**Power and logarithmic functions**

The logarithmic function is said to be the opposite of the exponential function. Log functions give numerical stability to the data with numerical convenience. Using log functions prevents numerical instability and helps handle the skewed distribution of the data. Log works efficiently rather than float or integer values in any model.

Few examples of the power and log functions in python:

**math.exp(X) -** returns the value e to the power X, where e = 2.718281 [base of natural logarithms]. This function is usually the most accurate.

**math.expm1(X)** - returns e to the power X, subtracted by 1. For small floats X, the subtraction in [math.exp(X) - 1] results in a significant loss of precision whereas the function math.eXpm1() computes this quantity in full precision.

**math.log(X[, base]) -** returns the natural logarithm of X to the base(passed as argument), computed as log(X)/log(base). When one argument is passed, the function returns the natural logarithm of X to the base of e.

**math.log1p(X)** - returns the natural logarithm of 1+X (base-e). The result is calculated as such is accurate for X near zero.

**math.log2(X) -** returns the base-2 log(X). This function is more accurate than log(X, 2).

**math.log10(X) -** returns the base-10 log(X). This function is more accurate than log(X, 10).

**math.pow(X, y) -** returns X raised to the power y. The built-in \*\* operator computes with the exact integer power whereas math.pow() converts both the arguments to type float. Hence \*\* or the pow() function is preferred while computing exact integer powers.