subgroup with invariants [4, 4] over QQ of
 Abelian subvariety of dimension 1 of J0(33)
sage: A.intersection(B)[0]
Finite subgroup with invariants [3, 3] over QQ of
 Abelian subvariety of dimension 2 of J0(33)
```

```
>>> from sage.all import *
>>> t = J0(Integer(33)).hecke_operator(Integer(7))
>>> G = t.kernel()[Integer(0)]; G
Finite subgroup with invariants [2, 2, 2, 2, 4, 4] over QQ of
 Abelian variety J0(33) of dimension 3
>>> A = J0(Integer(33)).old_subvariety()
>>> A.intersection(G)
Finite subgroup with invariants [2, 2, 2, 2] over QQ of
 Abelian subvariety of dimension 2 of J0(33)
>>> A.hecke_operator(Integer(7)).kernel()[Integer(0)]
Finite subgroup with invariants [2, 2, 2, 2] over QQ of
 Abelian subvariety of dimension 2 of J0(33)
>>> B = J0(Integer(33)).new_subvariety()
>>> B.intersection(G)
Finite subgroup with invariants [4, 4] over QQ of
 Abelian subvariety of dimension 1 of J0(33)
>>> B.hecke_operator(Integer(7)).kernel()[Integer(0)]
Finite subgroup with invariants [4, 4] over QQ of
 Abelian subvariety of dimension 1 of J0(33)
>>> A.intersection(B)[Integer(0)]
Finite subgroup with invariants [3, 3] over QQ of
 Abelian subvariety of dimension 2 of J0(33)
```

invariants()

Return elementary invariants of this abelian group, by which we mean a nondecreasing (immutable) sequence of integers n_i, $1 \leq i \leq k$, with n_i dividing n_{i+1}, and such that this group is abstractly isomorphic to $\mathbf{Z}/n_1\mathbf{Z} \times \cdots \times \mathbf{Z}/n_k\mathbf{Z}$.

EXAMPLES:

```
sage: J = J0(38)
sage: C = J.cuspidal_subgroup(); C
Finite subgroup with invariants [3, 45] over QQ of
 Abelian variety J0(38) of dimension 4
sage: v = C.invariants(); v
[3, 45]
sage: v[0] = 5
Traceback (most recent call last):
...
ValueError: object is immutable; please change a copy instead.
sage: type(v[0])
<class 'sage.rings.integer.Integer'>
```

```
>>> from sage.all import *
>>> J = J0(Integer(38))
>>> C = J.cuspidal_subgroup(); C
Finite subgroup with invariants [3, 45] over QQ of
 Abelian variety J0(38) of dimension 4
>>> v = C.invariants(); v
[3, 45]
>>> v[Integer(0)] = Integer(5)
Traceback (most recent call last):
...
ValueError: object is immutable; please change a copy instead.
>>> type(v[Integer(0)])
<class 'sage.rings.integer.Integer'>
```

```
sage: C * 3
Finite subgroup with invariants [15] over QQ of
 Abelian variety J0(38) of dimension 4
```

```
>>> from sage.all import *
>>> C * Integer(3)
Finite subgroup with invariants [15] over QQ of
 Abelian variety J0(38) of dimension 4
```

An example involving another cuspidal subgroup:

```
sage: C = J0(22).cuspidal_subgroup(); C
Finite subgroup with invariants [5, 5] over QQ of
 Abelian variety J0(22) of dimension 2
sage: C.lattice()
Free module of degree 4 and rank 4 over Integer Ring
Echelon basis matrix:
[1/5 1/5 4/5   0]
[  0   1   0   0]
[  0   0   1   0]
[  0   0   0 1/5]
sage: C.invariants()
[5, 5]
```

```
>>> from sage.all import *
>>> C = J0(Integer(22)).cuspidal_subgroup(); C
Finite subgroup with invariants [5, 5] over QQ of
 Abelian variety J0(22) of dimension 2
>>> C.lattice()
Free module of degree 4 and rank 4 over Integer Ring
Echelon basis matrix:
[1/5 1/5 4/5   0]
[  0   1   0   0]
[  0   0   1   0]
[  0   0   0 1/5]
>>> C.invariants()
[5, 5]
```

is_subgroup(*other*)

Return `True` exactly if `self` is a subgroup of `other`, and both are defined as subgroups of the same ambient abelian variety.

EXAMPLES:

```
sage: C = J0(22).cuspidal_subgroup()
sage: H = C.subgroup([C.0])
sage: K = C.subgroup([C.1])
sage: H.is_subgroup(K)
False
sage: K.is_subgroup(H)
False
sage: K.is_subgroup(C)
True
sage: H.is_subgroup(C)
True
```

```
>>> from sage.all import *
>>> C = J0(Integer(22)).cuspidal_subgroup()
>>> H = C.subgroup([C.gen(0)])
>>> K = C.subgroup([C.gen(1)])
>>> H.is_subgroup(K)
False
>>> K.is_subgroup(H)
False
>>> K.is_subgroup(C)
True
>>> H.is_subgroup(C)
True
```

lattice()

Return the lattice corresponding to this subgroup in the rational homology of the modular Jacobian product. The elements of the subgroup are represented by vectors in the ambient vector space (the rational homology), and this returns the lattice they span. EXAMPLES:

```
sage: J = J0(33); C = J[0].cuspidal_subgroup(); C
Finite subgroup with invariants [5] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
```

(continues on next page)

(continued from previous page)

```
sage: C.lattice()
Free module of degree 6 and rank 2 over Integer Ring
Echelon basis matrix:
[ 1/5 13/5   -2 -4/5    2 -1/5]
[   0    3   -2   -1    2    0]
```

```
>>> from sage.all import *
>>> J = J0(Integer(33)); C = J[Integer(0)].cuspidal_subgroup(); C
Finite subgroup with invariants [5] over QQ of
 Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> C.lattice()
Free module of degree 6 and rank 2 over Integer Ring
Echelon basis matrix:
[ 1/5 13/5   -2 -4/5    2 -1/5]
[   0    3   -2   -1    2    0]
```

order()

Return the order (number of elements) of this finite subgroup.

EXAMPLES:

```
sage: J = J0(42)
sage: C = J.cuspidal_subgroup()
sage: C.order()
2304
```

```
>>> from sage.all import *
>>> J = J0(Integer(42))
>>> C = J.cuspidal_subgroup()
>>> C.order()
2304
```

subgroup(*gens*)

Return the subgroup of `self` spanned by the given generators, which must all be elements of `self`.

EXAMPLES:

```
sage: J = J0(23)
sage: G = J.torsion_subgroup(11); G
Finite subgroup with invariants [11, 11, 11, 11] over QQ of
 Abelian variety J0(23) of dimension 2
```

```
>>> from sage.all import *
>>> J = J0(Integer(23))
>>> G = J.torsion_subgroup(Integer(11)); G
Finite subgroup with invariants [11, 11, 11, 11] over QQ of
 Abelian variety J0(23) of dimension 2
```

We create the subgroup of the 11-torsion subgroup of $J_0(23)$ generated by the first 11-torsion point:

```
sage: H = G.subgroup([G.0]); H
Finite subgroup with invariants [11] over QQbar of
```

(continues on next page)

(continued from previous page)

```
 Abelian variety J0(23) of dimension 2
sage: H.invariants()
[11]
```

```
>>> from sage.all import *
>>> H = G.subgroup([G.gen(0)]); H
Finite subgroup with invariants [11] over QQbar of
 Abelian variety J0(23) of dimension 2
>>> H.invariants()
[11]
```

We can also create a subgroup from a list of objects that can be converted into the ambient rational homology:

```
sage: H == G.subgroup([[1/11,0,0,0]])
True
```

```
>>> from sage.all import *
>>> H == G.subgroup([[Integer(1)/Integer(11),Integer(0),Integer(0),
↪Integer(0)]])
True
```

class sage.modular.abvar.finite_subgroup.**FiniteSubgroup_lattice**(*abvar*, *lattice*, *field_of_definition=None*, *check=True*)

Bases: *FiniteSubgroup*

A finite subgroup of a modular abelian variety that is defined by a given lattice.

INPUT:

- `abvar` – a modular abelian variety
- `lattice` – a lattice that contains the lattice of abvar
- `field_of_definition` – the field of definition of this finite group scheme
- `check` – boolean (default: `True`); whether or not to check that lattice contains the abvar lattice

EXAMPLES:

```
sage: J = J0(11)
sage: G = J.finite_subgroup([[1/3,0], [0,1/5]]); G
Finite subgroup with invariants [15] over QQbar of
 Abelian variety J0(11) of dimension 1
```

```
>>> from sage.all import *
>>> J = J0(Integer(11))
>>> G = J.finite_subgroup([[Integer(1)/Integer(3),Integer(0)], [Integer(0),
↪Integer(1)/Integer(5)]]); G
Finite subgroup with invariants [15] over QQbar of
 Abelian variety J0(11) of dimension 1
```

lattice()

Return lattice that defines this finite subgroup.

EXAMPLES:

```
sage: J = J0(11)
sage: G = J.finite_subgroup([[1/3,0], [0,1/5]]); G
Finite subgroup with invariants [15] over QQbar of
 Abelian variety J0(11) of dimension 1
sage: G.lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[1/3   0]
[  0 1/5]
```

```
>>> from sage.all import *
>>> J = J0(Integer(11))
>>> G = J.finite_subgroup([[Integer(1)/Integer(3),Integer(0)], [Integer(0),
↪Integer(1)/Integer(5)]]); G
Finite subgroup with invariants [15] over QQbar of
 Abelian variety J0(11) of dimension 1
>>> G.lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[1/3   0]
[  0 1/5]
```

CHAPTER

FIVE

TORSION SUBGROUPS OF MODULAR ABELIAN VARIETIES

Sage can compute information about the structure of the torsion subgroup of a modular abelian variety. Sage computes a multiple of the order by computing the greatest common divisor of the orders of the torsion subgroup of the reduction of the abelian variety modulo p for various primes p. Sage computes a divisor of the order by computing the rational cuspidal subgroup. When these two bounds agree (which is often the case), we determine the exact structure of the torsion subgroup.

AUTHORS:

- William Stein (2007-03)

EXAMPLES: First we consider $J_0(50)$ where everything works out nicely:

```
sage: J = J0(50)
sage: T = J.rational_torsion_subgroup(); T
Torsion subgroup of Abelian variety J0(50) of dimension 2
sage: T.multiple_of_order()
15
sage: T.divisor_of_order()
15
sage: T.gens()
[[(1/15, 3/5, 2/5, 14/15)]]
sage: T.invariants()
[15]
sage: d = J.decomposition(); d
[Simple abelian subvariety 50a(1,50) of dimension 1 of J0(50),
 Simple abelian subvariety 50b(1,50) of dimension 1 of J0(50)]
sage: d[0].rational_torsion_subgroup().order()
3
sage: d[1].rational_torsion_subgroup().order()
5
```

```
>>> from sage.all import *
>>> J = J0(Integer(50))
>>> T = J.rational_torsion_subgroup(); T
Torsion subgroup of Abelian variety J0(50) of dimension 2
>>> T.multiple_of_order()
15
>>> T.divisor_of_order()
15
>>> T.gens()
[[(1/15, 3/5, 2/5, 14/15)]]
>>> T.invariants()
```

(continues on next page)

(continued from previous page)

```
[15]
>>> d = J.decomposition(); d
[Simple abelian subvariety 50a(1,50) of dimension 1 of J0(50),
 Simple abelian subvariety 50b(1,50) of dimension 1 of J0(50)]
>>> d[Integer(0)].rational_torsion_subgroup().order()
3
>>> d[Integer(1)].rational_torsion_subgroup().order()
5
```

Next we make a table of the upper and lower bounds for each new factor.

```
sage: for N in range(1,38):
....:     for A in J0(N).new_subvariety().decomposition():
....:         T = A.rational_torsion_subgroup()
....:         print('%-5s%-5s%-5s%-5s'%(N, A.dimension(), T.divisor_of_order(), T.
↪multiple_of_order()))
11   1    5    5
14   1    6    6
15   1    8    8
17   1    4    4
19   1    3    3
20   1    6    6
21   1    8    8
23   2    11   11
24   1    8    8
26   1    3    3
26   1    7    7
27   1    3    3
29   2    7    7
30   1    6    6
31   2    5    5
32   1    4    4
33   1    4    4
34   1    6    6
35   1    3    3
35   2    16   16
36   1    6    6
37   1    1    1
37   1    3    3
```

```
>>> from sage.all import *
>>> for N in range(Integer(1),Integer(38)):
...     for A in J0(N).new_subvariety().decomposition():
...         T = A.rational_torsion_subgroup()
...         print('%-5s%-5s%-5s%-5s'%(N, A.dimension(), T.divisor_of_order(), T.
↪multiple_of_order()))
11   1    5    5
14   1    6    6
15   1    8    8
17   1    4    4
19   1    3    3
20   1    6    6
```

(continues on next page)

(continued from previous page)

```
21   1    8    8
23   2    11   11
24   1    8    8
26   1    3    3
26   1    7    7
27   1    3    3
29   2    7    7
30   1    6    6
31   2    5    5
32   1    4    4
33   1    4    4
34   1    6    6
35   1    3    3
35   2    16   16
36   1    6    6
37   1    1    1
37   1    3    3
```

class sage.modular.abvar.torsion_subgroup.**QQbarTorsionSubgroup**(*abvar*)

Bases: Module

Group of all torsion points over the algebraic closure on an abelian variety.

INPUT:

- abvar – an abelian variety

EXAMPLES:

```
sage: A = J0(23)
sage: A.qbar_torsion_subgroup()                                              #␣
↪needs sage.rings.number_field
Group of all torsion points in QQbar on Abelian variety J0(23) of dimension 2
```

```
>>> from sage.all import *
>>> A = J0(Integer(23))
>>> A.qbar_torsion_subgroup()                                                # needs␣
↪sage.rings.number_field
Group of all torsion points in QQbar on Abelian variety J0(23) of dimension 2
```

Element

alias of TorsionPoint

abelian_variety()

Return the abelian variety that this is the set of all torsion points on.

OUTPUT: abelian variety

EXAMPLES:

```
sage: J0(23).qbar_torsion_subgroup().abelian_variety()                       #␣
↪needs sage.rings.number_field
Abelian variety J0(23) of dimension 2
```

