

Contents

1	Classes	2
1.1	poly.termorder – 項順序	2
1.1.1	TermOrderInterface – 項順序のインターフェース	3
1.1.1.1	cmp	4
1.1.1.2	format	4
1.1.1.3	leading_coefficient	4
1.1.1.4	leading_term	4
1.1.2	UnivarTermOrder – 一変数多項式に対する項順序	4
1.1.2.1	format	5
1.1.2.2	degree	5
1.1.2.3	tail_degree	5
1.1.3	MultivarTermOrder – 多変数多項式に対する項順序	5
1.1.3.1	format	6
1.1.4	weight_order – 重み付き順序付け	6

Chapter 1

Classes

1.1 poly.termorder – 項順序

- **Classes**
 - † **TermOrderInterface**
 - † **UnivarTermOrder**
 - **MultivarTermOrder**
- **Functions**
 - **weight_order**

1.1.1 TermOrderInterface – 項順序のインターフェース

Initialize (Constructor)

`TermOrderInterface(comparator: function) → TermOrderInterface`

項順序は主に二つの項 (または単項式) の優先順位を決定する関数. 優先順位により, 全ての項は順序付けられる.

より正確に言うと, Python の形式では, 項順序は整数での二つのタプルをとり, そのそれぞれのタプルは項のべき指数を表す. そして組み込み関数の `cmp` のようにただ 0, 1 または -1 を返す.

A `TermOrder` オブジェクトは優先順位関数だけでなく, 次数や主係数などが記された, 多項式のフォーマットされた文字列を返すメソッドも提供.

`comparator` は整数での二つのタプルのようなオブジェクトをとり, それぞれのタプルは項のべき指数を表す. そして組み込み関数 `cmp` のようにただ 0, 1 または -1 を返す.

このクラスは抽象クラスでインスタンスが作られるべきではない. `k` のメソッドは下にオーバーライドされなければならない.

Methods

1.1.1.1 cmp

`cmp(self, left: tuple, right: tuple) → integer`

二つのインデックスタプル `left` と `right` を比較し優先順位を決定.

1.1.1.2 format

`format(self, polynom: polynomial, **keywords: dict)`
→ *string*

多項式 `polynom` のフォーマットされた文字列を返す.

1.1.1.3 leading_coefficient

`leading_coefficient(self, polynom: polynomial) → CommutativeRingElement`

多項式 `polynom` の項順序についての主係数を返す.

1.1.1.4 leading_term

`leading_term(self, polynom: polynomial) → tuple`

多項式 `polynom` の主項を項順序についてのタプル (`degree index`, `coefficient`) として返す.

1.1.2 UnivarTermOrder – 一変数多項式に対する項順序

Initialize (Constructor)

`UnivarTermOrder(comparator: function) → UnivarTermOrder`

一変数多項式に対しては一意的な項順序がある. 次数として知られている.

一変数の場合への特別なことは, べき数はタプルではなく, 単なる整数であるということである. このことから, メソッド signatures もまた `TermOrderInterface` 内の定義から変換する必要があるが, それは容易なため説明は省略.

`comparator` は二つの整数をとり, `cmp` のようにただ 0, 1 または -1 を返すために呼ばれ得る, すなわち, もしそれらが 0 を返す, 最初は 1 より大きい, そしてさもなれば -1. 理論上は期待できる比較関数は `cmp` 関数のみ.

このクラスは `TermOrderInterface` を継承する.

Methods

1.1.2.1 format

```
format(self, polynom: polynomial, varname: string='X', reverse:  
bool=False)  
    → string
```

多項式 `polynom` のフォーマットされた文字列を返す.

- `polynom` は一変数多項式でなければならない
- `varname` は変数名の設定ができる.
- `reverse` は `True` と `False` のどちらかになり得る. もしそれが `True` なら, 項は逆 (降) 順で現れる.

1.1.2.2 degree

```
degree(self, polynom: polynomial) → integer
```

多項式 `polynom` の次数を返す.

1.1.2.3 tail_degree

```
tail_degree(self, polynom: polynomial) → integer
```

`polynom` の全ての項の中での最小次数を返す.

このメソッドは *experimental* です.

1.1.3 MultivarTermOrder – 多変数多項式に対する項順序

Initialize (Constructor)

```
MultivarTermOrder(comparator: function) → MultivarTermOrder
```

このクラスは **TermOrderInterface** を継承する.

Methods

1.1.3.1 format

```
format(self, polynom: polynomial, varname: tuple=None, reverse:
bool=False, **kwds: dict)
    → string
```

多項式 `polynom` のフォーマットされた文字列を返す.

追加の引数である `varnames` は変数名が必要とされる.

- `polynom` は多変数多項式です.
- `varnames` は変数名の列.
- `reverse` は `True` と `False` のどちらかになり得る. もしそれが `True`, 項は逆 (降) 順で現れる.

1.1.4 weight_order – 重み付き順序付け

```
weight_order(weight: sequence, tie_breaker: function=None)
    → function
```

`weight` による重み付き順序の比較関数を返す.

w を `weight` をします. 重み付き順序付けは引数 x と y によって定義され, それらは以下を満たす. もし $w \cdot x < w \cdot y$ なら $x < y$ であり, また $w \cdot x == w \cdot y$ かつ `tie breaker` が $x < y$ と出したら $x < y$

. オプション `tie_breaker` は, もし重み付きベクトルのドット積が引数 `tie` と等しいままなら使われるもう一つの比較関数. もしそのオプションが `None` (初期設定) で, 与えられた引数を順序付けするため `tie breaker` が本当に必要なら, `TypeError` が起こる.

Examples

```
>>> w = termorder.MultivarTermOrder(
...     termorder.weight_order((6, 3, 1), cmp))
>>> w.cmp((1, 0, 0), (0, 1, 2))
1
```

Bibliography