**Number-theoretic and representation functions**

math.ceil(X)

Returns the neXt largest integer value around the passed value x.

math.copysign(X, y)

Returns a float number with the sign copied from y

math.fabs(X)

Return the absolute value of X.

math.factorial(X)

Returns the factorial of X as an integer. When X is not integral or negative the function throws ValueError. It accepts only positive integer.

math.floor(X)

Return the floor of X, the largest integer less than or equal to X. If X is not a float, delegates to X.\_\_floor\_\_(), which should return an Integral value.

math.fmod(X, y)

Return fmod(X, y), as defined by the platform C library. Note that the Python eXpression X % y may not return the same result. The intent of the C standard is that fmod(X, y) be eXactly (mathematically; to infinite precision) equal to X - n\*y for some integer n such that the result has the same sign as X and magnitude less than abs(y). Python’s X % y returns a result with the sign of y instead, and may not be eXactly computable for float arguments. For eXample, fmod(-1e-100, 1e100) is -1e-100, but the result of Python’s -1e-100 % 1e100 is 1e100-1e-100, which cannot be represented eXactly as a float, and rounds to the surprising 1e100. For this reason, function fmod() is generally preferred when working with floats, while Python’s X % y is preferred when working with integers.

math.freXp(X)

Return the mantissa and eXponent of X as the pair (m, e). m is a float and e is an integer such that X == m \* 2\*\*e eXactly. If X is zero, returns (0.0, 0), otherwise 0.5 <= abs(m) < 1. This is used to “pick apart” the internal representation of a float in a portable way.

math.fsum(iterable)

Return an accurate floating point sum of values in the iterable. Avoids loss of precision by tracking multiple intermediate partial sums:

math.gcd(\*integers)

Return the greatest common divisor of the specified integer arguments. If any of the arguments is nonzero, then the returned value is the largest positive integer that is a divisor of all arguments. If all arguments are zero, then the returned value is 0. gcd() without arguments returns 0.

math.isclose(a, b, \*, rel\_tol=1e-09, abs\_tol=0.0)

Return True if the values a and b are close to each other and False otherwise.

Whether or not two values are considered close is determined according to given absolute and relative tolerances.

rel\_tol is the relative tolerance – it is the maXimum allowed difference between a and b, relative to the larger absolute value of a or b. For eXample, to set a tolerance of 5%, pass rel\_tol=0.05. The default tolerance is 1e-09, which assures that the two values are the same within about 9 decimal digits. rel\_tol must be greater than zero.

abs\_tol is the minimum absolute tolerance – useful for comparisons near zero. abs\_tol must be at least zero.

If no errors occur, the result will be: abs(a-b) <= maX(rel\_tol \* maX(abs(a), abs(b)), abs\_tol).

math.isfinite(X)

Return True if X is neither an infinity nor a NaN, and False otherwise. (Note that 0.0 is considered finite.)

math.isinf(X)

Return True if X is a positive or negative infinity, and False otherwise.

math.isnan(X)

Return True if X is a NaN (not a number), and False otherwise.

math.isqrt(n)

Return the integer square root of the nonnegative integer n. This is the floor of the eXact square root of n, or equivalently the greatest integer a such that a² ≤ n.

For some applications, it may be more convenient to have the least integer a such that n ≤ a², or in other words the ceiling of the eXact square root of n. For positive n, this can be computed using a = 1 + isqrt(n - 1).

.

math.lcm(\*integers)

Return the least common multiple of the specified integer arguments. If all arguments are nonzero, then the returned value is the smallest positive integer that is a multiple of all arguments. If any of the arguments is zero, then the returned value is 0. lcm() without arguments returns 1.

.

math.ldeXp(X, i)

Return X \* (2\*\*i). This is essentially the inverse of function freXp().

math.modf(X)

Return the fractional and integer parts of X. Both results carry the sign of X and are floats.

math.neXtafter(X, y)

Return the neXt floating-point value after X towards y.

If X is equal to y, return y.

EXamples:

math.neXtafter(X, math.inf) goes up: towards positive infinity.

math.neXtafter(X, -math.inf) goes down: towards minus infinity.

math.neXtafter(X, 0.0) goes towards zero.

math.neXtafter(X, math.copysign(math.inf, X)) goes away from zero.

See also math.ulp().

.

math.perm(n, k=None)

Return the number of ways to choose k items from n items without repetition and with order.

Evaluates to n! / (n - k)! when k <= n and evaluates to zero when k > n.

If k is not specified or is None, then k defaults to n and the function returns n!.

Raises TypeError if either of the arguments are not integers. Raises ValueError if either of the arguments are negative.

.

math.prod(iterable, \*, start=1)

Calculate the product of all the elements in the input iterable. The default start value for the product is 1.

When the iterable is empty, return the start value. This function is intended specifically for use with numeric values and may reject non-numeric types.

math.remainder(X, y)

Return the IEEE 754-style remainder of X with respect to y. For finite X and finite nonzero y, this is the difference X - n\*y, where n is the closest integer to the eXact value of the quotient X / y. If X / y is eXactly halfway between two consecutive integers, the nearest even integer is used for n. The remainder r = remainder(X, y) thus always satisfies abs(r) <= 0.5 \* abs(y).

math.trunc(X)

Return the Real value X truncated to an Integral (usually an integer). Delegates to X.\_\_trunc\_\_().

math.ulp(X)

Return the value of the least significant bit of the float X:

If X is a NaN (not a number), return X.

If X is negative, return ulp(-X).

If X is a positive infinity, return X.

If X is equal to zero, return the smallest positive denormalized representable float (smaller than the minimum positive normalized float, sys.float\_info.min).

If X is equal to the largest positive representable float, return the value of the least significant bit of X, such that the first float smaller than X is X - ulp(X).

Otherwise (X is a positive finite number), return the value of the least significant bit of X, such that the first float bigger than X is X + ulp(X).

ULP stands for “Unit in the Last Place”.