DISTRIBUTION

You are hereby licensed to make as many copies of this software and documentation as you wish, and distribute the software and documentation in its unmodified form via electronic means. Registered user are prohibited from distributing their personal password or Registration File. There is no charge for any of the above. You are specifically prohibited from charging, or requesting donations, for any such copies, however made; and from distributing the software and/or documentation with other products (commercial or otherwise) without prior written permission by the Author.

CONTACTING AUTHOR

Author : Quyen H. Ho
E-mail : qho1@juno.com (preferred address)
or
qho@west.pima.edu

ORDERING INFORMATION (PLEASE PRINT OUT)
PRODUCT : CURVEPLOTS (3D PLOTTER 32-BIT)

Price guaranteed through January, 1998.
Please Contact Author For Other Arrangements and Current Prices
Author : Quyen H. Ho
E-mail : qho1@juno.com (preferred address)
or
qho@west.pima.edu

Send To:
CurvePlots
3751 W. Meadow Briar Dr.
Tucson AZ, 85741

Please Make Checks Payable To Quyen Ho (Only U.S. Currency)

Cost $15 each = _____
Arizona residents add 7% sales tax + _____

(Please Print All Information)
(Extra Names Please List on Back)
Name(No Longer Then 20 Characters):
_____
Company:
_____
Address:
_____
City, State, Zip:
_____
Country:
_____
Day Phone #:_____ Eve Phone #:_____

(Due to Current Funds, Updates are Made through E-mail only)
(It will be much faster to send your registration number
and current version through E-Mail)
Electronic Mail address:_____

How did you hear about CurvePlots? :
_____

What feature would you like in the next release of CurvePlots? :
_____
_____

Other Comments or Suggestions :
_____
_____
_____

LICENSE AGREEMENT FOR CURVEPLOTS (3D PLOTTER)

-WRITTEN BY QUYEN H. HO

YOU MUST AGREE TO THIS LICENSE AGREEMENT, AND THE CONDITIONS PROVIDED IN THIS DOCUMENTATION OR DO NOT USE THIS SOFTWARE AT ALL.

UNREGISTERED USERS
THIS IS NOT A FREE SOFTWARE.
You may use this software up to an evaluation period of 30 days. If this software is used beyond the 30 days evaluation period a registration fee of $15 U.S. Dollar is required (prices are subject to change). You May continue use of this software beyond the specified 30 Days trial period, provided that you have registered the software during or on the 30 day trial period and have not received your personal password.

REGISTERED USER
You may make one copy of your registration file onto your Computer and you may make one archive copy of the registration file with your software for the sole purpose of backing up the software and protecting your investment from loss. The software and your personal registration file may be moved from one computer to another, so long as there is no possibility of it being used by more than one person at a time.

ALL USERS
DISTRIBUTION
You are hereby licensed to make as many copies of this software and documentation as you wish, and distribute the software and documentation in its unmodified form via electronic means. Registered user are prohibited from distributing their personal password or Registration File. There is no charge for any of the above. You are specifically prohibited from charging, or requesting donations, for any such copies, however made; and from distributing the software and/or documentation with other products (commercial or otherwise) without prior written permission by the Author.

GOVERNING LAW
This agreement shall be governed by the laws of the State of Arizona

DISCLAIMER OF WARRANTY
THIS SOFTWARE AND THE ACCOMPANYING FILES ARE SOLD "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. THE USER ASSUME THE ENTIRE RISK OF USING THE PROGRAM. ANY LIABILITY OF THE SELLER WILL BE LIMITED EXCLUSIVELY TO PRODUCT REPLACEMENT OR REFUND OF PURCHASE PRICE.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PRODUCT, EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES/JURISDICTIONS DO NOT ALLOW THE

EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

INSTALLATION

a. Unzip CrPlt20.zip to an empty Folder
b. Read Readme File for up to date information
c. YOU MUST AGREE TO THE LICENSE AGREEMENT TO
   USE THIS PRODUCT
d. Run the Setup Program and follow Directions.
e. You may delete the setup files when done.
f. Icons should be created for you in the start menu.
   (default group is under the CurvePlots folder; click it and run the program.)