```
>>> from sage.all import *
>>> J0(Integer(23)).qbar_torsion_subgroup().abelian_variety()
↪     # needs sage.rings.number_field
Abelian variety J0(23) of dimension 2
```

field_of_definition()

Return the field of definition of this subgroup. Since this is the group of all torsion it is defined over the base field of this abelian variety.

OUTPUT: a field

EXAMPLES:

```
sage: J0(23).qbar_torsion_subgroup().field_of_definition()
↪needs sage.rings.number_field
Rational Field
```

```
>>> from sage.all import *
>>> J0(Integer(23)).qbar_torsion_subgroup().field_of_definition()
↪     # needs sage.rings.number_field
Rational Field
```

class sage.modular.abvar.torsion_subgroup.**RationalTorsionSubgroup**(*abvar*)

Bases: *FiniteSubgroup*

The torsion subgroup of a modular abelian variety.

divisor_of_order()

Return a divisor of the order of this torsion subgroup of a modular abelian variety.

OUTPUT: a divisor of this torsion subgroup

EXAMPLES:

```
sage: t = J0(37)[1].rational_torsion_subgroup()
sage: t.divisor_of_order()
3

sage: J = J1(19)
sage: J.rational_torsion_subgroup().divisor_of_order()
4383

sage: J = J0(45)
sage: J.rational_cusp_subgroup().order()
32
sage: J.rational_cuspidal_subgroup().order()
64
sage: J.rational_torsion_subgroup().divisor_of_order()
64
```

```
>>> from sage.all import *
>>> t = J0(Integer(37))[Integer(1)].rational_torsion_subgroup()
>>> t.divisor_of_order()
3
```

(continues on next page)

(continued from previous page)

```
>>> J = J1(Integer(19))
>>> J.rational_torsion_subgroup().divisor_of_order()
4383

>>> J = J0(Integer(45))
>>> J.rational_cusp_subgroup().order()
32
>>> J.rational_cuspidal_subgroup().order()
64
>>> J.rational_torsion_subgroup().divisor_of_order()
64
```

lattice()

Return lattice that defines this torsion subgroup, if possible.

> **Warning**
>
> There is no known algorithm in general to compute the rational torsion subgroup. Use rational_cusp_group to obtain a subgroup of the rational torsion subgroup in general.

EXAMPLES:

```
sage: J0(11).rational_torsion_subgroup().lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[  1   0]
[  0 1/5]
```

```
>>> from sage.all import *
>>> J0(Integer(11)).rational_torsion_subgroup().lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[  1   0]
[  0 1/5]
```

The following fails because in fact I know of no (reasonable) algorithm to provably compute the torsion subgroup in general.

```
sage: T = J0(33).rational_torsion_subgroup()
sage: T.lattice()
Traceback (most recent call last):
...
NotImplementedError: unable to compute the rational torsion subgroup
in this case (there is no known general algorithm yet)
```

```
>>> from sage.all import *
>>> T = J0(Integer(33)).rational_torsion_subgroup()
>>> T.lattice()
Traceback (most recent call last):
...
```

(continues on next page)

(continued from previous page)

```
NotImplementedError: unable to compute the rational torsion subgroup
in this case (there is no known general algorithm yet)
```

The problem is that the multiple of the order obtained by counting points over finite fields is twice the divisor of the order got from the rational cuspidal subgroup.

```
sage: T.multiple_of_order(30)
200
sage: J0(33).rational_cusp_subgroup().order()
100
```

```
>>> from sage.all import *
>>> T.multiple_of_order(Integer(30))
200
>>> J0(Integer(33)).rational_cusp_subgroup().order()
100
```

multiple_of_order(*maxp=None*, *proof=True*)

Return a multiple of the order.

INPUT:

- `proof` – boolean (default: `True`)

The computation of the rational torsion order of J1(p) is conjectural and will only be used if proof=False. See Section 6.2.3 of [CES2003].

EXAMPLES:

```
sage: J = J1(11); J
Abelian variety J1(11) of dimension 1
sage: J.rational_torsion_subgroup().multiple_of_order()
5

sage: J = J0(17)
sage: J.rational_torsion_subgroup().order()
4
```

```
>>> from sage.all import *
>>> J = J1(Integer(11)); J
Abelian variety J1(11) of dimension 1
>>> J.rational_torsion_subgroup().multiple_of_order()
5

>>> J = J0(Integer(17))
>>> J.rational_torsion_subgroup().order()
4
```

This is an example where proof=False leads to a better bound and better performance.

```
sage: J = J1(23)
sage: J.rational_torsion_subgroup().multiple_of_order()  # long time (2s)
9406793
```

(continues on next page)

(continued from previous page)

```
sage: J.rational_torsion_subgroup().multiple_of_order(proof=False)
408991
```

```
>>> from sage.all import *
>>> J = J1(Integer(23))
>>> J.rational_torsion_subgroup().multiple_of_order()  # long time (2s)
9406793
>>> J.rational_torsion_subgroup().multiple_of_order(proof=False)
408991
```

multiple_of_order_using_frobp(*maxp=None*)

Return a multiple of the order of this torsion group.

In the $Gamma_0$ case, the multiple is computed using characteristic polynomials of Hecke operators of odd index not dividing the level. In the $Gamma_1$ case, the multiple is computed by expressing the frobenius polynomial in terms of the characteristic polynomial of left multiplication by a_p for odd primes p not dividing the level.

INPUT:

- `maxp` – (default: `None`) if `maxp` is `None`, return gcd of best bound computed so far with bound obtained by computing GCD's of orders modulo p until this gcd stabilizes for 3 successive primes. If `maxp` is given, just use all primes up to and including `maxp`.

EXAMPLES:

```
sage: J = J0(11)
sage: G = J.rational_torsion_subgroup()
sage: G.multiple_of_order_using_frobp(11)
5
```

```
>>> from sage.all import *
>>> J = J0(Integer(11))
>>> G = J.rational_torsion_subgroup()
>>> G.multiple_of_order_using_frobp(Integer(11))
5
```

Increasing maxp may yield a tighter bound. If maxp=None, then Sage will use more primes until the multiple stabilizes for 3 successive primes.

```
sage: J = J0(389)
sage: G = J.rational_torsion_subgroup(); G
Torsion subgroup of Abelian variety J0(389) of dimension 32
sage: G.multiple_of_order_using_frobp()
97
sage: [G.multiple_of_order_using_frobp(p) for p in prime_range(3,11)]
[92645296242160800, 7275, 291]
sage: [G.multiple_of_order_using_frobp(p) for p in prime_range(3,13)]
[92645296242160800, 7275, 291, 97]
sage: [G.multiple_of_order_using_frobp(p) for p in prime_range(3,19)]
[92645296242160800, 7275, 291, 97, 97, 97]
```

```
>>> from sage.all import *
>>> J = J0(Integer(389))
>>> G = J.rational_torsion_subgroup(); G
Torsion subgroup of Abelian variety J0(389) of dimension 32
>>> G.multiple_of_order_using_frobp()
97
>>> [G.multiple_of_order_using_frobp(p) for p in prime_range(Integer(3),
↪Integer(11))]
[92645296242160800, 7275, 291]
>>> [G.multiple_of_order_using_frobp(p) for p in prime_range(Integer(3),
↪Integer(13))]
[92645296242160800, 7275, 291, 97]
>>> [G.multiple_of_order_using_frobp(p) for p in prime_range(Integer(3),
↪Integer(19))]
[92645296242160800, 7275, 291, 97, 97, 97]
```

We can compute the multiple of order of the torsion subgroup for Gamma0 and Gamma1 varieties, and their products.

```
sage: A = J0(11) * J0(33)
sage: A.rational_torsion_subgroup().multiple_of_order_using_frobp()
1000

sage: A = J1(23)
sage: A.rational_torsion_subgroup().multiple_of_order_using_frobp()
9406793
sage: A.rational_torsion_subgroup().multiple_of_order_using_frobp(maxp=50)
408991

sage: A = J1(19) * J0(21)
sage: A.rational_torsion_subgroup().multiple_of_order_using_frobp()
35064
```

```
>>> from sage.all import *
>>> A = J0(Integer(11)) * J0(Integer(33))
>>> A.rational_torsion_subgroup().multiple_of_order_using_frobp()
1000

>>> A = J1(Integer(23))
>>> A.rational_torsion_subgroup().multiple_of_order_using_frobp()
9406793
>>> A.rational_torsion_subgroup().multiple_of_order_using_
↪frobp(maxp=Integer(50))
408991

>>> A = J1(Integer(19)) * J0(Integer(21))
>>> A.rational_torsion_subgroup().multiple_of_order_using_frobp()
35064
```

The next example illustrates calling this function with a larger input and how the result may be cached when maxp is None:

```
sage: T = J0(43)[1].rational_torsion_subgroup()
sage: T.multiple_of_order_using_frobp()
14
sage: T.multiple_of_order_using_frobp(50)
7
sage: T.multiple_of_order_using_frobp()
7
```

```
>>> from sage.all import *
>>> T = J0(Integer(43))[Integer(1)].rational_torsion_subgroup()
>>> T.multiple_of_order_using_frobp()
14
>>> T.multiple_of_order_using_frobp(Integer(50))
7
>>> T.multiple_of_order_using_frobp()
7
```

This function is not implemented for general congruence subgroups unless the dimension is zero.

```
sage: A = JH(13,[2]); A
Abelian variety J0(13) of dimension 0
sage: A.rational_torsion_subgroup().multiple_of_order_using_frobp()
1

sage: A = JH(15, [2]); A
Abelian variety JH(15,[2]) of dimension 1
sage: A.rational_torsion_subgroup().multiple_of_order_using_frobp()
Traceback (most recent call last):
...
NotImplementedError: torsion multiple only implemented for Gamma0 and Gamma1
```

```
>>> from sage.all import *
>>> A = JH(Integer(13),[Integer(2)]); A
Abelian variety J0(13) of dimension 0
>>> A.rational_torsion_subgroup().multiple_of_order_using_frobp()
1

>>> A = JH(Integer(15), [Integer(2)]); A
Abelian variety JH(15,[2]) of dimension 1
>>> A.rational_torsion_subgroup().multiple_of_order_using_frobp()
Traceback (most recent call last):
...
NotImplementedError: torsion multiple only implemented for Gamma0 and Gamma1
```

order(*proof=True*)

Return the order of the torsion subgroup of this modular abelian variety.

This function may fail if the multiple obtained by counting points modulo p exceeds the divisor obtained from the rational cuspidal subgroup.

The computation of the rational torsion order of J1(p) is conjectural and will only be used if `proof=False`. See Section 6.2.3 of [CES2003].

INPUT:

- `proof` – boolean (default: `True`)

OUTPUT: the order of this torsion subgroup

EXAMPLES:

```
sage: A = J0(11)
sage: A.rational_torsion_subgroup().order()
5
sage: A = J0(23)
sage: A.rational_torsion_subgroup().order()
11
sage: T = J0(37)[1].rational_torsion_subgroup()
sage: T.order()
3

sage: J = J1(13)
sage: J.rational_torsion_subgroup().order()
19
```

```
>>> from sage.all import *
>>> A = J0(Integer(11))
>>> A.rational_torsion_subgroup().order()
5
>>> A = J0(Integer(23))
>>> A.rational_torsion_subgroup().order()
11
>>> T = J0(Integer(37))[Integer(1)].rational_torsion_subgroup()
>>> T.order()
3

>>> J = J1(Integer(13))
>>> J.rational_torsion_subgroup().order()
19
```

Sometimes the order can only be computed with `proof=False`.

```
sage: J = J1(23)
sage: J.rational_torsion_subgroup().order()
Traceback (most recent call last):
...
RuntimeError: Unable to compute order of torsion subgroup
(it is in [408991, 9406793])

sage: J.rational_torsion_subgroup().order(proof=False)
408991
```

```
>>> from sage.all import *
>>> J = J1(Integer(23))
>>> J.rational_torsion_subgroup().order()
Traceback (most recent call last):
...
RuntimeError: Unable to compute order of torsion subgroup
(it is in [408991, 9406793])
```

(continues on next page)

(continued from previous page)

```
>>> J.rational_torsion_subgroup().order(proof=False)
408991
```

possible_orders(*proof=True*)

Return the possible orders of this torsion subgroup. Outside of special cases, this is done by computing a divisor and multiple of the order.

INPUT:

- `proof` – boolean (default: `True`)

OUTPUT: an array of positive integers

The computation of the rational torsion order of J1(p) is conjectural and will only be used if `proof=False`. See Section 6.2.3 of [CES2003].

EXAMPLES:

```
sage: J0(11).rational_torsion_subgroup().possible_orders()
[5]
sage: J0(33).rational_torsion_subgroup().possible_orders()
[100, 200]

sage: J1(13).rational_torsion_subgroup().possible_orders()
[19]
sage: J1(16).rational_torsion_subgroup().possible_orders()
[1, 2, 4, 5, 10, 20]
```

```
>>> from sage.all import *
>>> J0(Integer(11)).rational_torsion_subgroup().possible_orders()
[5]
>>> J0(Integer(33)).rational_torsion_subgroup().possible_orders()
[100, 200]

>>> J1(Integer(13)).rational_torsion_subgroup().possible_orders()
[19]
>>> J1(Integer(16)).rational_torsion_subgroup().possible_orders()
[1, 2, 4, 5, 10, 20]
```

CHAPTER

SIX

CUSPIDAL SUBGROUPS OF MODULAR ABELIAN VARIETIES

AUTHORS:

- William Stein (2007-03, 2008-02)

EXAMPLES: We compute the cuspidal subgroup of $J_1(13)$:

```
sage: A = J1(13)
sage: C = A.cuspidal_subgroup(); C
Finite subgroup with invariants [19, 19] over QQ of Abelian variety J1(13) of␣
↪dimension 2
sage: C.gens()
[[(1/19, 0, 9/19, 9/19)], [(0, 1/19, 0, 9/19)]]
sage: C.order()
361
sage: C.invariants()
[19, 19]
```

```
>>> from sage.all import *
>>> A = J1(Integer(13))
>>> C = A.cuspidal_subgroup(); C
Finite subgroup with invariants [19, 19] over QQ of Abelian variety J1(13) of␣
↪dimension 2
>>> C.gens()
[[(1/19, 0, 9/19, 9/19)], [(0, 1/19, 0, 9/19)]]
>>> C.order()
361
>>> C.invariants()
[19, 19]
```

We compute the cuspidal subgroup of $J_0(54)$:

```
sage: A = J0(54)
sage: C = A.cuspidal_subgroup(); C
Finite subgroup with invariants [3, 3, 3, 3, 3, 9] over QQ of Abelian variety J0(54)␣
↪of dimension 4
sage: C.gens()
[[(1/3, 0, 0, 0, 0, 1/3, 0, 2/3)], [(0, 1/3, 0, 0, 0, 2/3, 0, 1/3)], [(0, 0, 1/9, 1/9,
↪ 1/9, 1/9, 1/9, 2/9)], [(0, 0, 0, 1/3, 0, 1/3, 0, 0)], [(0, 0, 0, 0, 1/3, 1/3, 0, 1/
↪3)], [(0, 0, 0, 0, 0, 0, 1/3, 2/3)]]
sage: C.order()
2187
```

(continues on next page)

(continued from previous page)

```
sage: C.invariants()
[3, 3, 3, 3, 3, 9]
```

```
>>> from sage.all import *
>>> A = J0(Integer(54))
>>> C = A.cuspidal_subgroup(); C
Finite subgroup with invariants [3, 3, 3, 3, 3, 9] over QQ of Abelian variety J0(54)␣
↪of dimension 4
>>> C.gens()
[[(1/3, 0, 0, 0, 0, 1/3, 0, 2/3)], [(0, 1/3, 0, 0, 0, 2/3, 0, 1/3)], [(0, 0, 1/9, 1/9,
↪ 1/9, 1/9, 1/9, 2/9)], [(0, 0, 0, 1/3, 0, 1/3, 0, 0)], [(0, 0, 0, 0, 1/3, 1/3, 0, 1/
↪3)], [(0, 0, 0, 0, 0, 0, 1/3, 2/3)]]
>>> C.order()
2187
>>> C.invariants()
[3, 3, 3, 3, 3, 9]
```

We compute the subgroup of the cuspidal subgroup generated by rational cusps.

