Contents

Chapter 1

Classes

1.1 imaginary – complex numbers and its functions

The module `imaginary` provides complex numbers. The functions provided are mainly corresponding to the cmath standard module.

- **Classes**
  - **ComplexField**
  - **Complex**
  - †**ExponentialPowerSeries**
  - †**AbsoluteError**
  - †**RelativeError**
- **Functions**
  - **exp**
  - **expi**
  - **log**
  - **sin**
  - **cos**
  - **tan**
  - **sinh**
  - **cosh**
  - **tanh**
  - **atanh**
  - **sqrt**

This module also provides following constants:

e :
`e` is the base of the natural logarithm function, also called Napier's constant. This is the same as **real.e**.

pi :
`pi` is the circular constant, also denoted by π .

This is the same as **real.pi**.

j :
`j` is the imaginary unit.

defaultError :
`defaultError` is the instance of **RelativeError**.

theComplexField :
`theComplexField` is the instance of **ComplexField**.

1.1.1 ComplexField – field of complex numbers

The class is for the field of complex numbers. The class has the single instance **theComplexField**.

Ths class is a subclass of **Field**.

Initialize (Constructor)

ComplexField() → *ComplexField*

Create an instance of ComplexField. You may not want to create an instance, since there is already **theComplexField**.

Attribute

zero :

It expresses The additive unit 0. (read only)

one :

It expresses The multiplicative unit 1. (read only)

Operations

operator	explanation
`in`	membership test; return whether an element is in or not.
`repr`	return representation string.
`str`	return string.

Methods

1.1.1.1 createElement – create Imaginary object

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

Return a Complex object with `seed`.

`seed` must be complex or numbers has embedding to complex.

1.1.1.2 getCharacteristic – get characterisitic

getCharacteristic(`self`) → *integer*

Return the characteristic, zero.

1.1.1.3 issubring – subring test

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

Report whether another ring contains the complex field as subring.

1.1.1.4 issuperring – superring test

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

Report whether the complex field contains another ring as subring.

1.1.2 Complex – a complex number

Complex is a class of complex number. Each instance has a coupled numbers; real and imaginary part of the number.

Ths class is a subclass of **FieldElement**.

All implemented operators in this class are delegated to complex type.

Initialize (Constructor)

Complex(`re`**:** ***number*** `im`**:** ***number*****=0)** → ***Imaginary***

Create a complex number.

`re` can be either real or complex number. If `re` is real and `im` is not given, then its imaginary part is zero.

Attribute

real :

It expresses the real part of complex number.

imag :

It expresses the imaginary part of complex number.

Methods

1.1.2.1 getRing – get ring object

getRing(`self`) → ***ComplexField***

Return the complex field instance.

1.1.2.2 arg – argument of complex

arg(`self`) → ***radian***

Return the angle between the x-axis and the number in the Gaussian plane. *radian* must be Float.

1.1.2.3 conjugate – complex conjugate

conjugate(`self`) → ***Complex***

Return the complex conjugate of the number.

1.1.2.4 copy – copied number

copy(`self`) → ***Complex***

Return the copy of the number itself.

1.1.2.5 inverse – complex inverse

inverse(`self`) → ***Complex***

Return the inverse of the number.
If the number is zero, ZeroDivisionError is raised.

1.1.3 ExponentialPowerSeries – exponential power series

ExponentialPowerSeries is a class for exponential power serieses, whose n-th term has form $\frac{a_n x^n}{n!}$.

Initialize (Constructor)

ExponentialPowerSeries(`iterator`**:** ***iterator*****)** → ***ExponentialPowerSeries***

Construct an exponential power series with coefficient generated by the given `iterator`, which can be an infinite iterator.

Operations

operator	explanation
`(x,maxerror)`	Return the value of the series with `x` assigned. The maximum error `maxerror` must be give

Examples

```
>>> expo = ExponentialPowerSeries(itertools.cycle([1]))
>>> expo(.5, defaultError)
Rational(5434422938503507, 3296144130048000)
```

Methods

1.1.3.1 terms – generator of terms of series

terms(`self`, `x`: *numbers*) → *ExponentialPowerSeries*

Generator of terms of series with assigned x value. `x` must be int, long or Float.

1.1.4 AbsoluteError – absolute error

AbsoluteError is the class of absolute error of imaginary numbers.
This class is deprecated.

1.1.5 RelativeError – relative error

AbsoluteError is the class of relative error of imaginary numbers.
This class is deprecated.

1.1.6 exp(function) – exponential value

exp(x: *number*, err: *Error*=**defaultError**) → *number*

Return exponential of x.
err must be **AbsoluteError** or **RelativeError**.

1.1.7 exp(function) – exponential value

exp(x: *number*, err: *Error*=**defaultError**) → *number*

Return exponential of x.
err must be **AbsoluteError** or **RelativeError**.

1.1.8 expi(function) – imaginary exponential value

expi(x: *real number*, err: *Error*=**defaultError**) → *number*

Return exponential of (x×**j**).
x must be in real numbers. err must be **AbsoluteError** or **RelativeError**.

1.1.9 log(function) – logarithm

log(x: *number*, base: *number*=**None**, err: *Error*=**defaultError**)
→ *number*

Return the natural logarithm of x.
There is one branch cut, from 0 along the negative real axis to -infinity, continuous from above.
err must be **AbsoluteError** or **RelativeError**.

1.1.10 sin(function) – sine function

sin(z: *number*, err: *Error*=**defaultError**) → *number*

Return the sine of z.
err must be **AbsoluteError** or **RelativeError**.

1.1.11 cos(function) – cosine function

cos(z: *number*, err: *Error*=**defaultError**) → *number*

Return the cosine of z.
err must be **AbsoluteError** or **RelativeError**.

1.1.12 tan(function) – tangent function

tan(z: *number*, err: *Error*=**defaultError**) → *number*

Return the tangent of z.
err must be **AbsoluteError** or **RelativeError**.

1.1.13 sinh(function) – hyperbolic sine function

sinh(z: *number*, err: *Error*=**defaultError**) → *number*

Return the hyperbolic sine of z.
err must be **AbsoluteError** or **RelativeError**.

1.1.14 cosh(function) – hyperbolic cosine function

cosh(z: *number*, err: *Error*=**defaultError**) → *number*

Return the hyperbolic cosine of z.
err must be **AbsoluteError** or **RelativeError**.

1.1.15 tanh(function) – hyperbolic tangent function

tanh(z: *number*, err: *Error*=**defaultError**) → *number*

Return the hyperbolic tangent of z.
err must be **AbsoluteError** or **RelativeError**.

1.1.16 atanh(function) – hyperbolic arc tangent function

atanh(z: *number*, err: *Error*=**defaultError**) → *number*

Return the arc tangent of z.
err must be **AbsoluteError** or **RelativeError**.

1.1.17 sqrt(function) – square root

sqrt(z: *number*, err: *Error*=**defaultError**) → *number*

Return square root of z.
err must be **AbsoluteError** or **RelativeError**.

Bibliography