UNINSTALLING

a. Click Start Menu/Settings/Control Panel
b. Double Click Add/Remove Programs Icon
c. Choose CurvePlots and click Add/Remove button.
d. Registered user may have Additional files in your setup folder.
Restart and remove folder.

LIST OF FORMULA VARIABLES

Note: One Argument Functions do not require parenthesis but a space if not used.

F6.X.Z(,)
F6.Y( )
F6.Y.X(,)
F6.Y.Z(,)
F6.Z( )
F6.Z.X(,)
F6.Z.Y(,)

F1, F2, F3, F4, F5, F6

The Formula Variables act like a real variable but evaluates the formula that it reference.

See Also :

Fn.X( ), Fn.X.Y(,), Fn.X.Z(,), Fn.Y( ), Fn.Y.X(,),
Fn.Y.Z(,), Fn.Z( ), Fn.Z.X(,), Fn.Z.Y(,)

F1.X(ARG1), F2.X(ARG1), F3.X(ARG1), F4.X(ARG1), F5.X(ARG1), F6.X(ARG1)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the ‘X’ variable in the formula

See Also :

Fn, Fn.X.Y(,), Fn.X.Z(,), Fn.Y( ), Fn.Y.X(,),
Fn.Y.Z(,), Fn.Z( ), Fn.Z.X(,), Fn.Z.Y(,)

F1.Y(ARG1), F2.Y(ARG1), F3.Y(ARG1), F4.Y(ARG1), F5.Y(ARG1), F6.Y(ARG1)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the 'Y' variable in the formula

See Also :

Fn, Fn.X( ), Fn.X.Y(,), Fn.X.Z(,), Fn.Y.X(,),
Fn.Y.Z(,), Fn.Z( ), Fn.Z.X(,), Fn.Z.Y(,)

F1.Z(ARG1), F2.Z(ARG1), F3.Z(ARG1), F4.Z(ARG1), F5.Z(ARG1), F6.Z(ARG1)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the ‘Z’ variable in the formula

See Also :

Fn, Fn.X( ), Fn.X.Y(,), Fn.X.Z(,), Fn.Y( ),
Fn.Y.X(,), Fn.Y.Z(,), Fn.Z.X(,), Fn.Z.Y(,)

F1.X.Y(ARG1, ARG2), F2.X.Y(ARG1, ARG2), F3.X.Y(ARG1, ARG2),
F4.X.Y(ARG1, ARG2), F5.X.Y(ARG1, ARG2), F6.X.Y(ARG1, ARG2)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the 'X' variable in the formula
Arg2 : Floating point value; it substitute the 'Y' variable in the formula

See Also :

Fn, Fn.X( ), Fn.X.Z(,), Fn.Y( ), Fn.Y.X(,),
Fn.Y.Z(,), Fn.Z( ), Fn.Z.X(,), Fn.Z.Y(,)

F1.X.Z(ARG1, ARG2), F2.X.Z(ARG1, ARG2), F3.X.Z(ARG1, ARG2),
F4.X.Z(ARG1, ARG2), F5.X.Z(ARG1, ARG2), F6.X.Z(ARG1, ARG2)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the 'X' variable in the formula
Arg2 : Floating point value; it substitute the 'Z' variable in the formula

See Also :

Fn, Fn.X( ), Fn.X.Y(,), Fn.Y( ), Fn.Y.X(,),
Fn.Y.Z(,), Fn.Z( ), Fn.Z.X(,), Fn.Z.Y(,)

F1.Z.Y(ARG1, ARG2), F2.Z.Y(ARG1, ARG2), F3.Z.Y(ARG1, ARG2),
F4.Z.Y(ARG1, ARG2), F5.Z.Y(ARG1, ARG2), F6.Z.Y(ARG1, ARG2)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the 'Z' variable in the formula
Arg2 : Floating point value; it substitute the 'Y' variable in the formula

See Also :