```
sage: C = J0(54).rational_cusp_subgroup(); C
Finite subgroup with invariants [3, 3, 9] over QQ of Abelian variety J0(54) of␣
↪dimension 4
sage: C.gens()
[[(1/3, 0, 0, 1/3, 2/3, 1/3, 0, 1/3)], [(0, 0, 1/9, 1/9, 7/9, 7/9, 1/9, 8/9)], [(0, 0,
↪ 0, 0, 0, 0, 1/3, 2/3)]]
sage: C.order()
81
sage: C.invariants()
[3, 3, 9]
```

```
>>> from sage.all import *
>>> C = J0(Integer(54)).rational_cusp_subgroup(); C
Finite subgroup with invariants [3, 3, 9] over QQ of Abelian variety J0(54) of␣
↪dimension 4
>>> C.gens()
[[(1/3, 0, 0, 1/3, 2/3, 1/3, 0, 1/3)], [(0, 0, 1/9, 1/9, 7/9, 7/9, 1/9, 8/9)], [(0, 0,
↪ 0, 0, 0, 0, 1/3, 2/3)]]
>>> C.order()
81
>>> C.invariants()
[3, 3, 9]
```

This might not give us the exact rational torsion subgroup, since it might be bigger than order 81:

```
sage: J0(54).rational_torsion_subgroup().multiple_of_order()
243
```

```
>>> from sage.all import *
>>> J0(Integer(54)).rational_torsion_subgroup().multiple_of_order()
243
```

class sage.modular.abvar.cuspidal_subgroup.**CuspidalSubgroup**(*abvar*, *field_of_definition=Rational Field*)

Bases: *CuspidalSubgroup_generic*

EXAMPLES:

```
sage: a = J0(65)[2]
sage: t = a.cuspidal_subgroup()
sage: t.order()
6
```

```
>>> from sage.all import *
>>> a = J0(Integer(65))[Integer(2)]
>>> t = a.cuspidal_subgroup()
>>> t.order()
6
```

lattice()

Returned cached tuple of vectors that define elements of the rational homology that generate this finite subgroup.

OUTPUT: tuple (cached)

EXAMPLES:

```
sage: J = J0(27)
sage: G = J.cuspidal_subgroup()
sage: G.lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[1/3   0]
[  0 1/3]
```

```
>>> from sage.all import *
>>> J = J0(Integer(27))
>>> G = J.cuspidal_subgroup()
>>> G.lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[1/3   0]
[  0 1/3]
```

Test that the result is cached:

```
sage: G.lattice() is G.lattice()
True
```

```
>>> from sage.all import *
>>> G.lattice() is G.lattice()
True
```

class sage.modular.abvar.cuspidal_subgroup.**CuspidalSubgroup_generic**(*abvar*, *field_of_definition=Rational Field*)

Bases: *FiniteSubgroup*

class sage.modular.abvar.cuspidal_subgroup.**RationalCuspSubgroup**(*abvar*, *field_of_definition=Rational Field*)

Bases: *CuspidalSubgroup_generic*

EXAMPLES:

```
sage: a = J0(65)[2]
sage: t = a.rational_cusp_subgroup()
sage: t.order()
6
```

```
>>> from sage.all import *
>>> a = J0(Integer(65))[Integer(2)]
>>> t = a.rational_cusp_subgroup()
>>> t.order()
6
```

lattice()

Return lattice that defines this group.

OUTPUT: lattice

EXAMPLES:

```
sage: G = J0(27).rational_cusp_subgroup()
sage: G.lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[1/3   0]
[  0   1]
```

```
>>> from sage.all import *
>>> G = J0(Integer(27)).rational_cusp_subgroup()
>>> G.lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[1/3   0]
[  0   1]
```

Test that the result is cached.

```
sage: G.lattice() is G.lattice()
True
```

```
>>> from sage.all import *
>>> G.lattice() is G.lattice()
True
```

class sage.modular.abvar.cuspidal_subgroup.**RationalCuspidalSubgroup**(*abvar*, *field_of_definition=Rational Field*)

Bases: *CuspidalSubgroup_generic*

EXAMPLES:

```
sage: a = J0(65)[2]
sage: t = a.rational_cuspidal_subgroup()
sage: t.order()
6
```

```
>>> from sage.all import *
>>> a = J0(Integer(65))[Integer(2)]
>>> t = a.rational_cuspidal_subgroup()
>>> t.order()
6
```

lattice()

Return lattice that defines this group.

OUTPUT: lattice

EXAMPLES:

```
sage: G = J0(27).rational_cuspidal_subgroup()
sage: G.lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[1/3   0]
[  0   1]
```

```
>>> from sage.all import *
>>> G = J0(Integer(27)).rational_cuspidal_subgroup()
>>> G.lattice()
Free module of degree 2 and rank 2 over Integer Ring
Echelon basis matrix:
[1/3   0]
[  0   1]
```

Test that the result is cached.

```
sage: G.lattice() is G.lattice()
True
```

```
>>> from sage.all import *
>>> G.lattice() is G.lattice()
True
```

sage.modular.abvar.cuspidal_subgroup.**is_rational_cusp_gamma0**(*c*, *N*, *data*)

Return `True` if the rational number c is a rational cusp of level N.

This uses remarks in Glenn Steven's Ph.D. thesis.

INPUT:

- `c` – a cusp
- `N` – positive integer
- `data` – the list [n for n in range(2,N) if gcd(n,N) == 1], which is passed in as a parameter purely for efficiency reasons.

EXAMPLES:

```
sage: from sage.modular.abvar.cuspidal_subgroup import is_rational_cusp_gamma0
sage: N = 27
sage: data = [n for n in range(2,N) if gcd(n,N) == 1]
sage: is_rational_cusp_gamma0(Cusp(1/3), N, data)
False
sage: is_rational_cusp_gamma0(Cusp(1), N, data)
True
sage: is_rational_cusp_gamma0(Cusp(oo), N, data)
True
sage: is_rational_cusp_gamma0(Cusp(2/9), N, data)
False
```

```
>>> from sage.all import *
>>> from sage.modular.abvar.cuspidal_subgroup import is_rational_cusp_gamma0
>>> N = Integer(27)
>>> data = [n for n in range(Integer(2),N) if gcd(n,N) == Integer(1)]
>>> is_rational_cusp_gamma0(Cusp(Integer(1)/Integer(3)), N, data)
False
>>> is_rational_cusp_gamma0(Cusp(Integer(1)), N, data)
True
>>> is_rational_cusp_gamma0(Cusp(oo), N, data)
True
>>> is_rational_cusp_gamma0(Cusp(Integer(2)/Integer(9)), N, data)
False
```

CHAPTER

SEVEN

HOMOLOGY OF MODULAR ABELIAN VARIETIES

Sage can compute with homology groups associated to modular abelian varieties with coefficients in any commutative ring. Supported operations include computing matrices and characteristic polynomials of Hecke operators, rank, and rational decomposition as a direct sum of factors (obtained by cutting out kernels of Hecke operators).

AUTHORS:

- William Stein (2007-03)

EXAMPLES:

```
sage: J = J0(43)
sage: H = J.integral_homology()
sage: H
Integral Homology of Abelian variety J0(43) of dimension 3
sage: H.hecke_matrix(19)
[ 0  0 -2  0  2  0]
[ 2 -4 -2  0  2  0]
[ 0  0 -2 -2  0  0]
[ 2  0 -2 -4  2 -2]
[ 0  2  0 -2 -2  0]
[ 0  2  0 -2  0  0]
sage: H.base_ring()
Integer Ring
sage: d = H.decomposition(); d
[Submodule of rank 2 of Integral Homology of Abelian variety J0(43) of dimension 3,
 Submodule of rank 4 of Integral Homology of Abelian variety J0(43) of dimension 3]
sage: a = d[0]
sage: a.hecke_matrix(5)
[-4  0]
[ 0 -4]
sage: a.T(7)
Hecke operator T_7 on
 Submodule of rank 2 of Integral Homology of Abelian variety J0(43) of dimension 3
```

```
>>> from sage.all import *
>>> J = J0(Integer(43))
>>> H = J.integral_homology()
>>> H
Integral Homology of Abelian variety J0(43) of dimension 3
>>> H.hecke_matrix(Integer(19))
[ 0  0 -2  0  2  0]
[ 2 -4 -2  0  2  0]
```

(continues on next page)

(continued from previous page)

```
[ 0  0 -2 -2  0  0]
[ 2  0 -2 -4  2 -2]
[ 0  2  0 -2 -2  0]
[ 0  2  0 -2  0  0]
>>> H.base_ring()
Integer Ring
>>> d = H.decomposition(); d
[Submodule of rank 2 of Integral Homology of Abelian variety J0(43) of dimension 3,
 Submodule of rank 4 of Integral Homology of Abelian variety J0(43) of dimension 3]
>>> a = d[Integer(0)]
>>> a.hecke_matrix(Integer(5))
[-4  0]
[ 0 -4]
>>> a.T(Integer(7))
Hecke operator T_7 on
 Submodule of rank 2 of Integral Homology of Abelian variety J0(43) of dimension 3
```

class sage.modular.abvar.homology.**Homology**(*base_ring*, *level*, *weight*, *category=None*)

Bases: HeckeModule_free_module

A homology group of an abelian variety, equipped with a Hecke action.

hecke_polynomial(*n*, *var='x'*)

Return the n-th Hecke polynomial in the given variable.

INPUT:

- `n` – positive integer
- `var` – string (default: `'x'`); the variable name

OUTPUT: a polynomial over **Z** in the given variable

EXAMPLES:

```
sage: H = J0(43).integral_homology(); H
Integral Homology of Abelian variety J0(43) of dimension 3
sage: f = H.hecke_polynomial(3); f
x^6 + 4*x^5 - 16*x^3 - 12*x^2 + 16*x + 16
sage: parent(f)
Univariate Polynomial Ring in x over Integer Ring
sage: H.hecke_polynomial(3,'w')
w^6 + 4*w^5 - 16*w^3 - 12*w^2 + 16*w + 16
```

```
>>> from sage.all import *
>>> H = J0(Integer(43)).integral_homology(); H
Integral Homology of Abelian variety J0(43) of dimension 3
>>> f = H.hecke_polynomial(Integer(3)); f
x^6 + 4*x^5 - 16*x^3 - 12*x^2 + 16*x + 16
>>> parent(f)
Univariate Polynomial Ring in x over Integer Ring
>>> H.hecke_polynomial(Integer(3),'w')
w^6 + 4*w^5 - 16*w^3 - 12*w^2 + 16*w + 16
```

class sage.modular.abvar.homology.**Homology_abvar**(*abvar*, *base*)

Bases: *Homology*

The homology of a modular abelian variety.

abelian_variety()

Return the abelian variety that this is the homology of.

EXAMPLES:

```
sage: H = J0(48).homology()
sage: H.abelian_variety()
Abelian variety J0(48) of dimension 3
```

```
>>> from sage.all import *
>>> H = J0(Integer(48)).homology()
>>> H.abelian_variety()
Abelian variety J0(48) of dimension 3
```

ambient_hecke_module()

Return the ambient Hecke module that this homology is contained in.

EXAMPLES:

```
sage: H = J0(48).homology(); H
Integral Homology of Abelian variety J0(48) of dimension 3
sage: H.ambient_hecke_module()
Integral Homology of Abelian variety J0(48) of dimension 3
```

```
>>> from sage.all import *
>>> H = J0(Integer(48)).homology(); H
Integral Homology of Abelian variety J0(48) of dimension 3
>>> H.ambient_hecke_module()
Integral Homology of Abelian variety J0(48) of dimension 3
```

free_module()

Return the underlying free module of this homology group.

EXAMPLES:

```
sage: H = J0(48).homology()
sage: H.free_module()
Ambient free module of rank 6 over the principal ideal domain Integer Ring
```

```
>>> from sage.all import *
>>> H = J0(Integer(48)).homology()
>>> H.free_module()
Ambient free module of rank 6 over the principal ideal domain Integer Ring
```

gen(n)

Return n-th generator of `self`.

This is not yet implemented!

EXAMPLES:

```
sage: H = J0(37).homology()
sage: H.gen(0)     # this will change
```

(continues on next page)

(continued from previous page)

```
Traceback (most recent call last):
...
NotImplementedError: homology classes not yet implemented
```

```
>>> from sage.all import *
>>> H = J0(Integer(37)).homology()
>>> H.gen(Integer(0))    # this will change
Traceback (most recent call last):
...
NotImplementedError: homology classes not yet implemented
```

gens()

Return generators of `self`.

This is not yet implemented!

EXAMPLES:

