Contents

Chapter 1

Classes

1.1 factor.misc – miscellaneous functions related factoring

- **Functions**
  - allDivisors
  - primeDivisors
  - primePowerTest
  - squarePart
- Classes
  - FactoredInteger

1.1.1 allDivisors – all divisors

allDivisors(`n`: *integer*) → *list*

Return all factors divide `n` as a list.

1.1.2 primeDivisors – prime divisors

primeDivisors(`n`: *integer*) → *list*

Return all prime factors divide `n` as a list.

1.1.3 primePowerTest – prime power test

primePowerTest(n: ***integer***) → (***integer***, ***integer***)

Judge whether `n` is of the form p^k with a prime p or not. If it is true, then `(p, k)` will be returned, otherwise `(n, 0)`.

This function is based on Algo. 1.7.5 in [1].

1.1.4 squarePart – square part

squarePart(n: ***integer***) → ***integer***

Return the largest integer whose square divides `n`.

Examples

```
>>> factor.misc.allDivisors(1001)
[1, 7, 11, 13L, 77, 91L, 143L, 1001L]
>>> factor.misc.primeDivisors(100)
[2, 5]
>>> factor.misc.primePowerTest(128)
(2, 7)
>>> factor.misc.squarePart(128)
8L
```

1.1.5 FactoredInteger – integer with its factorization

Initialize (Constructor)

FactoredInteger(integer: *integer*, factors: *dict*={})
→ ***FactoredInteger***

Integer with its factorization information.

If `factors` is given, it is a dict of type `prime:exponent` and the product of $prime^{exponent}$ is equal to the `integer`. Otherwise, factorization is carried out in initialization.

from_partial_factorization(cls, integer: *integer*, partial: *dict*)
→ ***FactoredInteger***

A class method to create a new **FactoredInteger** object from partial factorization information partial.

Operations

operator	explanation
`F * G`	multiplication (other operand can be an int)
`F ** n`	powering
`F == G`	equal
`F != G`	not equal
`F % G`	remainder (the result is an int)
`F // G`	same as **exact_division** method
`str(F)`	string
`int(F)`	convert to Python integer (forgetting factorization)

Methods

1.1.5.1 is_divisible_by

is_divisible_by(self, other: *integer*/FactoredInteger)
→ *bool*

Return True if other divides self.

1.1.5.2 exact_division

exact_division(self, other: *integer*/FactoredInteger)
→ FactoredInteger

Divide by other. The other must divide self.

1.1.5.3 divisors

divisors(self) → *list*

Return all divisors as a list.

1.1.5.4 proper_divisors

proper_divisors(self) → *list*

Return all proper divisors (i.e. divisors excluding 1 and self) as a list.

1.1.5.5 prime_divisors

prime_divisors(self) → *list*

Return all prime divisors as a list.

1.1.5.6 square_part

square_part(self, asfactored: *bool*=False) → *integer*/FactoredInteger object

Return the largest integer whose square divides self.

If an optional argument asfactored is true, then the result is also a **FactoredInteger object**. (default is False)

1.1.5.7 squarefree_part

squarefree_part(self, asfactored: *bool*=False) → *integer*/FactoredInteger object

Return the largest squarefree integer which divides self.

If an optional argument asfactored is true, then the result is also a **FactoredInteger object** object. (default is False)

1.1.5.8 copy

copy(self) → FactoredInteger object

Return a copy of the object.

Bibliography

[1] Henri Cohen. *A Course in Computational Algebraic Number Theory.* GTM138. Springer, 1st. edition, 1993.