Fn, Fn.X( ), Fn.X.Y(,), Fn.X.Z(,), Fn.Y( ),
Fn.Y.X(,), Fn.Y.Z(,), Fn.Z( ), Fn.Z.X(,)

F1.Z.X(ARG1, ARG2), F2.Z.X(ARG1, ARG2), F3.Z.X(ARG1, ARG2),
F4.Z.X(ARG1, ARG2), F5.Z.X(ARG1, ARG2), F6.Z.X(ARG1, ARG2)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the 'Z' variable in the formula
Arg2 : Floating point value; it substitute the 'X' variable in the formula

See Also :

Fn, Fn.X( ), Fn.X.Y(,), Fn.X.Z(,), Fn.Y( ),
Fn.Y.X(,), Fn.Y.Z(,), Fn.Z( ), Fn.Z.Y(,)

F1.Y.X(ARG1, ARG2), F2.Y.X(ARG1, ARG2), F3.Y.X(ARG1, ARG2),
F4.Y.X(ARG1, ARG2), F5.Y.X(ARG1, ARG2), F6.Y.X(ARG1, ARG2)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the 'Y' variable in the formula
Arg2 : Floating point value; it substitute the 'X' variable in the formula

See Also :

Fn, Fn.X( ), Fn.X.Y(,), Fn.X.Z(,), Fn.Y( ),
Fn.Y.Z(,), Fn.Z( ), Fn.Z.X(,), Fn.Z.Y(,)

F1.Y.Z(ARG1, ARG2), F2.Y.Z(ARG1, ARG2), F3.Y.Z(ARG1, ARG2),
F4.Y.Z(ARG1, ARG2), F5.Y.Z(ARG1, ARG2), F6.Y.Z(ARG1, ARG2)

The Formula Variables act like a real variable but evaluates the formula that it reference.

Arg1 : Floating point value; it substitute the 'Y' variable in the formula
Arg2 : Floating point value; it substitute the 'Z' variable in the formula

See Also :

Fn, Fn.X( ), Fn.X.Y(,), Fn.X.Z(,), Fn.Y( ),
Fn.Y.X(,), Fn.Z( ), Fn.Z.X(,), Fn.Z.Y(,)

LIST OF ALL AVAILABLE FUNCTIONS AND OPERATORS

Note: One Argument Functions do not require parenthesis but a space if not used.

MRAM(, , ,)
MRAMX(, , , ,)
NEG( )
NOT( )
OR
PERCENT( )
POWER( )
RAD( )
RAND( )
ROOT( )
ROUND( )
ROUNDX(,)
RRAM(, , ,)
RRAMX(, , , ,)
SEC( )
SIMP(, , ,)
SIMPX(, , , ,)
SIN( )
SINH( )
SQR( )
SQRT( )
TAN( )
TANH( )
TRAP(, , ,)
TRAPX(, , , ,)
XOR

LIST OF FUNCTIONS

Note: One Argument Functions do not require parenthesis but a space if not used.

TRAP(, , ,)
TRAPX(, , , ,)

LIST OF OPERATORS

*
=
/
=
^
=
-
=
+
=
<!
=
<
=
<<
=
<=
=
<>
=
=
=
>!
=
>
=
>=
=
>>
=
AND
DIV
MOD
OR
XOR

LIST OF BOOLEAN FUNCTIONS AND OPERATORS

Note: One Argument Functions do not require parenthesis but a space if not used.

INTEG(ARG1, ARG2, ARG3, ARG4) OR SIMP(ARG1, ARG2, ARG3, ARG4)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Simpson's rule. Limit is from Arg3 to Arg4 with N=50

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit

Example:

Integ(3*x^2, x, 0, 4) := 64
Integx(3*x^2, x, 0, 4, 50) := 64

See Also :

TRAP(, , ,), TRAPX(, , , ,), SIMPX(, , , ,), MRAM(, , ,), MRAMX(, , , ,),
LRAM(, , ,), LRAMX(, , , ,), RRAM(, , ,), RRAMX(, , , ,), INTEGX(, , , ,)

INTEGX(ARG1, ARG2, ARG3, ARG4, ARG5) OR SIMPX(ARG1, ARG2, ARG3, ARG4, ARG5)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Simpson's rule. Limit is from Arg3 to Arg4

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit
Arg5 : N value; the amount of segment for area.