```
sage: H = J0(37).homology()
sage: H.gens()    # this will change
Traceback (most recent call last):
...
NotImplementedError: homology classes not yet implemented
```

```
>>> from sage.all import *
>>> H = J0(Integer(37)).homology()
>>> H.gens()    # this will change
Traceback (most recent call last):
...
NotImplementedError: homology classes not yet implemented
```

hecke_bound()

Return bound on the number of Hecke operators needed to generate the Hecke algebra as a $\mathbf{Z}$-module acting on this space.

EXAMPLES:

```
sage: J0(48).homology().hecke_bound()
16
sage: J1(15).homology().hecke_bound()
32
```

```
>>> from sage.all import *
>>> J0(Integer(48)).homology().hecke_bound()
16
>>> J1(Integer(15)).homology().hecke_bound()
32
```

hecke_matrix(n)

Return the matrix of the n-th Hecke operator acting on this homology group.

INPUT:

- `n` – positive integer

OUTPUT: a matrix over the coefficient ring of this homology group

EXAMPLES:

```
sage: H = J0(23).integral_homology()
sage: H.hecke_matrix(3)
[-1 -2  2  0]
[ 0 -3  2 -2]
[ 2 -4  3 -2]
[ 2 -2  0  1]
```

```
>>> from sage.all import *
>>> H = J0(Integer(23)).integral_homology()
>>> H.hecke_matrix(Integer(3))
[-1 -2  2  0]
[ 0 -3  2 -2]
[ 2 -4  3 -2]
[ 2 -2  0  1]
```

The matrix is over the coefficient ring:

```
sage: J = J0(23)
sage: J.homology(QQ[I]).hecke_matrix(3).parent()
Full MatrixSpace of 4 by 4 dense matrices over
 Number Field in I with defining polynomial x^2 + 1 with I = 1*I
```

```
>>> from sage.all import *
>>> J = J0(Integer(23))
>>> J.homology(QQ[I]).hecke_matrix(Integer(3)).parent()
Full MatrixSpace of 4 by 4 dense matrices over
 Number Field in I with defining polynomial x^2 + 1 with I = 1*I
```

rank()

Return the rank as a module or vector space of this homology group.

EXAMPLES:

```
sage: H = J0(5077).homology(); H
Integral Homology of Abelian variety J0(5077) of dimension 422
sage: H.rank()
844
```

```
>>> from sage.all import *
>>> H = J0(Integer(5077)).homology(); H
Integral Homology of Abelian variety J0(5077) of dimension 422
>>> H.rank()
844
```

submodule(*U*, *check=True*)

Return the submodule of this homology group given by U, which should be a submodule of the free module associated to this homology group.

INPUT:

- `U` – submodule of ambient free module (or something that defines one)

- `check` – currently ignored

> **Note**
>
> We do *not* check that U is invariant under all Hecke operators.

EXAMPLES:

```
sage: H = J0(23).homology(); H
Integral Homology of Abelian variety J0(23) of dimension 2
sage: F = H.free_module()
sage: U = F.span([[1,2,3,4]])
sage: M = H.submodule(U); M
Submodule of rank 1 of Integral Homology of Abelian variety J0(23) of␣
↪dimension 2
```

```
>>> from sage.all import *
>>> H = J0(Integer(23)).homology(); H
Integral Homology of Abelian variety J0(23) of dimension 2
>>> F = H.free_module()
>>> U = F.span([[Integer(1),Integer(2),Integer(3),Integer(4)]])
>>> M = H.submodule(U); M
Submodule of rank 1 of Integral Homology of Abelian variety J0(23) of␣
↪dimension 2
```

Note that the submodule command doesn't actually check that the object defined is a homology group or is invariant under the Hecke operators. For example, the fairly random M that we just defined is not invariant under the Hecke operators, so it is not a Hecke submodule - it is only a $\mathbf{Z}$-submodule.

```
sage: M.hecke_matrix(3)
Traceback (most recent call last):
...
ArithmeticError: subspace is not invariant under matrix
```

```
>>> from sage.all import *
>>> M.hecke_matrix(Integer(3))
Traceback (most recent call last):
...
ArithmeticError: subspace is not invariant under matrix
```

class sage.modular.abvar.homology.**Homology_over_base**(*abvar*, *base_ring*)

Bases: *Homology_abvar*

The homology over a modular abelian variety over an arbitrary base commutative ring (not $\mathbf{Z}$ or $\mathbf{Q}$).

hecke_matrix(*n*)

Return the matrix of the n-th Hecke operator acting on this homology group.

EXAMPLES:

```
sage: t = J1(13).homology(GF(3)).hecke_matrix(3); t
[1 2 2 1]
[1 0 2 0]
```

(continues on next page)

(continued from previous page)

```
[0 0 0 1]
[0 0 2 1]
sage: t.base_ring()
Finite Field of size 3
```

```
>>> from sage.all import *
>>> t = J1(Integer(13)).homology(GF(Integer(3))).hecke_matrix(Integer(3)); t
[1 2 2 1]
[1 0 2 0]
[0 0 0 1]
[0 0 2 1]
>>> t.base_ring()
Finite Field of size 3
```

class sage.modular.abvar.homology.**Homology_submodule**(*ambient*, *submodule*)

Bases: *Homology*

A submodule of the homology of a modular abelian variety.

ambient_hecke_module()

Return the ambient Hecke module that this homology is contained in.

EXAMPLES:

```
sage: H = J0(48).homology(); H
Integral Homology of Abelian variety J0(48) of dimension 3
sage: d = H.decomposition(); d
[Submodule of rank 2 of Integral Homology of Abelian variety J0(48) of␣
↪dimension 3,
 Submodule of rank 4 of Integral Homology of Abelian variety J0(48) of␣
↪dimension 3]
sage: d[0].ambient_hecke_module()
Integral Homology of Abelian variety J0(48) of dimension 3
```

```
>>> from sage.all import *
>>> H = J0(Integer(48)).homology(); H
Integral Homology of Abelian variety J0(48) of dimension 3
>>> d = H.decomposition(); d
[Submodule of rank 2 of Integral Homology of Abelian variety J0(48) of␣
↪dimension 3,
 Submodule of rank 4 of Integral Homology of Abelian variety J0(48) of␣
↪dimension 3]
>>> d[Integer(0)].ambient_hecke_module()
Integral Homology of Abelian variety J0(48) of dimension 3
```

free_module()

Return the underlying free module of the homology group.

EXAMPLES:

```
sage: H = J0(48).homology()
sage: K = H.decomposition()[1]; K
Submodule of rank 4 of Integral Homology of Abelian variety J0(48) of␣
```

(continues on next page)

(continued from previous page)

```
↪dimension 3
sage: K.free_module()
Free module of degree 6 and rank 4 over Integer Ring
Echelon basis matrix:
[ 1  0  0  0  0  0]
[ 0  1  0  0  1 -1]
[ 0  0  1  0 -1  1]
[ 0  0  0  1  0 -1]
```

```
>>> from sage.all import *
>>> H = J0(Integer(48)).homology()
>>> K = H.decomposition()[Integer(1)]; K
Submodule of rank 4 of Integral Homology of Abelian variety J0(48) of␣
↪dimension 3
>>> K.free_module()
Free module of degree 6 and rank 4 over Integer Ring
Echelon basis matrix:
[ 1  0  0  0  0  0]
[ 0  1  0  0  1 -1]
[ 0  0  1  0 -1  1]
[ 0  0  0  1  0 -1]
```

hecke_bound()

Return a bound on the number of Hecke operators needed to generate the Hecke algebra acting on this homology group.

EXAMPLES:

```
sage: d = J0(43).homology().decomposition(2); d
[Submodule of rank 2 of Integral Homology of Abelian variety J0(43) of␣
↪dimension 3,
 Submodule of rank 4 of Integral Homology of Abelian variety J0(43) of␣
↪dimension 3]
```

```
>>> from sage.all import *
>>> d = J0(Integer(43)).homology().decomposition(Integer(2)); d
[Submodule of rank 2 of Integral Homology of Abelian variety J0(43) of␣
↪dimension 3,
 Submodule of rank 4 of Integral Homology of Abelian variety J0(43) of␣
↪dimension 3]
```

Because the first factor has dimension 2 it corresponds to an elliptic curve, so we have a Hecke bound of 1.

```
sage: d[0].hecke_bound()
1
sage: d[1].hecke_bound()
8
```

```
>>> from sage.all import *
>>> d[Integer(0)].hecke_bound()
1
```

(continues on next page)

(continued from previous page)

```
>>> d[Integer(1)].hecke_bound()
8
```

hecke_matrix(*n*)

Return the matrix of the n-th Hecke operator acting on this homology group.

EXAMPLES:

```
sage: d = J0(125).homology(GF(17)).decomposition(2); d
[Submodule of rank 4 of Homology with coefficients in Finite Field of size 17␣
↪of Abelian variety J0(125) of dimension 8,
 Submodule of rank 4 of Homology with coefficients in Finite Field of size 17␣
↪of Abelian variety J0(125) of dimension 8,
 Submodule of rank 8 of Homology with coefficients in Finite Field of size 17␣
↪of Abelian variety J0(125) of dimension 8]
sage: t = d[0].hecke_matrix(17); t
[16 15 15  0]
[ 0  5  0  2]
[ 2  0  5 15]
[ 0 15  0 16]
sage: t.base_ring()
Finite Field of size 17
sage: t.fcp()
(x^2 + 13*x + 16)^2
```

```
>>> from sage.all import *
>>> d = J0(Integer(125)).homology(GF(Integer(17))).decomposition(Integer(2));␣
↪d
[Submodule of rank 4 of Homology with coefficients in Finite Field of size 17␣
↪of Abelian variety J0(125) of dimension 8,
 Submodule of rank 4 of Homology with coefficients in Finite Field of size 17␣
↪of Abelian variety J0(125) of dimension 8,
 Submodule of rank 8 of Homology with coefficients in Finite Field of size 17␣
↪of Abelian variety J0(125) of dimension 8]
>>> t = d[Integer(0)].hecke_matrix(Integer(17)); t
[16 15 15  0]
[ 0  5  0  2]
[ 2  0  5 15]
[ 0 15  0 16]
>>> t.base_ring()
Finite Field of size 17
>>> t.fcp()
(x^2 + 13*x + 16)^2
```

rank()

Return the rank of this homology group.

EXAMPLES:

```
sage: d = J0(43).homology().decomposition(2)
sage: [H.rank() for H in d]
[2, 4]
```

```
>>> from sage.all import *
>>> d = J0(Integer(43)).homology().decomposition(Integer(2))
>>> [H.rank() for H in d]
[2, 4]
```

class sage.modular.abvar.homology.**IntegralHomology**(*abvar*)

Bases: *Homology_abvar*

The integral homology $H_1(A, \mathbf{Z})$ of a modular abelian variety.

hecke_matrix(*n*)

Return the matrix of the n-th Hecke operator acting on this homology group.

EXAMPLES:

```
sage: J0(48).integral_homology().hecke_bound()
16
sage: t = J1(13).integral_homology().hecke_matrix(3); t
[-2  2  2 -2]
[-2  0  2  0]
[ 0  0  0 -2]
[ 0  0  2 -2]
sage: t.base_ring()
Integer Ring
```

```
>>> from sage.all import *
>>> J0(Integer(48)).integral_homology().hecke_bound()
16
>>> t = J1(Integer(13)).integral_homology().hecke_matrix(Integer(3)); t
[-2  2  2 -2]
[-2  0  2  0]
[ 0  0  0 -2]
[ 0  0  2 -2]
>>> t.base_ring()
Integer Ring
```

hecke_polynomial(*n*, *var='x'*)

Return the n-th Hecke polynomial on this integral homology group.

EXAMPLES:

```
sage: f = J0(43).integral_homology().hecke_polynomial(2)
sage: f.base_ring()
Integer Ring
sage: factor(f)
(x + 2)^2 * (x^2 - 2)^2
```

```
>>> from sage.all import *
>>> f = J0(Integer(43)).integral_homology().hecke_polynomial(Integer(2))
>>> f.base_ring()
Integer Ring
>>> factor(f)
(x + 2)^2 * (x^2 - 2)^2
```

class sage.modular.abvar.homology.**RationalHomology**(*abvar*)

Bases: *Homology_abvar*

The rational homology $H_1(A, \mathbf{Q})$ of a modular abelian variety.

hecke_matrix(*n*)

Return the matrix of the n-th Hecke operator acting on this homology group.

EXAMPLES:

```
sage: t = J1(13).homology(QQ).hecke_matrix(3); t
[-2  2  2 -2]
[-2  0  2  0]
[ 0  0  0 -2]
[ 0  0  2 -2]
sage: t.base_ring()
Rational Field
sage: t = J1(13).homology(GF(3)).hecke_matrix(3); t
[1 2 2 1]
[1 0 2 0]
[0 0 0 1]
[0 0 2 1]
sage: t.base_ring()
Finite Field of size 3
```

```
>>> from sage.all import *
>>> t = J1(Integer(13)).homology(QQ).hecke_matrix(Integer(3)); t
[-2  2  2 -2]
[-2  0  2  0]
[ 0  0  0 -2]
[ 0  0  2 -2]
>>> t.base_ring()
Rational Field
>>> t = J1(Integer(13)).homology(GF(Integer(3))).hecke_matrix(Integer(3)); t
[1 2 2 1]
[1 0 2 0]
[0 0 0 1]
[0 0 2 1]
>>> t.base_ring()
Finite Field of size 3
```

hecke_polynomial(*n*, *var='x'*)

Return the n-th Hecke polynomial on this rational homology group.

EXAMPLES:

```
sage: f = J0(43).rational_homology().hecke_polynomial(2)
sage: f.base_ring()
Rational Field
sage: factor(f)
(x + 2) * (x^2 - 2)
```

```
>>> from sage.all import *
>>> f = J0(Integer(43)).rational_homology().hecke_polynomial(Integer(2))
```

(continues on next page)

(continued from previous page)

```
>>> f.base_ring()
Rational Field
>>> factor(f)
(x + 2) * (x^2 - 2)
```

CHAPTER

EIGHT

SPACES OF HOMOMORPHISMS BETWEEN MODULAR ABELIAN VARIETIES

EXAMPLES:

First, we consider J0(37). This Jacobian has two simple factors, corresponding to distinct newforms. These two intersect nontrivially in J0(37).

