CVA & SimpleCalc X Function Reference

Special Symbols

`.`	decimal point or thousands separator (depending on the number format settings)	
`,`	decimal point or thousands separator (depending on the number format settings)	
`;`	separator of argument lists	
`"`	first character of a comment:	`sin(π/3) "this is a comment…`
`?`	displays accompanying text of currency symbols	`?BRL = Brazilian Real` `?GBP = U.K. Pound Sterling`

Dyadic Operators (by ascending order of precedence)

`=`	subtraction with lowest precedence; this is especially useful in combination with the Newton Solver (s. below), to subtract the right side of an equation from the left side the following results in 5:	`solve(x := 3; x^2 - 3*x + 2 = 12)`
`:=`	variable assignment operator	`x := 5` `(x + 2)*3 = 21`
	variables keep its definition even upon restart until undefined with:	`x := 0`
`´`	number of decimal digits:	`π´6 = 3.141593` (s. rounding functions)
`` ` ``	number of significant digits:	`10*π`4 = 31.42`
`+ -`	addition and subtraction	`1 + 2 = 3`
`or ||`	logical OR	`true or false = true` or `3.4||0 = true`
`* · /`	multiplication and division	`1*1 = 1` or `2·2 = 4` <shift><option><9> = `·` `π/4 = 0.7853981634…`
`and &&`	logical AND	`true and false = false` or `2&&1 = true`
`xor`	logical XOR	`true xor true = false  2 xor 0 = true`
`mod`	modulo	`17 mod 5 = 2`
`div`	integral division	`17 div 5 = 3`
`^`	power (evaluation from left):	`2^3^2 = (2^3)^2 = 64`
`√`	root:	`3√5 = 1.709975947…`

`@`	2D distance:	`x := 6`
		`y := 8`
		`x @ y = 10 = 2√(x^2 + y^2)`

Monadic Operators (always highest precedence)

-	negation:	`x := 6  -x = -6`	
√	square root:	`√3 = sqrt(3) = 2√3 = 1.7320508…`	

!	logical NOT	`!true = false`
		`!0 = true`
		`!0.0 = true`
		`!5.5 = false`
		`!nan = nan`	

°	radiant → grad:	`°x = x·180/π`	(angle conversions, useful for
•	grad → radiant:	`•x = x·π/180`	the trigonometric functions)
ç	pt → cm:	`çx = x·2.54/72`	(measurement conversions,
¶	cm → pt:	`¶x = x·72/2.54`	useful in graphics applications)

Bitwise Operators

Bitwise Operators can be used only for integral operands. SimpleCalc/InstantCalc use boolean (1 bit) and 64 bit integers. The bitwise operators affect ALL bits, including the sign bit and the leading bits containing zeros, for this reason bitwise inverse of a positive number gives allways a negative number and vice versa.

~	bitwise inverse:	`~5 = -6`
		`\~5 = #1111111111111111111111111111111`
		`1111111111111111111111111111010`	

&	bitwise AND	`5 & 3 = 1`	
|	bitwise OR	`5 | 3 = 7`	
≈	bitwise XOR	`5 ≈ 3 = 6`	
<<	bitwise SHL	`128 << 2 = 512`	shift left
>>	bitwise SHR	`32 >> 4 = 2`	shift right

Relational Operators (return true or false)

==	is equal	`sin(π) == cos(π) = false`
!= <>	is not equal	`0 != 0.0 = false`
<	is less than	`have(3;6;7;9) < ave(3;6;7;9) = true`
<=	is less or equal than	`exp(2) <= 10 = true`
>	is greater than	`5 > 3 = true`
>=	is greater or equal than	`stdev(3;6;7;9) >= 2 = true`

Decimal/Hexadecimal and Decimal/Binary Conversion Operators

`$` precedes a hexadecimal number (only for integral values)

```
$20 + 5  =  37
```

