Math Library Tutorial

The math library is documented in section 5.5 of the Reference Manual. Below is a summary of the functions provided. Each function is described, with an example, on this page.

```
math.abs     math.acos    math.asin    math.atan    math.atan2
math.ceil    math.cos     math.deg     math.exp     math.floor
math.log     math.log10   math.max     math.min     math.mod
math.pow     math.rad     math.sin     math.sqrt    math.tan
math.frexp   math.ldexp   math.random  math.randomseed
```

math.abs

Return the absolute, or positive value, of a given value.

```
> = math.abs(-100)
100
> = math.abs(25.67)
25.67
```

math.acos , math.asin

Return the inverse cosine and sine of the given value.

```
> = math.acos(1)
0
> = math.acos(0)
1.5707963267949
> = math.asin(0)
0
> = math.asin(1)
1.5707963267949
```

math.atan , math.atan2

Return the inverse tangent. We can do this by supplying y/x ourselves using `math.atan` or we can pass y and x to `math.atan2` to do this for us.

```
> c, s = math.cos(0.8), math.sin(0.8)
> = math.atan(s/c)
0.8
> = math.atan2(s,c)
0.8
```

`math.atan2` should usually be preferred, particularly when converting rectangular co-ordinates to polar co-ordinates. `math.atan2` uses the sign of both arguments to place the result into the correct quadrant, and also produces correct values when one of its arguments is 0 or very close to 0.

```
> = math.atan2(1, 0), math.atan2(-1, 0), math.atan2(0, 1), math.atan2(0, -1)
1.5707963267949 -1.5707963267949        0       3.1415926535898
```

math.ceil , math.floor

Return the integer above and below a given value.

```
> = math.floor(0.5)
0
> = math.ceil(0.5)
1
```

math.cos , math.sin , math.tan

Return the cosine, sine and tangent value for a given value.

```
> = math.cos(math.pi / 4)
0.70710678118655
> = math.sin(0.123)
0.12269009002432
> = math.tan(5/4)
3.0095696738628
> = math.tan(.77)
0.96966832796149
```

math.deg , math.rad

Convert from radians to degrees and vice versa.

```
> = math.deg(math.pi)
180
```

```
> = math.deg(math.pi / 2)
90
> = math.rad(180)
3.1415926535898
> = math.rad(1)
0.017453292519943
```

math.exp , math.log

`math.exp()` returns *e* (the base of natural logarithms) raised to a power given. `math.log()` returns the inverse of this. *e* has the value returned by `math.exp(1)`.

```
> = math.exp(0)
1
> = math.exp(1)
2.718281828459
> = math.exp(27)
532048240601.8
> = math.log(532048240601)
26.999999999998
> = math.log(3)
1.0986122886681
```

math.log10

Return the base 10 logarithm of a given number. The number must be positive.

```
> = math.log10(100)
2
> = math.log10(256)
2.4082399653118
> = math.log10(-1)
-1.#IND
```

math.pow , x^y

`math.pow()` raises the first parameter to the power of the second parameter and returns the result. The binary `^` operator performs the same job as `math.pow()`, i.e. *math.pow(x,y) == x^y*.

```
> = math.pow(100,0)
1
> = math.pow(7,2)
```

```
49
> = math.pow(2,8)
256
> = math.pow(3,2.7)
19.419023519771
> = 5 ^ 2
25
> = 2^8
256
```

math.min , math.max

Return the minimum or maximum value from a variable length list of arguments.

```
> = math.min(1,2)
1
> = math.min(1.2, 7, 3)
1.2
> = math.min(1.2, -7, 3)
-7
> = math.max(1.2, -7, 3)
3
> = math.max(1.2, 7, 3)
7
```

math.mod

Divide the first argument by the second and return the remainder.

```
> = math.mod(7,3)
1
> = math.mod(9,3)
0
> = math.mod(100,2.3)
1.1
```

math.sqrt

Return the square root of a given number. Only positive arguments are allowed.

```
> = math.sqrt(100)
```

```
10
> = math.sqrt(1234)
35.128336140501
> = math.sqrt(-7)
-1.#IND
```

math.random , math.randomseed

`math.random()` generates random numbers. Supplying argument alters its behaviour:

- `math.random()` with no arguments generates a number between 0 and 1.
- `math.random(upper)` generates integer numbers between 1 and *upper*.
- `math.random(lower, upper)` generates integer numbers between *lower* and *upper*.

```
> = math.random()
0.0012512588885159
> = math.random()
0.56358531449324
> = math.random(100)
20
> = math.random(100)
81
> = math.random(70,80)
76
> = math.random(70,80)
75
```

The `math.randomseed()` function sets a *seed* for the pseudo-random generator: Equal seeds produce equal sequences of numbers.

```
> math.randomseed(1234)
> = math.random(), math.random(), math.random()
0.12414929654836        0.0065004425183874      0.3894466994232
> math.randomseed(1234)
> = math.random(), math.random(), math.random()
0.12414929654836        0.0065004425183874      0.3894466994232
```

A good 'seed' is os.time(). To get nice random numbers use:

```
math.randomseed( os.time() )
```

But beware! The first random number you get is not really 'randomized' (at least in Windows 2k). To get better pseudo-random number just pop some random

number before using them for real:

```
-- Initialize the pseudo random number generator
math.randomseed( os.time() )
math.random(); math.random(); math.random()
-- done. :-)
```

math.frexp , math.ldexp

These are normalisation functions [1]. The `math.frexp()` function is used to split the number value into a normalized fraction and an exponent. Two values are returned: the first is a value always in the range 1/2 (inclusive) to 1 (exclusive) and the second is an exponent. The `math.ldexp()` function takes a normalised number and returns the floating point representation. This is the value multiplied by 2 to the power of the exponent.

```
> = math.frexp(2)
0.5     2
> = math.frexp(3)
0.75    2
> = math.frexp(128)
0.5     8
> = math.frexp(3.1415927)
0.785398175     2
> = math.ldexp(0.785,2)
3.14
> = math.ldexp(0.5,8)
128
```