```
sage: J = J0(37)
sage: D = J.decomposition() ; D
[Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37),
 Simple abelian subvariety 37b(1,37) of dimension 1 of J0(37)]
sage: D[0].intersection(D[1])
(Finite subgroup with invariants [2, 2] over QQ of
  Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37),
 Simple abelian subvariety of dimension 0 of J0(37))
```

```
>>> from sage.all import *
>>> J = J0(Integer(37))
>>> D = J.decomposition() ; D
[Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37),
 Simple abelian subvariety 37b(1,37) of dimension 1 of J0(37)]
>>> D[Integer(0)].intersection(D[Integer(1)])
(Finite subgroup with invariants [2, 2] over QQ of
  Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37),
 Simple abelian subvariety of dimension 0 of J0(37))
```

As an abstract product, since these newforms are distinct, the corresponding simple abelian varieties are not isogenous, and so there are no maps between them. The endomorphism ring of the corresponding product is thus isomorphic to the direct sum of the endomorphism rings for each factor. Since the factors correspond to abelian varieties of dimension 1, these endomorphism rings are each isomorphic to ZZ.

```
sage: Hom(D[0],D[1]).gens()
()
sage: A = D[0] * D[1] ; A
Abelian subvariety of dimension 2 of J0(37) x J0(37)
sage: A.endomorphism_ring().gens()
(Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(37) x J0(37),
 Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(37) x J0(37))
sage: [ x.matrix() for x in A.endomorphism_ring().gens() ]
[
[1 0 0 0]  [0 0 0 0]
```

(continues on next page)

(continued from previous page)

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

```
>>> from sage.all import *
>>> Hom(D[Integer(0)],D[Integer(1)]).gens()
()
>>> A = D[Integer(0)] * D[Integer(1)] ; A
Abelian subvariety of dimension 2 of J0(37) x J0(37)
>>> A.endomorphism_ring().gens()
(Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(37) x J0(37),
 Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(37) x J0(37))
>>> [ x.matrix() for x in A.endomorphism_ring().gens() ]
[
[1 0 0 0]  [0 0 0 0]
[0 1 0 0]  [0 0 0 0]
[0 0 0 0]  [0 0 1 0]
[0 0 0 0], [0 0 0 1]
]
```

However, these two newforms have a congruence between them modulo 2, which gives rise to interesting endomorphisms of J0(37).

```
sage: E = J.endomorphism_ring()
sage: E.gens()
(Abelian variety endomorphism of Abelian variety J0(37) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(37) of dimension 2)
sage: [ x.matrix() for x in E.gens() ]
[
[1 0 0 0]  [ 0  1  1 -1]
[0 1 0 0]  [ 1  0  1  0]
[0 0 1 0]  [ 0  0 -1  1]
[0 0 0 1], [ 0  0  0  1]
]
sage: (-1*E.gens()[0] + E.gens()[1]).matrix()
[-1  1  1 -1]
[ 1 -1  1  0]
[ 0  0 -2  1]
[ 0  0  0  0]
```

```
>>> from sage.all import *
>>> E = J.endomorphism_ring()
>>> E.gens()
(Abelian variety endomorphism of Abelian variety J0(37) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(37) of dimension 2)
>>> [ x.matrix() for x in E.gens() ]
[
[1 0 0 0]  [ 0  1  1 -1]
[0 1 0 0]  [ 1  0  1  0]
[0 0 1 0]  [ 0  0 -1  1]
[0 0 0 1], [ 0  0  0  1]
```

(continues on next page)

(continued from previous page)

```
]
>>> (-Integer(1)*E.gens()[Integer(0)] + E.gens()[Integer(1)]).matrix()
[-1  1  1 -1]
[ 1 -1  1  0]
[ 0  0 -2  1]
[ 0  0  0  0]
```

Of course, these endomorphisms will be reflected in the Hecke algebra, which is in fact the full endomorphism ring of J0(37) in this case:

```
sage: J.hecke_operator(2).matrix()
[-1  1  1 -1]
[ 1 -1  1  0]
[ 0  0 -2  1]
[ 0  0  0  0]
sage: T = E.image_of_hecke_algebra()
sage: T.gens()
(Abelian variety endomorphism of Abelian variety J0(37) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(37) of dimension 2)
sage: [ x.matrix() for x in T.gens() ]
[
[1 0 0 0]  [ 0  1  1 -1]
[0 1 0 0]  [ 1  0  1  0]
[0 0 1 0]  [ 0  0 -1  1]
[0 0 0 1], [ 0  0  0  1]
]
sage: T.index_in(E)
1
```

```
>>> from sage.all import *
>>> J.hecke_operator(Integer(2)).matrix()
[-1  1  1 -1]
[ 1 -1  1  0]
[ 0  0 -2  1]
[ 0  0  0  0]
>>> T = E.image_of_hecke_algebra()
>>> T.gens()
(Abelian variety endomorphism of Abelian variety J0(37) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(37) of dimension 2)
>>> [ x.matrix() for x in T.gens() ]
[
[1 0 0 0]  [ 0  1  1 -1]
[0 1 0 0]  [ 1  0  1  0]
[0 0 1 0]  [ 0  0 -1  1]
[0 0 0 1], [ 0  0  0  1]
]
>>> T.index_in(E)
1
```

Next, we consider J0(33). In this case, we have both oldforms and newforms. There are two copies of J0(11), one for each degeneracy map from J0(11) to J0(33). There is also one newform at level 33. The images of the two degeneracy maps are, of course, isogenous.

```
sage: J = J0(33)
sage: D = J.decomposition()
sage: D
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)]
sage: Hom(D[0],D[1]).gens()
(Abelian variety morphism:
  From: Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
  To:   Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),)
sage: Hom(D[0],D[1]).gens()[0].matrix()
[ 0  1]
[-1  0]
```

```
>>> from sage.all import *
>>> J = J0(Integer(33))
>>> D = J.decomposition()
>>> D
[Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33),
 Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),
 Simple abelian subvariety 33a(1,33) of dimension 1 of J0(33)]
>>> Hom(D[Integer(0)],D[Integer(1)]).gens()
(Abelian variety morphism:
  From: Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
  To:   Simple abelian subvariety 11a(3,33) of dimension 1 of J0(33),)
>>> Hom(D[Integer(0)],D[Integer(1)]).gens()[Integer(0)].matrix()
[ 0  1]
[-1  0]
```

Then this gives that the component corresponding to the sum of the oldforms will have a rank 4 endomorphism ring. We also have a rank one endomorphism ring for the newform 33a (since it is again 1-dimensional), which gives a rank 5 endomorphism ring for J0(33).

```
sage: DD = J.decomposition(simple=False) ; DD
[Abelian subvariety of dimension 2 of J0(33),
 Abelian subvariety of dimension 1 of J0(33)]
sage: A, B = DD
sage: A == D[0] + D[1]
True
sage: A.endomorphism_ring().gens()
(Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(33),
 Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(33),
 Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(33),
 Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(33))
sage: B.endomorphism_ring().gens()
(Abelian variety endomorphism of Abelian subvariety of dimension 1 of J0(33),)
sage: E = J.endomorphism_ring() ; E.gens()  # long time (3s on sage.math, 2011)
(Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3)
```

```
>>> from sage.all import *
>>> DD = J.decomposition(simple=False) ; DD
[Abelian subvariety of dimension 2 of J0(33),
 Abelian subvariety of dimension 1 of J0(33)]
>>> A, B = DD
>>> A == D[Integer(0)] + D[Integer(1)]
True
>>> A.endomorphism_ring().gens()
(Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(33),
 Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(33),
 Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(33),
 Abelian variety endomorphism of Abelian subvariety of dimension 2 of J0(33))
>>> B.endomorphism_ring().gens()
(Abelian variety endomorphism of Abelian subvariety of dimension 1 of J0(33),)
>>> E = J.endomorphism_ring() ; E.gens()  # long time (3s on sage.math, 2011)
(Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3)
```

In this case, the image of the Hecke algebra will only have rank 3, so that it is of infinite index in the full endomorphism ring. However, if we call this image T, we can still ask about the index of T in its saturation, which is 1 in this case.

```
sage: # long time
sage: T = E.image_of_hecke_algebra()
sage: T.gens()
(Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3)
sage: T.index_in(E)
+Infinity
sage: T.index_in_saturation()
1
```

```
>>> from sage.all import *
>>> # long time
>>> T = E.image_of_hecke_algebra()
>>> T.gens()
(Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3)
>>> T.index_in(E)
+Infinity
>>> T.index_in_saturation()
1
```

AUTHORS:

- William Stein (2007-03)
- Craig Citro, Robert Bradshaw (2008-03): Rewrote with modabvar overhaul

class sage.modular.abvar.homspace.**EndomorphismSubring**(*A*, *gens=None*, *category=None*)

Bases: *Homspace*

A subring of the endomorphism ring.

INPUT:

- `A` – an abelian variety
- `gens` – (default: `None`) if given should be a tuple of the generators as matrices

EXAMPLES:

```
sage: J0(23).endomorphism_ring()
Endomorphism ring of Abelian variety J0(23) of dimension 2
sage: sage.modular.abvar.homspace.EndomorphismSubring(J0(25))
Endomorphism ring of Abelian variety J0(25) of dimension 0
sage: E = J0(11).endomorphism_ring()
sage: type(E)
<class 'sage.modular.abvar.homspace.EndomorphismSubring_with_category'>
sage: E.homset_category()
Category of modular abelian varieties over Rational Field
sage: E.category()
Category of endsets of modular abelian varieties over Rational Field
sage: E in Rings()
True
sage: TestSuite(E).run(skip=["_test_prod"])
```

```
>>> from sage.all import *
>>> J0(Integer(23)).endomorphism_ring()
Endomorphism ring of Abelian variety J0(23) of dimension 2
>>> sage.modular.abvar.homspace.EndomorphismSubring(J0(Integer(25)))
Endomorphism ring of Abelian variety J0(25) of dimension 0
>>> E = J0(Integer(11)).endomorphism_ring()
>>> type(E)
<class 'sage.modular.abvar.homspace.EndomorphismSubring_with_category'>
>>> E.homset_category()
Category of modular abelian varieties over Rational Field
>>> E.category()
Category of endsets of modular abelian varieties over Rational Field
>>> E in Rings()
True
>>> TestSuite(E).run(skip=["_test_prod"])
```

abelian_variety()

Return the abelian variety that this endomorphism ring is attached to.

EXAMPLES:

```
sage: J0(11).endomorphism_ring().abelian_variety()
Abelian variety J0(11) of dimension 1
```

```
>>> from sage.all import *
>>> J0(Integer(11)).endomorphism_ring().abelian_variety()
Abelian variety J0(11) of dimension 1
```

discriminant()

Return the discriminant of this ring, which is the discriminant of the trace pairing.

Note

One knows that for modular abelian varieties, the endomorphism ring should be isomorphic to an order in a number field. However, the discriminant returned by this function will be 2^n (n = self.dimension()) times the discriminant of that order, since the elements are represented as 2d x 2d matrices. Notice, for example, that the case of a one dimensional abelian variety, whose endomorphism ring must be ZZ, has discriminant 2, as in the example below.

EXAMPLES:

```
sage: J0(33).endomorphism_ring().discriminant()
-64800
sage: J0(46).endomorphism_ring().discriminant()  # long time (6s on sage.math,
↪ 2011)
24200000000
sage: J0(11).endomorphism_ring().discriminant()
2
```

```
>>> from sage.all import *
>>> J0(Integer(33)).endomorphism_ring().discriminant()
-64800
>>> J0(Integer(46)).endomorphism_ring().discriminant()  # long time (6s on␣
↪sage.math, 2011)
24200000000
>>> J0(Integer(11)).endomorphism_ring().discriminant()
2
```

image_of_hecke_algebra(*check_every=1*)

Compute the image of the Hecke algebra inside this endomorphism subring.

We simply calculate Hecke operators up to the Sturm bound, and look at the submodule spanned by them. While computing, we can check to see if the submodule spanned so far is saturated and of maximal dimension, in which case we may be done. The optional argument check_every determines how many Hecke operators we add in before checking to see if this condition is met.

INPUT:

- `check_every` – integer (default: 1); if this integer is positive, this integer determines how many Hecke operators we add in before checking to see if the submodule spanned so far is maximal and saturated

OUTPUT: the image of the Hecke algebra as a subring of `self`

EXAMPLES:

```
sage: E = J0(33).endomorphism_ring()
sage: E.image_of_hecke_algebra()
Subring of endomorphism ring of Abelian variety J0(33) of dimension 3
sage: E.image_of_hecke_algebra().gens()
(Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3)
sage: [ x.matrix() for x in E.image_of_hecke_algebra().gens() ]
[
[1 0 0 0 0 0]  [ 0  2  0 -1  1 -1]  [ 0  0  1 -1  1 -1]
[0 1 0 0 0 0]  [-1 -2  2 -1  2 -1]  [ 0 -1  1  0  1 -1]
```

(continues on next page)

(continued from previous page)

```
[0 0 1 0 0 0]  [ 0  0  1 -1  3 -1]  [ 0  0  1  0  2 -2]
[0 0 0 1 0 0]  [-2  2  0  1  1 -1]  [-2  0  1  1  1 -1]
[0 0 0 0 1 0]  [-1  1  0  2  0 -3]  [-1  0  1  1  0 -1]
[0 0 0 0 0 1], [-1  1 -1  1  1 -2], [-1  0  0  1  0 -1]
]
sage: J0(33).hecke_operator(2).matrix()
[-1  0  1 -1  1 -1]
[ 0 -2  1  0  1 -1]
[ 0  0  0  0  2 -2]
[-2  0  1  0  1 -1]
[-1  0  1  1 -1 -1]
[-1  0  0  1  0 -2]
```

