Contents

Chapter 1

Classes

1.1 rational – integer and rational number

rational module provides integer and rational numbers, as class Rational, Integer, RationalField, and IntegerRing.

- **Classes**
  - **Integer**
  - **IntegerRing**
  - **Rational**
  - **RationalField**

This module also provides following constants:

theIntegerRing :
: `theIntegerRing` is represents the ring of rational integers. An instance of **IntegerRing**.

theRationalField :
: `theRationalField` is represents the field of rational numbers. An instance of **RationalField**.

1.1.1 Integer – integer

Integer is a class of integer. Since 'int' and 'long' do not return rational for division, it is needed to create a new class.

This class is a subclass of **CommutativeRingElement** and long.

Initialize (Constructor)

Integer(`integer`: ***integer***) → ***Integer***

Construct a Integer object. If argument is omitted, the value becomes 0.

Methods

1.1.1.1 getRing – get ring object

getRing(self) → ***IntegerRing***

Return an IntegerRing object.

1.1.1.2 actAdditive – addition of binary addition chain

actAdditive(self, other: ***integer***) → ***Integer***

Act on other additively, i.e. n is expanded to n time additions of `other`. Naively, it is:

```
return sum([+other for _ in range(self)])
```

but, here we use a binary addition chain.

1.1.1.3 actMultiplicative – multiplication of binary addition chain

actMultiplicative(self, other: ***integer***) → ***Integer***

Act on other multiplicatively, i.e. n is expanded to n time multiplications of `other`. Naively, it is:

```
return reduce(lambda x,y: x*y, [+other for _ in range(self)])
```

but, here we use a binary addition chain.

1.1.2 IntegerRing – integer ring

The class is for the ring of rational integers.

This class is a subclass of **CommutativeRing**.

Initialize (Constructor)

IntegerRing() → *IntegerRing*

Create an instance of IntegerRing. You may not want to create an instance, since there is already theIntegerRing.

Attribute

zero :
It expresses the additive unit 0. (read only)

one :
It expresses the multiplicative unit 1. (read only)

Operations

operator	explanation
`x in Z`	return whether an element is in or not.
`repr(Z)`	return representation string.
`str(Z)`	return string.

Methods

1.1.2.1 createElement – create Integer object

createElement(`self`, `seed`: ***integer***) → ***Integer***

Return an Integer object with `seed`.
`seed` must be int, long or rational.Integer.

1.1.2.2 gcd – greatest common divisor

gcd(`self`, `n`: ***integer***, `m`: ***integer***) → ***Integer***

Return the greatest common divisor of given 2 integers.

1.1.2.3 extgcd – extended GCD

extgcd(`self`, `n`: ***integer***, `m`: ***integer***) → ***Integer***

Return a tuple (u, v, d); they are the greatest common divisor d of two given integers `n` and `m` and u, v such that $d = \mathtt{n}u + \mathtt{m}v$.

1.1.2.4 lcm – lowest common multiplier

lcm(`self`, `n`: ***integer***, `m`: ***integer***) → ***Integer***

Return the lowest common multiple of given 2 integers. If both are zero, it raises an exception.

1.1.2.5 getQuotientField – get rational field object

getQuotientField(`self`) → ***RationalField***

Return the rational field (**RationalField**).

1.1.2.6 issubring – subring test

issubring(`self`, `other`: **Ring**) → ***bool***

Report whether another ring contains the integer ring as subring.

If other is also the integer ring, the output is True. In other cases it depends on the implementation of another ring's issuperring method.

1.1.2.7 issuperring – superring test

issuperring(`self`, `other`: **Ring**) $\rightarrow$ ***bool***

Report whether the integer ring contains another ring as subring.

If other is also the integer ring, the output is True. In other cases it depends on the implementation of another ring's issubring method.

1.1.3 Rational – rational number

The class of rational numbers.

Initialize (Constructor)

Rational(`numerator`: ***numbers***, `denominator`: ***numbers*=1**)
→ ***Integer***

Construct a rational number from:

- integers,
- float, or
- Rational.

Other objects can be converted if they have toRational methods. Otherwise raise TypeError.

Methods

1.1.3.1 getRing – get ring object

getRing(`self`) → ***RationalField***

Return a RationalField object.

1.1.3.2 decimalString – represent decimal

decimalString(`self`, `N`: ***integer***) → ***string***

Return a string of the number to `N` decimal places.

1.1.3.3 expand – continued-fraction representation

expand(`self`, `base`: ***integer***, `limit`: ***integer***) → ***string***

Return the nearest rational number whose denominator is a power of `base` and at most `limit` if `base` is positive integer.

Otherwise, i.e. `base`=0, returns the nearest rational number whose denominator is at most `limit`.

`base` must be non-negative integer.

1.1.4 RationalField – the rational field

RationalField is a class of field of rationals. The class has the single instance **theRationalField**.

This class is a subclass of **QuotientField**.

Initialize (Constructor)

RationalField() → *RationalField*

Create an instance of RationalField. You may not want to create an instance, since there is already theRationalField.

Attribute

zero :
It expresses the additive unit 0, namely Rational(0, 1). (read only)

one :
It expresses the multiplicative unit 1, namely Rational(1, 1). (read only)

Operations

operator	explanation
`x in Q`	return whether an element is in or not.
`str(Q)`	return string.

Methods

1.1.4.1 createElement – create Rational object

createElement(`self`, `numerator`: ***integer or*** **Rational**, `denominator`: ***integer***=**1**)
→ ***Rational***

Create a Rational object.

1.1.4.2 classNumber – get class number

classNumber(`self`) → ***integer***

Return 1, since the class number of the rational field is one.

1.1.4.3 getQuotientField – get rational field object

getQuotientField(`self`) → ***RationalField***

Return the rational field itself.

1.1.4.4 issubring – subring test

issubring(`self`, `other`: **Ring**) → ***bool***

Report whether another ring contains the rational field as subring.

If other is also the rational field, the output is True. In other cases it depends on the implementation of another ring's issuperring method.

1.1.4.5 issuperring – superring test

issuperring(`self`, `other`: **Ring**) → ***bool***

Report whether the rational field contains another ring as subring.

If other is also the rational field, the output is True. In other cases it depends on the implementation of another ring's issubring method.