`&` converts the result to a hexadecimal number – non integral results are rounded before (see round). `&` must be the first character of the expression.

```
&10 + 20 + $1E = $3C
```

`#` precedes a binary number (only for integral values)

```
#11100101 + 5 = 234
```

`\` converts the result into binary number format– non integral results are rounded before (see rint). \ must be the first character of the expression.

```
\1234 = #10011010010
```

Display of Fractions

ƒ converts the result to a fraction – *ƒ* must be the first character of the expression.

```
ƒ0.1875 = 3/16
ƒ11.5 = 11 1/2 – " " (option space) works like "+"
11 1/2 = 11.5
```

Functions With One Argument

`abs`	absolute value:	`abs(x)`	=	|x|
`sign`	signum function:	`sign(x)`	=	{ -1 for x < 0; 0 for x = 0; 1 for x > 0 }
`sqrt`	square root:	`sqrt(x)`	=	$\sqrt{x}$
`sqr`	power of 2:	`sqr(x)`	=	x^2
`cube`	power of 3:	`cube(x)`	=	x^3
`grad`	radians → grad:	`grad(x)`	=	$x \cdot 180/\pi$ (angle conversions, useful for
`rad`	grad → radians:	`rad(x)`	=	$x \cdot \pi/180$ the trigonometric functions)
`round`	to integral:	`round(1.5)`	=	2
`trunc`	…ate to integral:	`trunc(1.5)`	=	1
`fact`	factorial:	`fact(6)`	=	720 (x!, if x ≤ 20 and integral,
		`fact(6.01)`	=	733.616732 otherwise gamma(x + 1))
`factLn`	log-factorial:	`factLn(x)`	=	ln(x!) if x ≤ 20 and integral, otherwise gammaLn(x + 1)
`perm`	permutations:	`perm(n)`	=	n! if n ≤ 20 and integral, otherwise gamma(n + 1)
`gamma`	gamma function	`gamma(5.5)`	=	52.34277778…
`gammaLn`	log-gamma function	`gammaLn(x)`	=	ln(gamma(x))

Exponential Functions

`exp`	base e	`exp(x)`	=	e^x
`expm1`	base e - 1	`expm1(x)`	=	e^x - 1
`exp10`	base 10	`exp10(x)`	=	10^x
`exp2`	base 2	`exp2(x)`	=	2^x

Logarithms

`ln`	base e	`ln(x)`		Logarithmus Naturalis
`ln1p`	base e	`ln1p(5)`	=	ln(5 + 1) = 1.791759469...
`lg`	base 10	`lg(100)`	=	2
`lb`	base 2	`lb(64)`	=	6

Trigonometric Functions (argument in radians)

`sin`	sine	`sin(π/3)`	=	0.8660254038
`cos`	cosine	`cos(•45)`	=	0.7071067812
`tan`	tangent	`tan(π/8)`	=	0.4142135624
`cot`	cotangent	`cot(π/8)`	=	2.414213562 1/tan(π/8)

Inverse Trigonometric Functions (result in radians)

`asin`	inverse sine	`asin(0.5)`	=	0.5235987756
`acos`	inverse cosine	`°acos(0.5)`	=	60
`atan`	inverse tangent	`°atan(inf)`	=	90
`acot`	inverse cotangent	`°acot(1)`	=	45

Hyperbolic Functions

`sinh`	hyperbolic sine
`cosh`	hyperbolic cosine
`tanh`	hyperbolic tangent
`coth`	hyperbolic cotangent

Inverse Hyperbolic Functions

`asinh`	inverse hyperbolic sine
`acosh`	inverse hyperbolic cosine
`atanh`	inverse hyperbolic tangent
`acoth`	inverse hyperbolic cotangent

Functions With Mulitiple Arguments (separated by semicolon)