```
>>> from sage.all import *
>>> E = J0(Integer(33)).endomorphism_ring()
>>> E.image_of_hecke_algebra()
Subring of endomorphism ring of Abelian variety J0(33) of dimension 3
>>> E.image_of_hecke_algebra().gens()
(Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3,
 Abelian variety endomorphism of Abelian variety J0(33) of dimension 3)
>>> [ x.matrix() for x in E.image_of_hecke_algebra().gens() ]
[
[1 0 0 0 0 0]  [ 0  2  0 -1  1 -1]  [ 0  0  1 -1  1 -1]
[0 1 0 0 0 0]  [-1 -2  2 -1  2 -1]  [ 0 -1  1  0  1 -1]
[0 0 1 0 0 0]  [ 0  0  1 -1  3 -1]  [ 0  0  1  0  2 -2]
[0 0 0 1 0 0]  [-2  2  0  1  1 -1]  [-2  0  1  1  1 -1]
[0 0 0 0 1 0]  [-1  1  0  2  0 -3]  [-1  0  1  1  0 -1]
[0 0 0 0 0 1], [-1  1 -1  1  1 -2], [-1  0  0  1  0 -1]
]
>>> J0(Integer(33)).hecke_operator(Integer(2)).matrix()
[-1  0  1 -1  1 -1]
[ 0 -2  1  0  1 -1]
[ 0  0  0  0  2 -2]
[-2  0  1  0  1 -1]
[-1  0  1  1 -1 -1]
[-1  0  0  1  0 -2]
```

index_in(*other*, *check=True*)

Return the index of `self` in `other`.

INPUT:

- `other` – another endomorphism subring of the same abelian variety
- `check` – boolean (default: `True`); whether to do some type and other consistency checks

EXAMPLES:

```
sage: R = J0(33).endomorphism_ring()
sage: R.index_in(R)
1
sage: J = J0(37) ; E = J.endomorphism_ring() ; T = E.image_of_hecke_algebra()
sage: T.index_in(E)
```

(continues on next page)

(continued from previous page)

```
1
sage: J = J0(22) ; E = J.endomorphism_ring() ; T = E.image_of_hecke_algebra()
sage: T.index_in(E)
+Infinity
```

```
>>> from sage.all import *
>>> R = J0(Integer(33)).endomorphism_ring()
>>> R.index_in(R)
1
>>> J = J0(Integer(37)) ; E = J.endomorphism_ring() ; T = E.image_of_hecke_
↪algebra()
>>> T.index_in(E)
1
>>> J = J0(Integer(22)) ; E = J.endomorphism_ring() ; T = E.image_of_hecke_
↪algebra()
>>> T.index_in(E)
+Infinity
```

index_in_saturation()

Given a Hecke algebra T, compute its index in its saturation.

EXAMPLES:

```
sage: End(J0(23)).image_of_hecke_algebra().index_in_saturation()
1
sage: End(J0(44)).image_of_hecke_algebra().index_in_saturation()
2
```

```
>>> from sage.all import *
>>> End(J0(Integer(23))).image_of_hecke_algebra().index_in_saturation()
1
>>> End(J0(Integer(44))).image_of_hecke_algebra().index_in_saturation()
2
```

class sage.modular.abvar.homspace.**Homspace**(*domain*, *codomain*, *cat*)

Bases: HomsetWithBase

A space of homomorphisms between two modular abelian varieties.

Element

alias of *Morphism*

calculate_generators()

If generators haven't already been computed, calculate generators for this homspace. If they have been computed, do nothing.

EXAMPLES:

```
sage: E = End(J0(11))
sage: E.calculate_generators()
```

```
>>> from sage.all import *
>>> E = End(J0(Integer(11)))
>>> E.calculate_generators()
```

free_module()

Return this endomorphism ring as a free submodule of a big $\mathbf{Z}^{4nm}$, where n is the dimension of the domain abelian variety and m the dimension of the codomain.

OUTPUT: free module

EXAMPLES:

```
sage: E = Hom(J0(11), J0(22))
sage: E.free_module()
Free module of degree 8 and rank 2 over Integer Ring
Echelon basis matrix:
[ 1  0 -3  1  1  1 -1 -1]
[ 0  1 -3  1  1  1 -1  0]
```

```
>>> from sage.all import *
>>> E = Hom(J0(Integer(11)), J0(Integer(22)))
>>> E.free_module()
Free module of degree 8 and rank 2 over Integer Ring
Echelon basis matrix:
[ 1  0 -3  1  1  1 -1 -1]
[ 0  1 -3  1  1  1 -1  0]
```

gen(*i=0*)

Return i-th generator of `self`.

INPUT:

- `i` – integer

OUTPUT: a morphism

EXAMPLES:

```
sage: E = End(J0(22))
sage: E.gen(0).matrix()
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

```
>>> from sage.all import *
>>> E = End(J0(Integer(22)))
>>> E.gen(Integer(0)).matrix()
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

gens()

Return tuple of generators for this endomorphism ring.

EXAMPLES:

```
sage: E = End(J0(22))
sage: E.gens()
```

(continues on next page)

(continued from previous page)

```
(Abelian variety endomorphism of Abelian variety J0(22) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(22) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(22) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(22) of dimension 2)
```

```
>>> from sage.all import *
>>> E = End(J0(Integer(22)))
>>> E.gens()
(Abelian variety endomorphism of Abelian variety J0(22) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(22) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(22) of dimension 2,
 Abelian variety endomorphism of Abelian variety J0(22) of dimension 2)
```

identity()

Return the identity endomorphism.

EXAMPLES:

```
sage: E = End(J0(11))
sage: E.identity()
Abelian variety endomorphism of Abelian variety J0(11) of dimension 1
sage: E.one()
Abelian variety endomorphism of Abelian variety J0(11) of dimension 1

sage: H = Hom(J0(11), J0(22))
sage: H.identity()
Traceback (most recent call last):
...
TypeError: the identity map is only defined for endomorphisms
```

```
>>> from sage.all import *
>>> E = End(J0(Integer(11)))
>>> E.identity()
Abelian variety endomorphism of Abelian variety J0(11) of dimension 1
>>> E.one()
Abelian variety endomorphism of Abelian variety J0(11) of dimension 1

>>> H = Hom(J0(Integer(11)), J0(Integer(22)))
>>> H.identity()
Traceback (most recent call last):
...
TypeError: the identity map is only defined for endomorphisms
```

matrix_space()

Return the underlying matrix space that we view this endomorphism ring as being embedded into.

EXAMPLES:

```
sage: E = End(J0(22))
sage: E.matrix_space()
Full MatrixSpace of 4 by 4 dense matrices over Integer Ring
```

```
>>> from sage.all import *
>>> E = End(J0(Integer(22)))
>>> E.matrix_space()
Full MatrixSpace of 4 by 4 dense matrices over Integer Ring
```

ngens()

Return number of generators of `self`.

OUTPUT: integer

EXAMPLES:

```
sage: E = End(J0(22))
sage: E.ngens()
4
```

```
>>> from sage.all import *
>>> E = End(J0(Integer(22)))
>>> E.ngens()
4
```

CHAPTER

NINE

HECKE OPERATORS AND MORPHISMS BETWEEN MODULAR ABELIAN VARIETIES

Sage can compute with Hecke operators on modular abelian varieties. A Hecke operator is defined by given a modular abelian variety and an index. Given a Hecke operator, Sage can compute the characteristic polynomial, and the action of the Hecke operator on various homology groups.

AUTHORS:

- William Stein (2007-03)
- Craig Citro (2008-03)

EXAMPLES:

```
sage: A = J0(54)
sage: t5 = A.hecke_operator(5); t5
Hecke operator T_5 on Abelian variety J0(54) of dimension 4
sage: t5.charpoly().factor()
(x - 3) * (x + 3) * x^2
sage: B = A.new_subvariety(); B
Abelian subvariety of dimension 2 of J0(54)
sage: t5 = B.hecke_operator(5); t5
Hecke operator T_5 on Abelian subvariety of dimension 2 of J0(54)
sage: t5.charpoly().factor()
(x - 3) * (x + 3)
sage: t5.action_on_homology().matrix()
[ 0  3  3 -3]
[-3  3  3  0]
[ 3  3  0 -3]
[-3  6  3 -3]
```

```
>>> from sage.all import *
>>> A = J0(Integer(54))
>>> t5 = A.hecke_operator(Integer(5)); t5
Hecke operator T_5 on Abelian variety J0(54) of dimension 4
>>> t5.charpoly().factor()
(x - 3) * (x + 3) * x^2
>>> B = A.new_subvariety(); B
Abelian subvariety of dimension 2 of J0(54)
>>> t5 = B.hecke_operator(Integer(5)); t5
Hecke operator T_5 on Abelian subvariety of dimension 2 of J0(54)
>>> t5.charpoly().factor()
(x - 3) * (x + 3)
```

(continues on next page)

(continued from previous page)

```
>>> t5.action_on_homology().matrix()
[ 0  3  3 -3]
[-3  3  3  0]
[ 3  3  0 -3]
[-3  6  3 -3]
```

class sage.modular.abvar.morphism.**DegeneracyMap**(*parent*, *A*, *t*, *side='left'*)

Bases: *Morphism*

Create the degeneracy map of index t in parent defined by the matrix A.

INPUT:

- `parent` – a space of homomorphisms of abelian varieties
- `A` – a matrix defining self
- `t` – list of indices defining the degeneracy map

EXAMPLES:

```
sage: J0(44).degeneracy_map(11,2)
Degeneracy map from Abelian variety J0(44) of dimension 4 to Abelian variety␣
↪J0(11) of dimension 1 defined by [2]
sage: J0(44)[0].degeneracy_map(88,2)
Degeneracy map from Simple abelian subvariety 11a(1,44) of dimension 1 of J0(44)␣
↪to Abelian variety J0(88) of dimension 9 defined by [2]
```

```
>>> from sage.all import *
>>> J0(Integer(44)).degeneracy_map(Integer(11),Integer(2))
Degeneracy map from Abelian variety J0(44) of dimension 4 to Abelian variety␣
↪J0(11) of dimension 1 defined by [2]
>>> J0(Integer(44))[Integer(0)].degeneracy_map(Integer(88),Integer(2))
Degeneracy map from Simple abelian subvariety 11a(1,44) of dimension 1 of J0(44)␣
↪to Abelian variety J0(88) of dimension 9 defined by [2]
```

t()

Return the list of indices defining `self`.

EXAMPLES:

```
sage: J0(22).degeneracy_map(44).t()
[1]
sage: J = J0(22) * J0(11)
sage: J.degeneracy_map([44,44], [2,1])
Degeneracy map from Abelian variety J0(22) x J0(11) of dimension 3 to Abelian␣
↪variety J0(44) x J0(44) of dimension 8 defined by [2, 1]
sage: J.degeneracy_map([44,44], [2,1]).t()
[2, 1]
```

```
>>> from sage.all import *
>>> J0(Integer(22)).degeneracy_map(Integer(44)).t()
[1]
>>> J = J0(Integer(22)) * J0(Integer(11))
>>> J.degeneracy_map([Integer(44),Integer(44)], [Integer(2),Integer(1)])
```

(continues on next page)

(continued from previous page)

```
Degeneracy map from Abelian variety J0(22) x J0(11) of dimension 3 to Abelian␣
↪variety J0(44) x J0(44) of dimension 8 defined by [2, 1]
>>> J.degeneracy_map([Integer(44),Integer(44)], [Integer(2),Integer(1)]).t()
[2, 1]
```

class sage.modular.abvar.morphism.**HeckeOperator**(*abvar*, *n*, *side='left'*)

Bases: *Morphism*

A Hecke operator acting on a modular abelian variety.

action_on_homology(*R=Integer Ring*)

Return the action of this Hecke operator on the homology $H_1(A; R)$ of this abelian variety with coefficients in R.

EXAMPLES:

```
sage: A = J0(43)
sage: t2 = A.hecke_operator(2); t2
Hecke operator T_2 on Abelian variety J0(43) of dimension 3
sage: h2 = t2.action_on_homology(); h2
Hecke operator T_2 on Integral Homology of Abelian variety J0(43) of␣
↪dimension 3
sage: h2.matrix()
[-2  1  0  0  0  0]
[-1  1  1  0 -1  0]
[-1  0 -1  2 -1  1]
[-1  0  1  1 -1  1]
[ 0 -2  0  2 -2  1]
[ 0 -1  0  1  0 -1]
sage: h2 = t2.action_on_homology(GF(2)); h2
Hecke operator T_2 on Homology with coefficients in Finite Field of size 2 of␣
↪Abelian variety J0(43) of dimension 3
sage: h2.matrix()
[0 1 0 0 0 0]
[1 1 1 0 1 0]
[1 0 1 0 1 1]
[1 0 1 1 1 1]
[0 0 0 0 0 1]
[0 1 0 1 0 1]
```

```
>>> from sage.all import *
>>> A = J0(Integer(43))
>>> t2 = A.hecke_operator(Integer(2)); t2
Hecke operator T_2 on Abelian variety J0(43) of dimension 3
>>> h2 = t2.action_on_homology(); h2
Hecke operator T_2 on Integral Homology of Abelian variety J0(43) of␣
↪dimension 3
>>> h2.matrix()
[-2  1  0  0  0  0]
[-1  1  1  0 -1  0]
[-1  0 -1  2 -1  1]
[-1  0  1  1 -1  1]
[ 0 -2  0  2 -2  1]
```

(continues on next page)

(continued from previous page)

```
[ 0 -1  0  1  0 -1]
>>> h2 = t2.action_on_homology(GF(Integer(2))); h2
Hecke operator T_2 on Homology with coefficients in Finite Field of size 2 of␣
↪Abelian variety J0(43) of dimension 3
>>> h2.matrix()
[0 1 0 0 0 0]
[1 1 1 0 1 0]
[1 0 1 0 1 1]
[1 0 1 1 1 1]
[0 0 0 0 0 1]
[0 1 0 1 0 1]
```

characteristic_polynomial(*var='x'*)

Return the characteristic polynomial of this Hecke operator in the given variable.

INPUT:

- `var` – string (default: `'x'`)

OUTPUT: a polynomial in var over the rational numbers

EXAMPLES:

```
sage: A = J0(43)[1]; A
Simple abelian subvariety 43b(1,43) of dimension 2 of J0(43)
sage: t2 = A.hecke_operator(2); t2
Hecke operator T_2 on Simple abelian subvariety 43b(1,43) of dimension 2 of␣
↪J0(43)
sage: f = t2.characteristic_polynomial(); f
x^2 - 2
sage: f.parent()
Univariate Polynomial Ring in x over Integer Ring
sage: f.factor()
x^2 - 2
sage: t2.characteristic_polynomial('y')
y^2 - 2
```

