Contents

Chapter 1

Classes

1.1 factor.util – utilities for factorization

- **Classes**
  - **FactoringInteger**
  - **FactoringMethod**

This module uses following type:

factorlist :
`f`actorlist is a list which consists of pairs `(base, index)`. Each pair means $base^{index}$. The product of those terms expresses whole prime factorization.

1.1.1 FactoringInteger – keeping track of factorization

Initialize (Constructor)

FactoringInteger(number: *integer*) → *FactoringInteger*

This is the base class for factoring integers.

number is stored in the attribute **number**. The factors will be stored in the attribute **factors**, and primality of factors will be tracked in the attribute **primality**.

The given number must be a composite number.

Attributes

number :
The composite number.

factors :
Factors known at the time being referred.

primality :
A dictionary of primality information of known factors. `True` if the factor is prime, `False` composite, or `None` undetermined.

Methods

1.1.1.1 getNextTarget – next target

getNextTarget(self, cond: *function*=None) → *integer*

Return the next target which meets cond.

If cond is not specified, then the next target is a composite (or undetermined) factor of **number**.

cond should be a binary predicate whose arguments are base and index.
If there is no target factor, LookupError will be raised.

1.1.1.2 getResult – result of factorization

getResult(self) → factors

number の因数分解をする。

1.1.1.3 register – register a new factor

register(self, divisor: *integer*, isprime: *bool*=None) →

divisor が本当にある数を割るとき、**number** の divisor を記憶する。

その数は divisor により可能な限り割られる。

isprime tells the primality of the divisor (default to undetermined).

1.1.1.4 sortFactors – sort factors

sortFactors(self) →
要素のリストを並べる。

この関数は **getResult** に関係している。

Examples

```
>>> A = factor.util.FactoringInteger(100)
>>> A.getNextTarget()
100
```

```
>>> A.getResult()
[(100, 1)]
>>> A.register(5, True)
>>> A.getResult()
[(5, 2), (4, 1)]
>>> A.sortFactors()
>>> A.getResult()
[(4, 1), (5, 2)
>>> A.primality
{4: None, 5: True}
>>> A.getNextTarget()
4
```

1.1.2 FactoringMethod – method of factorization

Initialize (Constructor)

FactoringMethod() → *FactoringMethod*

Base class of factoring methods.

すべての方法は **factor.methods** で定義されている。implemented as derived classes of this class. この方法は **factor** と呼ぶこともある。　他の方法は

Methods

1.1.2.1 factor – do factorization

factor(self, number: *integer*, return_type: *str*='list', need_sort: *bool*=False)
→ factorlist

与えられた正の整数 number の因数分解を行う。
不履行の場合は **factorlist** を返す。
A keyword option return_type can be as the following:

1. `'list'` for default type (**factorlist**).
2. `'tracker'` for **FactoringInteger**.

Another keyword option need_sort is Boolean: `True` to sort the result. This should be specified with `return_type='list'`.

1.1.2.2 †continue_factor – continue factorization

continue_factor(self, tracker: FactoringInteger, return_type: *str*='tracker', primeq: *func*=primeq)
→ FactoringInteger

Continue factoring of the given tracker and return the result of factorization.

The default returned type is **FactoringInteger**, but if return_type is specified as `'list'` then it returns **factorlist**. The primality is judged by a function specified in primeq optional keyword argument, which default is **primeq**.

1.1.2.3 †find – find a factor

find(self, target: *integer*, **options) → *integer*

target から要素を探す。

この方法は優先されるべきである。または **factor** 法も

1.1.2.4 †generate – generate prime factors

generate(self, target: *integer*, **options) → *integer*

Generate prime factors of the target number with their valuations.

この関数が `(1, 1)` を返したら因数分解は不完全であることを示す。to indicate the factorization is incomplete.
This method has to be overridden, or **factor** method should be overridden not to call this method.

Bibliography