`root`	nth root	`root(n; x)`	= $\sqrt[n]{x}$
`scalb`	binary scale	`scalb(x; n)`	= $x \cdot 2^n$
`pyth2D`	geometric distance	`pyth2D(x; y)`	= $\sqrt{(x^2 + y^2)}$
`pyth3D`	geometric distance	`pyth3D(x; y; z)`	= $\sqrt{(x^2 + y^2 + z^2)}$
`gauss`	Gauss distribution	`gauss(x; m; s)`	x around m with standard deviation s
`gaussLn`	log-Gauss distribution	`gaussLn(x;m;s)`	ln(x) around ln(m) with std. dev. ln(s)
`ggt`	largest common divisor	`ggt(x; y)`	ggt(528; 112) = 16
`kgv`	smallest common multiple	`kgv(x; y)`	kgv(528; 112) = 3696
`comb`	combinations	`comb(n; k)`	= n!/[k! ·(n-k)!] binomial coefficient
		`comb(49; 6)`	= 13983816
`var`	variations	`var(n; k)`	= n!/(n-k)!
		`var(4; 2)`	= 12
`compound`	(interest)	`compound(r; n)`	= $(1 + r)^n$ - compound is more exact
20000 $ at 1.5 %/a after 60 m		`20000*compound(0.015/12; 60)`	= 21556.67… $
`annuity`	(payments)	`annuity(r; n)`	= $[1 - (1+r)^{-n}]/r$ - annuity is more exact
mortgage rate, 4.5 %/a, 60 m		`20000/annuity(0.045/12; 60)`	= 372.86 $/m
`if`	conditional result	`if(true; 1; 2) = 1`	1. expression if condition is true
		`if(false; 1; 2) = 2`	2. expression if condition is false
		`if(random < 0.5; 0; H*2+O) = 18.01528`	
		`if(random < 0.5; 0; H*2+O) = 0`	
		`if(random < 0.5; 0; H*2+O) = 0`	
		`if(random < 0.5; 0; H*2+O) = 18.01528`	

Functions With Variable Number Of Arguments

Almost unlimited number of arguments, separated by semicolon

`sum`	explicit summation:	`sum(1;2;3;4;5;6;7;8;9)`	= 45
`prod`	explicit product:	`prod(1;2;3;4;5;6;7;8;9)`	= 362880
`ave`	arithmetic average:	`ave(1;2;3;4;5;6;7;8;9)`	= 5
`gave`	geometric average:	`gave(1;2;3;4;5;6;7;8;9)`	= 4.147166274…
`have`	harmonic average:	`have(1;2;3;4;5;6;7;8;9)`	= 3.181371861…
`stdev`	standard deviation:	`stdev(1;2;3;4;5;6;7;8;9)`	= 2.738612788…

Function Without Argument

`random`	equally distributed random numbers between 0 and 1

Numerical Mathematical Methods

Numerical accuracy can be adjusted in the SimpleCalc peferences drawer. The numerical methods are invoked by function calls with two or three arguments which are separated by semicolon. The first argument denotes the variable for the method, and usually it shall be set to the initial/start value by using the variable assignment operator **:=**. The last argument is composed by the expression, that should be evaluated with respect to the variable of the first argument. In the case of Integration, Summation, and Serial Product, the expression in second argument should evaluate the stop value.

solve() Newton solver:

```
solve(u := 3; sin(u))              = 3.1415927...
solve(v := 2; 3·v^2 + 6·v = 45)    = 3
```

Δ() Numerical differentiation:

```
Δ(u := 1; sin(u))  "cos(1)         = 0.5403023...
Δ(v := 2; 3·v^2 + 6·v)             = 18
```

∫() Romberg integration:

```
∫(u := 0; π; sin(u))               = 2
∫(p := -1; 1; 1/(1 + sqr(p)))      = 1.5707963... (π/2)
∫(q := -1; 1; gauss(q; 0; 1))      = 0.6826895...
```

double integration:

```
∫(p := a; b; ∫(q := c; d; f(p; q)))
∫(p :=-3;3; ∫(q :=-2;2; gauss(p;0;1)·gauss(q;0;1)))= 0.9519228...
```

∑() Summation (implicit):

```
∑(k := 1; 9; k)                    = 45
∑(k := 0; 47; (1 + 0.0625/12)^k)   = 54.3729838...
```

same as: `annuity(0.0625/12;48)*compound(0.0625/12;48)` = 54.3729838...