```
>>> from sage.all import *
>>> A = J0(Integer(43))[Integer(1)]; A
Simple abelian subvariety 43b(1,43) of dimension 2 of J0(43)
>>> t2 = A.hecke_operator(Integer(2)); t2
Hecke operator T_2 on Simple abelian subvariety 43b(1,43) of dimension 2 of␣
↪J0(43)
>>> f = t2.characteristic_polynomial(); f
x^2 - 2
>>> f.parent()
Univariate Polynomial Ring in x over Integer Ring
>>> f.factor()
x^2 - 2
>>> t2.characteristic_polynomial('y')
y^2 - 2
```

charpoly(*var='x'*)

Synonym for `self.characteristic_polynomial(var)`.

INPUT:

- `var` – string (default: `'x'`)

EXAMPLES:

```
sage: A = J1(13)
sage: t2 = A.hecke_operator(2); t2
Hecke operator T_2 on Abelian variety J1(13) of dimension 2
sage: f = t2.charpoly(); f
x^2 + 3*x + 3
sage: f.factor()
x^2 + 3*x + 3
sage: t2.charpoly('y')
y^2 + 3*y + 3
```

```
>>> from sage.all import *
>>> A = J1(Integer(13))
>>> t2 = A.hecke_operator(Integer(2)); t2
Hecke operator T_2 on Abelian variety J1(13) of dimension 2
>>> f = t2.charpoly(); f
x^2 + 3*x + 3
>>> f.factor()
x^2 + 3*x + 3
>>> t2.charpoly('y')
y^2 + 3*y + 3
```

fcp(*var='x'*)

Return the factorization of the characteristic polynomial.

EXAMPLES:

```
sage: t2 = J0(33).hecke_operator(2)
sage: t2.charpoly()
x^3 + 3*x^2 - 4
sage: t2.fcp()
(x - 1) * (x + 2)^2
```

```
>>> from sage.all import *
>>> t2 = J0(Integer(33)).hecke_operator(Integer(2))
>>> t2.charpoly()
x^3 + 3*x^2 - 4
>>> t2.fcp()
(x - 1) * (x + 2)^2
```

index()

Return the index of this Hecke operator. (For example, if this is the operator T_n, then the index is the integer n.)

OUTPUT:

- `n` – a (Sage) Integer

EXAMPLES:

```
sage: J = J0(15)
sage: t = J.hecke_operator(53)
sage: t
Hecke operator T_53 on Abelian variety J0(15) of dimension 1
sage: t.index()
53
sage: t = J.hecke_operator(54)
sage: t
Hecke operator T_54 on Abelian variety J0(15) of dimension 1
sage: t.index()
54
```

```
>>> from sage.all import *
>>> J = J0(Integer(15))
>>> t = J.hecke_operator(Integer(53))
>>> t
Hecke operator T_53 on Abelian variety J0(15) of dimension 1
>>> t.index()
53
>>> t = J.hecke_operator(Integer(54))
>>> t
Hecke operator T_54 on Abelian variety J0(15) of dimension 1
>>> t.index()
54
```

```
sage: J = J1(12345)
sage: t = J.hecke_operator(997) ; t
Hecke operator T_997 on Abelian variety J1(12345) of dimension 5405473
sage: t.index()
997
sage: type(t.index())
<class 'sage.rings.integer.Integer'>
```

```
>>> from sage.all import *
>>> J = J1(Integer(12345))
>>> t = J.hecke_operator(Integer(997)) ; t
Hecke operator T_997 on Abelian variety J1(12345) of dimension 5405473
>>> t.index()
997
>>> type(t.index())
<class 'sage.rings.integer.Integer'>
```

matrix()

Return the matrix of `self` acting on the homology $H_1(A, ZZ)$ of this abelian variety with coefficients in $\mathbf{Z}$.

EXAMPLES:

```
sage: J0(47).hecke_operator(3).matrix()
[ 0  0  1 -2  1  0 -1  0]
[ 0  0  1  0 -1  0  0  0]
[-1  2  0  0  2 -2  1 -1]
[-2  1  1 -1  3 -1 -1  0]
```

(continues on next page)

(continued from previous page)

```
[-1 -1  1  0  1  0 -1  1]
[-1  0  0 -1  2  0 -1  0]
[-1 -1  2 -2  2  0 -1  0]
[ 0 -1  0  0  1  0 -1  1]
```

```
>>> from sage.all import *
>>> J0(Integer(47)).hecke_operator(Integer(3)).matrix()
[ 0  0  1 -2  1  0 -1  0]
[ 0  0  1  0 -1  0  0  0]
[-1  2  0  0  2 -2  1 -1]
[-2  1  1 -1  3 -1 -1  0]
[-1 -1  1  0  1  0 -1  1]
[-1  0  0 -1  2  0 -1  0]
[-1 -1  2 -2  2  0 -1  0]
[ 0 -1  0  0  1  0 -1  1]
```

```
sage: J0(11).hecke_operator(7).matrix()
[-2  0]
[ 0 -2]
sage: (J0(11) * J0(33)).hecke_operator(7).matrix()
[-2  0  0  0  0  0  0  0]
[ 0 -2  0  0  0  0  0  0]
[ 0  0  0  0  2 -2  2 -2]
[ 0  0  0 -2  2  0  2 -2]
[ 0  0  0  0  2  0  4 -4]
[ 0  0 -4  0  2  2  2 -2]
[ 0  0 -2  0  2  2  0 -2]
[ 0  0 -2  0  0  2  0 -2]
```

```
>>> from sage.all import *
>>> J0(Integer(11)).hecke_operator(Integer(7)).matrix()
[-2  0]
[ 0 -2]
>>> (J0(Integer(11)) * J0(Integer(33))).hecke_operator(Integer(7)).matrix()
[-2  0  0  0  0  0  0  0]
[ 0 -2  0  0  0  0  0  0]
[ 0  0  0  0  2 -2  2 -2]
[ 0  0  0 -2  2  0  2 -2]
[ 0  0  0  0  2  0  4 -4]
[ 0  0 -4  0  2  2  2 -2]
[ 0  0 -2  0  2  2  0 -2]
[ 0  0 -2  0  0  2  0 -2]
```

```
sage: J0(23).hecke_operator(2).matrix()
[ 0  1 -1  0]
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
```

```
>>> from sage.all import *
>>> J0(Integer(23)).hecke_operator(Integer(2)).matrix()
```

(continues on next page)

(continued from previous page)

```
[ 0  1 -1  0]
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
```

n()

Alias for `self.index()`.

EXAMPLES:

```
sage: J = J0(17)
sage: J.hecke_operator(5).n()
5
```

```
>>> from sage.all import *
>>> J = J0(Integer(17))
>>> J.hecke_operator(Integer(5)).n()
5
```

class sage.modular.abvar.morphism.**Morphism**(*parent*, *A*, *copy_matrix=True*, *side='left'*)

Bases: *Morphism_abstract*, MatrixMorphism

restrict_domain(*sub*)

Restrict `self` to the subvariety sub of `self.domain()`.

EXAMPLES:

```
sage: J = J0(37) ; A, B = J.decomposition()
sage: A.lattice().matrix()
[ 1 -1  1  0]
[ 0  0  2 -1]
sage: B.lattice().matrix()
[1 1 1 0]
[0 0 0 1]
sage: T = J.hecke_operator(2) ; T.matrix()
[-1  1  1 -1]
[ 1 -1  1  0]
[ 0  0 -2  1]
[ 0  0  0  0]
sage: T.restrict_domain(A)
Abelian variety morphism:
  From: Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37)
  To:   Abelian variety J0(37) of dimension 2
sage: T.restrict_domain(A).matrix()
[-2  2 -2  0]
[ 0  0 -4  2]
sage: T.restrict_domain(B)
Abelian variety morphism:
  From: Simple abelian subvariety 37b(1,37) of dimension 1 of J0(37)
  To:   Abelian variety J0(37) of dimension 2
sage: T.restrict_domain(B).matrix()
[0 0 0 0]
[0 0 0 0]
```

```
>>> from sage.all import *
>>> J = J0(Integer(37)) ; A, B = J.decomposition()
>>> A.lattice().matrix()
[ 1 -1  1  0]
[ 0  0  2 -1]
>>> B.lattice().matrix()
[1 1 1 0]
[0 0 0 1]
>>> T = J.hecke_operator(Integer(2)) ; T.matrix()
[-1  1  1 -1]
[ 1 -1  1  0]
[ 0  0 -2  1]
[ 0  0  0  0]
>>> T.restrict_domain(A)
Abelian variety morphism:
  From: Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37)
  To:   Abelian variety J0(37) of dimension 2
>>> T.restrict_domain(A).matrix()
[-2  2 -2  0]
[ 0  0 -4  2]
>>> T.restrict_domain(B)
Abelian variety morphism:
  From: Simple abelian subvariety 37b(1,37) of dimension 1 of J0(37)
  To:   Abelian variety J0(37) of dimension 2
>>> T.restrict_domain(B).matrix()
[0 0 0 0]
[0 0 0 0]
```

class sage.modular.abvar.morphism.**Morphism_abstract**(*parent*, *side='left'*)

Bases: MatrixMorphism_abstract

A morphism between modular abelian varieties.

EXAMPLES:

```
sage: t = J0(11).hecke_operator(2)
sage: from sage.modular.abvar.morphism import Morphism
sage: isinstance(t, Morphism)
True
```

```
>>> from sage.all import *
>>> t = J0(Integer(11)).hecke_operator(Integer(2))
>>> from sage.modular.abvar.morphism import Morphism
>>> isinstance(t, Morphism)
True
```

cokernel()

Return the cokernel of `self`.

OUTPUT:

- `A` – an abelian variety (the cokernel)
- `phi` – a quotient map from `self.codomain()` to the cokernel of self

EXAMPLES:

```
sage: t = J0(33).hecke_operator(2)
sage: (t-1).cokernel()
(Abelian subvariety of dimension 1 of J0(33),
 Abelian variety morphism:
  From: Abelian variety J0(33) of dimension 3
  To:   Abelian subvariety of dimension 1 of J0(33))
```

```
>>> from sage.all import *
>>> t = J0(Integer(33)).hecke_operator(Integer(2))
>>> (t-Integer(1)).cokernel()
(Abelian subvariety of dimension 1 of J0(33),
 Abelian variety morphism:
  From: Abelian variety J0(33) of dimension 3
  To:   Abelian subvariety of dimension 1 of J0(33))
```

Projection will always have cokernel zero.

```
sage: J0(37).projection(J0(37)[0]).cokernel()
(Simple abelian subvariety of dimension 0 of J0(37),
 Abelian variety morphism:
  From: Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37)
  To:   Simple abelian subvariety of dimension 0 of J0(37))
```

```
>>> from sage.all import *
>>> J0(Integer(37)).projection(J0(Integer(37))[Integer(0)]).cokernel()
(Simple abelian subvariety of dimension 0 of J0(37),
 Abelian variety morphism:
  From: Simple abelian subvariety 37a(1,37) of dimension 1 of J0(37)
  To:   Simple abelian subvariety of dimension 0 of J0(37))
```

Here we have a nontrivial cokernel of a Hecke operator, as the T_2-eigenvalue for the newform 37b is 0.

```
sage: J0(37).hecke_operator(2).cokernel()
(Abelian subvariety of dimension 1 of J0(37),
 Abelian variety morphism:
  From: Abelian variety J0(37) of dimension 2
  To:   Abelian subvariety of dimension 1 of J0(37))
sage: AbelianVariety('37b').newform().q_expansion(5)
q + q^3 - 2*q^4 + O(q^5)
```

```
>>> from sage.all import *
>>> J0(Integer(37)).hecke_operator(Integer(2)).cokernel()
(Abelian subvariety of dimension 1 of J0(37),
 Abelian variety morphism:
  From: Abelian variety J0(37) of dimension 2
  To:   Abelian subvariety of dimension 1 of J0(37))
>>> AbelianVariety('37b').newform().q_expansion(Integer(5))
q + q^3 - 2*q^4 + O(q^5)
```

complementary_isogeny()

Return the complementary isogeny of `self`.

EXAMPLES:

```
sage: J = J0(43)
sage: A = J[1]
sage: T5 = A.hecke_operator(5)
sage: T5.is_isogeny()
True
sage: T5.complementary_isogeny()
Abelian variety endomorphism of Simple abelian subvariety 43b(1,43) of␣
↪dimension 2 of J0(43)
sage: (T5.complementary_isogeny() * T5).matrix()
[2 0 0 0]
[0 2 0 0]
[0 0 2 0]
[0 0 0 2]
```

```
>>> from sage.all import *
>>> J = J0(Integer(43))
>>> A = J[Integer(1)]
>>> T5 = A.hecke_operator(Integer(5))
>>> T5.is_isogeny()
True
>>> T5.complementary_isogeny()
Abelian variety endomorphism of Simple abelian subvariety 43b(1,43) of␣
↪dimension 2 of J0(43)
>>> (T5.complementary_isogeny() * T5).matrix()
[2 0 0 0]
[0 2 0 0]
[0 0 2 0]
[0 0 0 2]
```

factor_out_component_group()

View `self` as a morphism $f : A \to B$. Then $\ker(f)$ is an extension of an abelian variety C by a finite component group G. This function constructs a morphism g with domain A and codomain Q isogenous to C such that $\ker(g)$ is equal to C.

OUTPUT: a morphism

EXAMPLES:

```
sage: A,B,C = J0(33)
sage: pi = J0(33).projection(A)
sage: pi.kernel()
(Finite subgroup with invariants [5] over QQbar of Abelian variety J0(33) of␣
↪dimension 3,
 Abelian subvariety of dimension 2 of J0(33))
sage: psi = pi.factor_out_component_group()
sage: psi.kernel()
(Finite subgroup with invariants [] over QQbar of Abelian variety J0(33) of␣
↪dimension 3,
 Abelian subvariety of dimension 2 of J0(33))
```

```
>>> from sage.all import *
>>> A,B,C = J0(Integer(33))
>>> pi = J0(Integer(33)).projection(A)
```

(continues on next page)

(continued from previous page)

```
>>> pi.kernel()
(Finite subgroup with invariants [5] over QQbar of Abelian variety J0(33) of␣
↪dimension 3,
 Abelian subvariety of dimension 2 of J0(33))
>>> psi = pi.factor_out_component_group()
>>> psi.kernel()
(Finite subgroup with invariants [] over QQbar of Abelian variety J0(33) of␣
↪dimension 3,
 Abelian subvariety of dimension 2 of J0(33))
```

ALGORITHM: We compute a subgroup G of B so that the composition $h : A \to B \to B/G$ has kernel that contains $A[n]$ and component group isomorphic to $(\mathbf{Z}/n\mathbf{Z})^{2d}$, where d is the dimension of A. Then h factors through multiplication by n, so there is a morphism $g : A \to B/G$ such that $g \circ [n] = h$. Then g is the desired morphism. We give more details below about how to transform this into linear algebra.

image()

Return the image of this morphism.

OUTPUT: an abelian variety

EXAMPLES: We compute the image of projection onto a factor of $J_0(33)$:

