Contents

Chapter 1

Classes

1.1 poly.formalsum – formal sum

- **Classes**
  - †**FormalSumContainerInterface**
  - **DictFormalSum**
  - †**ListFormalSum**

The formal sum is mathematically a finite sum of terms, A term consists of two parts: coefficient and base. All coefficients in a formal sum are in a common ring, while bases are arbitrary.

Two formal sums can be added in the following way. If there are terms with common base, they are fused into a new term with the same base and coefficients added.

A coefficient can be looked up from the base. If the specified base does not appear in the formal sum, it is null.

We refer the following for convenience as `terminit`:

terminit :
`terminit` means one of types to initialize dict. The dictionary constructed from it will be considered as a mapping from bases to coefficients.

Note for beginner You may need USE only **DictFormalSum**, but may have to READ the description of **FormalSumContainerInterface** because interface (all method names and their semantics) is defined in it.

1.1.1 FormalSumContainerInterface – interface class

Initialize (Constructor)

Since the interface is an abstract class, do not instantiate.

The interface defines what "formal sum" is. Derived classes must provide the following operations and methods.

Operations

operator	explanation
`f + g`	addition
`f - g`	subtraction
`-f`	negation
`+f`	new copy
`f * a, a * f`	multiplication by scalar `a`
`f == g`	equality
`f != g`	inequality
`f[b]`	get coefficient corresponding to a base `b`
`b in f`	return whether base `b` is in `f`
`len(f)`	number of terms
`hash(f)`	hash

Methods

1.1.1.1 construct_with_default – copy-constructing

construct_with_default(`self`, `maindata`: *terminit*)
→ *FormalSumContainerInterface*

Create a new formal sum of the same class with `self`, with given only the `maindata` and use copy of `self`'s data if necessary.

1.1.1.2 iterterms – iterator of terms

iterterms(`self`) → *iterator*

Return an iterator of the terms.
Each term yielded from iterators is a `(base, coefficient)` pair.

1.1.1.3 itercoefficients – iterator of coefficients

itercoefficients(`self`) → *iterator*

Return an iterator of the coefficients.

1.1.1.4 iterbases – iterator of bases

iterbases(`self`) → *iterator*

Return an iterator of the bases.

1.1.1.5 terms – list of terms

terms(`self`) → *list*

Return a list of the terms.
Each term in returned lists is a `(base, coefficient)` pair.

1.1.1.6 coefficients – list of coefficients

coefficients(`self`) → *list*

Return a list of the coefficients.

1.1.1.7 bases – list of bases

bases(`self`) → ***list***

Return a list of the bases.

1.1.1.8 terms_map – list of terms

terms_map(`self`, `func`: ***function***) → ***FormalSumContainerInterface***

Map on terms, i.e., create a new formal sum by applying `func` to each term.

`func` has to accept two parameters `base` and `coefficient`, then return a new term pair.

1.1.1.9 coefficients_map – list of coefficients

coefficients_map(`self`) → ***FormalSumContainerInterface***

Map on coefficients, i.e., create a new formal sum by applying `func` to each coefficient.

`func` has to accept one parameters `coefficient`, then return a new coefficient.

1.1.1.10 bases_map – list of bases

bases_map(`self`) → ***FormalSumContainerInterface***

Map on bases, i.e., create a new formal sum by applying `func` to each base.

`func` has to accept one parameters `base`, then return a new base.

1.1.2 DictFormalSum – formal sum implemented with dictionary

A formal sum implementation based on `dict`.

This class inherits **FormalSumContainerInterface**. All methods of the interface are implemented.

Initialize (Constructor)

DictFormalSum(`args`: *terminit*, `defaultvalue`: *RingElement*=None) → *DictFormalSum*

See **terminit** for type of `args`. It makes a mapping from bases to coefficients.
The optional argument `defaultvalue` is the default value for `__getitem__`, i.e., if there is no term with the specified base, a look up attempt returns the `defaultvalue`. It is, thus, an element of the ring to which other coefficients belong.

1.1.3 ListFormalSum – formal sum implemented with list

A formal sum implementation based on list.

This class inherits **FormalSumContainerInterface**. All methods of the interface are implemented.

Initialize (Constructor)

ListFormalSum(`args`: *terminit*, `defaultvalue`: *RingElement*=None) → *ListFormalSum*

See **terminit** for type of `args`. It makes a mapping from bases to coefficients.
The optional argument `defaultvalue` is the default value for `__getitem__`, i.e., if there is no term with the specified base, a look up attempt returns the `defaultvalue`. It is, thus, an element of the ring to which other coefficients belong.