∏() Serial Product:

```
∏(k := 1; 9; k)                    = 362880
```

Encapsulation of the numerical methods into each other, for example for double integration, or for using integration, differentiation, summation, and serial product together with the Newton Solver, is well possible:

```
∫(u := -1; 1; ∫(v := -1; 1; pyth2D(u; v)))                  = 3.0607829...

solve(p := 1.5; ∫(q := -p; p; gauss(q;0;1)) = 0.95)         = 1.9599640...

solve(u := 2; Δ(d := x; π/2·sqr(d) + 4/d) = 0)              = 1.0838521...

solve(r := 1000; 50000·(1+0.0625/12)^48
                 - r·∑(k := 0; 47; (1+0.0625/12)^k))        = 1179.990993...
```

above yields the same as `50000/annuity(0.0625/12; 48)` = 1179.990993...

Constants

`true`	=	1	
`false`	=	0	
`c0`	=	299792458	speed of light in the vacuum
`e`	=	2.71828182845904523536028747135266	Euler´s number
`π, pi`	=	3.14159265358979323846264338327948	
`ln10`	=	2.30258509299404590109361379290931	
`gold`	=	1.61803398874989484820458683436564	golden ratio
`∞, inf`	=	1/0	IEEE-inf
`nan`	=	Not a Number (type 0)	
`none`	=	Not a Number (type 255)	
`maxshort`	=	32767	
`maxlong`	=	2147483647	
`maxint`	=	9223372036854775807	
`maxreal`	=	1.79769313486231580793728971405 30e+308	
`prec`	=	2.46519032881566189191165176650 87e-32	numerical precision

Table of the Atomic Weights of the Elements in g/mol

Compiled from IUPAC Technical Report, Pure Appl. Chem., 78, (2006) 2051-2066.
Applied 2007 changes to the values for lutetium, molybdenum, ytterbium, and zinc.
*Radioactive/Artificial Elements from CRC handbook of Chemistry and Physics 85th ed.

Usage	`H*2+O`	= 18.01528	molecular weight	of water
	`C*2+H*5+O+H`	= 46.06844	"	of ethanol
	`H*2+S+O*4`	= 98.07848	"	of sulfuric acid
	`250*Cr/(Cr+O*3)´4`	= 129.9977	chromium content of 250 g chromic acid	

1.	`H`	=	1.00794	Hydrogen
	`He`	=	4.002602	Helium
2.	`Li`	=	6.941	Lithium
	`Be`	=	9.012182	Beryllium
	`B`	=	10.811	Boron
	`C`	=	12.0107	Carbon
	`N`	=	14.0067	Nitrogen
	`O`	=	15.9994	Oxygen
	`F`	=	18.9984032	Fluorine
	`Ne`	=	20.1797	Neon

3. `Na` = 22.98976928 Sodium
   `Mg` = 24.3050 Magnesium
   `Al` = 26.9815386 Aluminium
   `Si` = 28.0855 Silicon
   `P` = 30.973762 Phosphorus
   `S` = 32.065 Sulfur
   `Cl` = 35.453 Chlorine
   `Ar` = 39.948 Argon

4. `K` = 39.0983 Potassium
   `Ca` = 40.078 Calcium
   `Sc` = 44.955912 Scandium
   `Ti` = 47.867 Titanium
   `V` = 50.9415 Vanadium
   `Cr` = 51.9961 Chromium
   `Mn` = 54.938045 Manganese
   `Fe` = 55.845 Iron
   `Co` = 58.933195 Cobalt
   `Ni` = 58.6934 Nickel
   `Cu` = 63.546 Copper
   `Zn` = 65.38 Zinc
   `Ga` = 69.723 Gallium
   `Ge` = 72.64 Germanium
   `As` = 74.92160 Arsenic
   `Se` = 78.963 Selenium
   `Br` = 79.904 Bromium
   `Kr` = 83.798 Krypton