```
sage: A,B,C = J0(33)
sage: A
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
sage: f = J0(33).projection(A)
sage: f.image()
Abelian subvariety of dimension 1 of J0(33)
sage: f.image() == A
True
```

```
>>> from sage.all import *
>>> A,B,C = J0(Integer(33))
>>> A
Simple abelian subvariety 11a(1,33) of dimension 1 of J0(33)
>>> f = J0(Integer(33)).projection(A)
>>> f.image()
Abelian subvariety of dimension 1 of J0(33)
>>> f.image() == A
True
```

We compute the image of a Hecke operator:

```
sage: t2 = J0(33).hecke_operator(2); t2.fcp()
(x - 1) * (x + 2)^2
sage: phi = t2 + 2
sage: phi.image()
Abelian subvariety of dimension 1 of J0(33)
```

```
>>> from sage.all import *
>>> t2 = J0(Integer(33)).hecke_operator(Integer(2)); t2.fcp()
(x - 1) * (x + 2)^2
>>> phi = t2 + Integer(2)
```

(continues on next page)

(continued from previous page)

```
>>> phi.image()
Abelian subvariety of dimension 1 of J0(33)
```

The sum of the image and the kernel is the whole space:

```
sage: phi.kernel()[1] + phi.image() == J0(33)
True
```

```
>>> from sage.all import *
>>> phi.kernel()[Integer(1)] + phi.image() == J0(Integer(33))
True
```

is_isogeny()

Return `True` if this morphism is an isogeny of abelian varieties.

EXAMPLES:

```
sage: J = J0(39)
sage: Id = J.hecke_operator(1)
sage: Id.is_isogeny()
True
sage: J.hecke_operator(19).is_isogeny()
False
```

```
>>> from sage.all import *
>>> J = J0(Integer(39))
>>> Id = J.hecke_operator(Integer(1))
>>> Id.is_isogeny()
True
>>> J.hecke_operator(Integer(19)).is_isogeny()
False
```

kernel()

Return the kernel of this morphism.

OUTPUT:

- `G` – a finite group
- `A` – an abelian variety (identity component of the kernel)

EXAMPLES: We compute the kernel of a projection map. Notice that the kernel has a nontrivial abelian variety part.

```
sage: A, B, C = J0(33)
sage: pi = J0(33).projection(B)
sage: pi.kernel()
(Finite subgroup with invariants [20] over QQbar of Abelian variety J0(33) of␣
↪dimension 3,
 Abelian subvariety of dimension 2 of J0(33))
```

```
>>> from sage.all import *
>>> A, B, C = J0(Integer(33))
>>> pi = J0(Integer(33)).projection(B)
```

(continues on next page)

(continued from previous page)

```
>>> pi.kernel()
(Finite subgroup with invariants [20] over QQbar of Abelian variety J0(33) of␣
↪dimension 3,
 Abelian subvariety of dimension 2 of J0(33))
```

We compute the kernels of some Hecke operators:

```
sage: t2 = J0(33).hecke_operator(2)
sage: t2
Hecke operator T_2 on Abelian variety J0(33) of dimension 3
sage: t2.kernel()
(Finite subgroup with invariants [2, 2, 2, 2] over QQ of Abelian variety␣
↪J0(33) of dimension 3,
 Abelian subvariety of dimension 0 of J0(33))
sage: t3 = J0(33).hecke_operator(3)
sage: t3.kernel()
(Finite subgroup with invariants [3, 3] over QQ of Abelian variety J0(33) of␣
↪dimension 3,
 Abelian subvariety of dimension 0 of J0(33))
```

```
>>> from sage.all import *
>>> t2 = J0(Integer(33)).hecke_operator(Integer(2))
>>> t2
Hecke operator T_2 on Abelian variety J0(33) of dimension 3
>>> t2.kernel()
(Finite subgroup with invariants [2, 2, 2, 2] over QQ of Abelian variety␣
↪J0(33) of dimension 3,
 Abelian subvariety of dimension 0 of J0(33))
>>> t3 = J0(Integer(33)).hecke_operator(Integer(3))
>>> t3.kernel()
(Finite subgroup with invariants [3, 3] over QQ of Abelian variety J0(33) of␣
↪dimension 3,
 Abelian subvariety of dimension 0 of J0(33))
```

CHAPTER

TEN

ABELIAN VARIETIES ATTACHED TO NEWFORMS

class sage.modular.abvar.abvar_newform.**ModularAbelianVariety_newform**(*f*, *internal_name=False*)

Bases: *ModularAbelianVariety_modsym_abstract*

A modular abelian variety attached to a specific newform.

endomorphism_ring()

Return the endomorphism ring of this newform abelian variety.

EXAMPLES:

```
sage: A = AbelianVariety('23a')
sage: E = A.endomorphism_ring(); E
Endomorphism ring of Newform abelian subvariety 23a of dimension 2 of J0(23)
```

```
>>> from sage.all import *
>>> A = AbelianVariety('23a')
>>> E = A.endomorphism_ring(); E
Endomorphism ring of Newform abelian subvariety 23a of dimension 2 of J0(23)
```

We display the matrices of these two basis matrices:

```
sage: E.0.matrix()
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
sage: E.1.matrix()
[ 0  1 -1  0]
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
```

```
>>> from sage.all import *
>>> E.gen(0).matrix()
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
>>> E.gen(1).matrix()
[ 0  1 -1  0]
```

(continues on next page)

(continued from previous page)

```
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
```

The result is cached:

```
sage: E is A.endomorphism_ring()
True
```

```
>>> from sage.all import *
>>> E is A.endomorphism_ring()
True
```

factor_number()

Return factor number.

OUTPUT: int

EXAMPLES:

```
sage: A = AbelianVariety('43b')
sage: A.factor_number()
1
```

```
>>> from sage.all import *
>>> A = AbelianVariety('43b')
>>> A.factor_number()
1
```

label()

Return canonical label that defines this newform modular abelian variety.

OUTPUT: string

EXAMPLES:

```
sage: A = AbelianVariety('43b')
sage: A.label()
'43b'
```

```
>>> from sage.all import *
>>> A = AbelianVariety('43b')
>>> A.label()
'43b'
```

newform(*names=None*)

Return the newform that this modular abelian variety is attached to.

EXAMPLES:

```
sage: f = Newform('37a')
sage: A = f.abelian_variety()
sage: A.newform()
q - 2*q^2 - 3*q^3 + 2*q^4 - 2*q^5 + O(q^6)
```

(continues on next page)

(continued from previous page)

```
sage: A.newform() is f
True
```

```
>>> from sage.all import *
>>> f = Newform('37a')
>>> A = f.abelian_variety()
>>> A.newform()
q - 2*q^2 - 3*q^3 + 2*q^4 - 2*q^5 + O(q^6)
>>> A.newform() is f
True
```

If the a variable name has not been specified, we must specify one:

```
sage: A = AbelianVariety('67b')
sage: A.newform()
Traceback (most recent call last):
...
TypeError: You must specify the name of the generator.
sage: A.newform('alpha')
q + alpha*q^2 + (-alpha - 3)*q^3 + (-3*alpha - 3)*q^4 - 3*q^5 + O(q^6)
```

```
>>> from sage.all import *
>>> A = AbelianVariety('67b')
>>> A.newform()
Traceback (most recent call last):
...
TypeError: You must specify the name of the generator.
>>> A.newform('alpha')
q + alpha*q^2 + (-alpha - 3)*q^3 + (-3*alpha - 3)*q^4 - 3*q^5 + O(q^6)
```

If the eigenform is actually over $\mathbf{Q}$ then we don't have to specify the name:

```
sage: A = AbelianVariety('67a')
sage: A.newform()
q + 2*q^2 - 2*q^3 + 2*q^4 + 2*q^5 + O(q^6)
```

```
>>> from sage.all import *
>>> A = AbelianVariety('67a')
>>> A.newform()
q + 2*q^2 - 2*q^3 + 2*q^4 + 2*q^5 + O(q^6)
```

CHAPTER

ELEVEN

L-SERIES OF MODULAR ABELIAN VARIETIES

AUTHOR:

- William Stein (2007-03)

class sage.modular.abvar.lseries.**Lseries**(*abvar*)

Bases: SageObject

Base class for L-series attached to modular abelian varieties.

This is a common base class for complex and p-adic L-series of modular abelian varieties.

abelian_variety()

Return the abelian variety that this L-series is attached to.

OUTPUT: a modular abelian variety

EXAMPLES:

```
sage: J0(11).padic_lseries(7).abelian_variety()
Abelian variety J0(11) of dimension 1
```

```
>>> from sage.all import *
>>> J0(Integer(11)).padic_lseries(Integer(7)).abelian_variety()
Abelian variety J0(11) of dimension 1
```

class sage.modular.abvar.lseries.**Lseries_complex**(*abvar*)

Bases: *Lseries*

A complex L-series attached to a modular abelian variety.

EXAMPLES:

```
sage: A = J0(37)
sage: A.lseries()
Complex L-series attached to Abelian variety J0(37) of dimension 2
```

```
>>> from sage.all import *
>>> A = J0(Integer(37))
>>> A.lseries()
Complex L-series attached to Abelian variety J0(37) of dimension 2
```

lratio()

Return the rational part of this L-function at the central critical value 1.

OUTPUT: a rational number

EXAMPLES:

```
sage: A, B = J0(43).decomposition()
sage: A.lseries().rational_part()
0
sage: B.lseries().rational_part()
2/7
```

```
>>> from sage.all import *
>>> A, B = J0(Integer(43)).decomposition()
>>> A.lseries().rational_part()
0
>>> B.lseries().rational_part()
2/7
```

rational_part()

Return the rational part of this L-function at the central critical value 1.

OUTPUT: a rational number

EXAMPLES:

```
sage: A, B = J0(43).decomposition()
sage: A.lseries().rational_part()
0
sage: B.lseries().rational_part()
2/7
```

```
>>> from sage.all import *
>>> A, B = J0(Integer(43)).decomposition()
>>> A.lseries().rational_part()
0
>>> B.lseries().rational_part()
2/7
```

vanishes_at_1()

Return `True` if $L(1) = 0$ and return `False` otherwise.

OUTPUT: boolean

EXAMPLES:

Numerically, the L-series for $J_0(389)$ appears to vanish at 1. This is confirmed by this algebraic computation:

```
sage: L = J0(389)[0].lseries(); L
Complex L-series attached to Simple abelian subvariety 389a(1,389) of␣
↪dimension 1 of J0(389)
sage: L(1) # long time (2s) abstol 1e-10
-1.33139759782370e-19
sage: L.vanishes_at_1()
True
```

```
>>> from sage.all import *
>>> L = J0(Integer(389))[Integer(0)].lseries(); L
Complex L-series attached to Simple abelian subvariety 389a(1,389) of␣
```

(continues on next page)

(continued from previous page)

```
↪dimension 1 of J0(389)
>>> L(Integer(1)) # long time (2s) abstol 1e-10
-1.33139759782370e-19
>>> L.vanishes_at_1()
True
```

Numerically, one might guess that the L-series for $J_1(23)$ and $J_1(31)$ vanish at 1. This algebraic computation shows otherwise:

```
sage: L = J1(23).lseries(); L
Complex L-series attached to Abelian variety J1(23) of dimension 12
sage: L(1)  # long time (about 3 s)
0.0001295198...
sage: L.vanishes_at_1()
False
sage: abs(L(1, prec=100)- 0.00012951986142702571478817757148) < 1e-32  # long␣
↪time (about 3 s)
True

sage: L = J1(31).lseries(); L
Complex L-series attached to Abelian variety J1(31) of dimension 26
sage: abs(L(1) - 3.45014267547611e-7) < 1e-15  # long time (about 8 s)
True
sage: L.vanishes_at_1()  # long time (about 6 s)
False
```

```
>>> from sage.all import *
>>> L = J1(Integer(23)).lseries(); L
Complex L-series attached to Abelian variety J1(23) of dimension 12
>>> L(Integer(1))  # long time (about 3 s)
0.0001295198...
>>> L.vanishes_at_1()
False
>>> abs(L(Integer(1), prec=Integer(100))- RealNumber('0.
↪00012951986142702571478817757148')) < RealNumber('1e-32')  # long time␣
↪(about 3 s)
True

>>> L = J1(Integer(31)).lseries(); L
Complex L-series attached to Abelian variety J1(31) of dimension 26
>>> abs(L(Integer(1)) - RealNumber('3.45014267547611e-7')) < RealNumber('1e-15
↪')  # long time (about 8 s)
True
>>> L.vanishes_at_1()  # long time (about 6 s)
False
```

class sage.modular.abvar.lseries.**Lseries_padic**(*abvar*, *p*)

Bases: *Lseries*

A p-adic L-series attached to a modular abelian variety.

power_series(*n=2*, *prec=5*)

Return the n-th approximation to this p-adic L-series as a power series in T.

Each coefficient is a p-adic number whose precision is provably correct.

NOTE: This is not yet implemented.

EXAMPLES:

```
sage: L = J0(37)[0].padic_lseries(5)
sage: L.power_series()
Traceback (most recent call last):
...
NotImplementedError
sage: L.power_series(3,7)
Traceback (most recent call last):
...
NotImplementedError
```

```
>>> from sage.all import *
>>> L = J0(Integer(37))[Integer(0)].padic_lseries(Integer(5))
>>> L.power_series()
Traceback (most recent call last):
...
NotImplementedError
>>> L.power_series(Integer(3),Integer(7))
Traceback (most recent call last):
...
NotImplementedError
```

prime()

Return the prime p of this p-adic L-series.

EXAMPLES:

```
sage: J0(11).padic_lseries(7).prime()
7
```

```
>>> from sage.all import *
>>> J0(Integer(11)).padic_lseries(Integer(7)).prime()
7
```

CHAPTER

TWELVE

INDICES AND TABLES

- Index
- Module Index
- Search Page

PYTHON MODULE INDEX

m

INDEX

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

V

Z