9.3. cmath — Mathematical functions for complex numbers

This module is always available. It provides access to mathematical functions for complex numbers. The functions in this module accept integers, floating-point numbers or complex numbers as arguments. They will also accept any Python object that has either a `__complex__()` or a `__float__()` method: these methods are used to convert the object to a complex or floating-point number, respectively, and the function is then applied to the result of the conversion.

Note: On platforms with hardware and system-level support for signed zeros, functions involving branch cuts are continuous on *both* sides of the branch cut: the sign of the zero distinguishes one side of the branch cut from the other. On platforms that do not support signed zeros the continuity is as specified below.

9.3.1. Conversions to and from polar coordinates

A Python complex number `z` is stored internally using *rectangular* or *Cartesian* coordinates. It is completely determined by its *real part* `z.real` and its *imaginary part* `z.imag`. In other words:

```
z == z.real + z.imag*1j
```

Polar coordinates give an alternative way to represent a complex number. In polar coordinates, a complex number *z* is defined by the modulus *r* and the phase angle *phi*. The modulus *r* is the distance from *z* to the origin, while the phase *phi* is the counterclockwise angle, measured in radians, from the positive x-axis to the line segment that joins the origin to *z*.

The following functions can be used to convert from the native rectangular coordinates to polar coordinates and back.

`cmath.`**`phase`**(*x*)

Return the phase of *x* (also known as the *argument* of *x*), as a float. `phase(x)` is equivalent to `math.atan2(x.imag, x.real)`. The result lies in the range [-π, π], and the branch cut for this operation lies along the negative real axis, continuous from above. On systems with support for signed zeros (which includes most systems in current use), this means that the sign of the result is the same as the sign of `x.imag`, even when `x.imag` is zero:

```
>>> phase(complex(-1.0, 0.0))
3.1415926535897931
>>> phase(complex(-1.0, -0.0))
-3.1415926535897931
```

New in version 2.6.

Note: The modulus (absolute value) of a complex number *x* can be computed using the built-in `abs()` function. There is no separate `cmath` module function for this operation.

cmath.**polar**(*x*)

Return the representation of *x* in polar coordinates. Returns a pair `(r, phi)` where *r* is the modulus of *x* and phi is the phase of *x*. `polar(x)` is equivalent to `(abs(x), phase(x))`.

New in version 2.6.

cmath.**rect**(*r*, *phi*)

Return the complex number *x* with polar coordinates *r* and *phi*. Equivalent to `r * (math.cos(phi) + math.sin(phi)*1j)`.

New in version 2.6.

9.3.2. Power and logarithmic functions

cmath.**exp**(*x*)

Return the exponential value `e**x`.

cmath.**log**(*x*[, *base*])

Returns the logarithm of *x* to the given *base*. If the *base* is not specified, returns the natural logarithm of *x*. There is one branch cut, from 0 along the negative real axis to -∞, continuous from above.

Changed in version 2.4: base argument added.

cmath.**log10**(*x*)

Return the base-10 logarithm of *x*. This has the same branch cut as `log()`.

cmath.**sqrt**(*x*)

Return the square root of *x*. This has the same branch cut as `log()`.

9.3.3. Trigonometric functions

cmath.**acos**(*x*)

Return the arc cosine of *x*. There are two branch cuts: One extends right from 1 along the real axis to ∞, continuous from below. The other extends left from -1 along the real axis to -∞, continuous from above.

cmath.**asin**(*x*)

Return the arc sine of *x*. This has the same branch cuts as `acos()`.

cmath.**atan**(*x*)

Return the arc tangent of *x*. There are two branch cuts: One extends from `1j` along the imaginary axis to `∞j`, continuous from the right. The other extends from `-1j` along the imaginary axis to `-∞j`, continuous from the left.

Changed in version 2.6: direction of continuity of upper cut reversed

cmath.**cos**(*x*)

Return the cosine of *x*.

`cmath.`**`sin`**(*x*)
Return the sine of *x*.

`cmath.`**`tan`**(*x*)
Return the tangent of *x*.

9.3.4. Hyperbolic functions

`cmath.`**`acosh`**(*x*)
Return the inverse hyperbolic cosine of *x*. There is one branch cut, extending left from 1 along the real axis to -∞, continuous from above.

`cmath.`**`asinh`**(*x*)
Return the inverse hyperbolic sine of *x*. There are two branch cuts: One extends from `1j` along the imaginary axis to `∞j`, continuous from the right. The other extends from `-1j` along the imaginary axis to `-∞j`, continuous from the left.

Changed in version 2.6: branch cuts moved to match those recommended by the C99 standard

`cmath.`**`atanh`**(*x*)
Return the inverse hyperbolic tangent of *x*. There are two branch cuts: One extends from `1` along the real axis to `∞`, continuous from below. The other extends from `-1` along the real axis to `-∞`, continuous from above.

Changed in version 2.6: direction of continuity of right cut reversed

`cmath.`**`cosh`**(*x*)
Return the hyperbolic cosine of *x*.

`cmath.`**`sinh`**(*x*)
Return the hyperbolic sine of *x*.

`cmath.`**`tanh`**(*x*)
Return the hyperbolic tangent of *x*.

9.3.5. Classification functions

`cmath.`**`isinf`**(*x*)
Return `True` if the real or the imaginary part of x is positive or negative infinity.

New in version 2.6.

`cmath.`**`isnan`**(*x*)
Return `True` if the real or imaginary part of x is not a number (NaN).

New in version 2.6.

9.3.6. Constants

`cmath.`**`pi`**
The mathematical constant *π*, as a float.

`cmath.`**`e`**
The mathematical constant *e*, as a float.

Note that the selection of functions is similar, but not identical, to that in module `math`. The reason for having two modules is that some users aren't interested in complex numbers, and perhaps don't even know what they are. They would rather have `math.sqrt(-1)` raise an exception than return a complex number. Also note that the functions defined in `cmath` always return a complex number, even if the answer can be expressed as a real number (in which case the complex number has an imaginary part of zero).

A note on branch cuts: They are curves along which the given function fails to be continuous. They are a necessary feature of many complex functions. It is assumed that if you need to compute with complex functions, you will understand about branch cuts. Consult almost any (not too elementary) book on complex variables for enlightenment. For information of the proper choice of branch cuts for numerical purposes, a good reference should be the following:

See also: Kahan, W: Branch cuts for complex elementary functions; or, Much ado about nothing's sign bit. In Iserles, A., and Powell, M. (eds.), The state of the art in numerical analysis. Clarendon Press (1987) pp165–211.