5. `Rb` = 85.4678 Rubidium
   `Sr` = 87.62 Strontium
   `Y` = 88.90585 Yttrium
   `Zr` = 91.224 Zirconium
   `Nb` = 92.90638 Niobium
   `Mo` = 95.96 Molybdenum
   `Tc` = 97.9072 Technetium*
   `Ru` = 101.07 Ruthenium
   `Rh` = 102.90550 Rhodium
   `Pd` = 106.42 Palladium
   `Ag` = 107.8682 Silver
   `Cd` = 112.411 Cadmium
   `In` = 114.818 Indium
   `Sn` = 118.710 Tin
   `Sb` = 121.760 Antimony
   `Te` = 127.60 Tellurium
   `I` = 126.90447 Iodine
   `Xe` = 131.293 Xenon

6. `Cs` = 132.90054519 Cesium
   `Ba` = 137.327 Barium
   `La` = 138.90547 Lanthanum
   `Ce` = 140.116 Cerium
   `Pr` = 140.90765 Praseodymium
   `Nd` = 144.242 Neodymium
   `Pm` = 144.9127 Promethium*
   `Sm` = 150.36 Samarium
   `Eu` = 151.964 Europium
   `Gd` = 157.25 Gadolinium
   `Tb` = 158.92535 Terbium
   `Dy` = 162.500 Dysprosium
   `Ho` = 164.93032 Holmium
   `Er` = 167.259 Erbium
   `Tm` = 168.93421 Thulium
   `Yb` = 173.054 Ytterbium
   `Lu` = 174.9668 Lutetium
   `Hf` = 178.49 Hafnium
   `Ta` = 180.94788 Tantalum
   `W` = 183.84 Tungsten
   `Re` = 186.207 Rhenium
   `Os` = 190.23 Osmium
   `Ir` = 192.217 Iridium
   `Pt` = 195.084 Platinum
   `Au` = 196.966569 Gold
   `Hg` = 200.59 Mercury
   `Tl` = 204.3833 Thallium
   `Pb` = 207.2 Lead
   `Bi` = 208.98040 Bismuth
   `Po` = 208.9824 Polonium*
   `At` = 209.9871 Astatine*
   `Rn` = 222.0176 Radon*

7. `Fr` = 223.0197 Francium*
   `Ra` = 226.0254 Radium*
   `Ac` = 227.0277 Actinium*
   `Th` = 232.03806 Thorium
   `Pa` = 231.03588 Protactinium
   `U` = 238.02891 Uranium
   `Np` = 237.0482 Neptunium*
   `Pu` = 244.0642 Plutonium*
   `Am` = 243.0614 Americium*
   `Cm` = 247.0704 Curium*
   `Bk` = 247.0703 Berkelium*
   `Cf` = 251.0796 Californium*
   `Es` = 252.0830 Einsteinium*

Fm	=	257.0951	Fermium*
Md	=	258.0984	Mendelevium*
No	=	259.1010	Nobelium*
Lr	=	262.1097	Lawrencium*
Rf	=	261.1088	Rutherfordium*
Db	=	262.1141	Dubnium*
Sg	=	266.1219	Seaborgium*
Bh	=	264.12	Bohrium*
Hs	=	277	Hassium*
Mt	=	268.1388	Meitnerium*
Ds	=	281	Darmstadtium*
Uuu	=	272.1535	Unununium*
Uub	=	285	Ununbium*
Uuq	=	289	Ununquadium*
Uuh	=	289	Ununhexium*
Uuo	=	293	Ununoctium