Example:

Integx(3*x^2, x, 0, 4, 50) := 64
Integ(3*x^2, x, 0, 4) := 64

See Also :

TRAP(, , ,), TRAPX(, , , ,), SIMP(, , ,), MRAM(, , ,), MRAMX(, , , ,), LRAM(, , ,), LRAMX(, , , ,), RRAM(, , ,), RRAMX(, , , ,), INTEG(, , ,)

LRAM(ARG1, ARG2, ARG3, ARG4)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Left Riemann Sum. Limit is from Arg3 to Arg4 with N=50

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit

Example:

Lram(3*x^2, x, 0, 2) := 7.76160000000001
LramX(3*x^2, x, 0, 2, 50) := 7.76160000000001

See Also :

TRAP(, , ,), TRAPX(, , , ,), SIMP(, , ,), SIMPX(, , , ,), MRAM(, , ,), MRAMX(, , , ,), LRAMX(, , , ,), RRAM(, , ,), RRAMX(, , , ,), INTEG(, , ,), INTEGX(, , , ,)

LRAMX(ARG1, ARG2, ARG3, ARG4, ARG5)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Left Riemann Sum. Limit is from Arg3 to Arg4

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit
Arg5 : N value; the amount of segment for area.

Example:

LramX(3*x^2, x, 0, 2, 50) := 7.76160000000001
Lram(3*x^2, x, 0, 2) := 7.76160000000001

See Also :

TRAP(, , ,), TRAPX(, , , ,), SIMP(, , ,), SIMPX(, , , ,), MRAM(, , ,), MRAMX(, , , ,),
LRAM(, , ,), RRAM(, , ,), RRAMX(, , , ,), INTEG(, , ,), INTEGX(, , , ,)

RRAM(ARG1, ARG2, ARG3, ARG4)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Right Riemann Sum. Limit is from Arg3 to Arg4 with N=50

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit

Example:

RRam(3*x^2, x, 0, 2) := 8.24160000000001
RRamX(3*x^2, x, 0, 2, 50) := 8.24160000000001

See Also :

TRAP(, , ,), TRAPX(, , , ,), SIMP(, , ,), SIMPX(, , , ,), MRAM(, , ,), MRAMX(, , , ,), LRAM(, , ,), LRAMX(, , , ,), RRAMX(, , , ,), INTEG(, , ,), INTEGX(, , , ,)

RRAMX(ARG1, ARG2, ARG3, ARG4, ARG5)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Right Riemann Sum. Limit is from Arg3 to Arg4

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit
Arg5 : N value; the amount of segment for area.

Example:

RRamX(3*x^2, x, 0, 2, 50) := 8.24160000000001
RRam(3*x^2, x, 0, 2) := 8.24160000000001

See Also :

TRAP(, , ,), TRAPX(, , , ,), SIMP(, , ,), SIMPX(, , , ,), MRAM(, , ,), MRAMX(, , , ,),
LRAM(, , ,), LRAMX(, , , ,), RRAM(, , ,), INTEG(, , ,), INTEGX(, , , ,)

MRAM(ARG1, ARG2, ARG3, ARG4)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Middle Riemann Sum. Limit is from Arg3 to Arg4 with N=50

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit

Example:

MRam(3*x^2, x, 0, 2) := 7.99920000000001
MRamX(3*x^2, x, 0, 2, 50) := 7.99920000000001

See Also :

TRAP(, , ,), TRAPX(, , , ,), SIMP(, , ,), SIMPX(, , , ,), MRAMX(, , , ,), LRAM(, , ,), LRAMX(, , , ,), RRAM(, , ,), RRAMX(, , , ,), INTEG(, , ,), INTEGX(, , , ,)

MRAMX(ARG1, ARG2, ARG3, ARG4, ARG5)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Middle Riemann Sum. Limit is from Arg3 to Arg4

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit
Arg5 : N value; the amount of segment for area.

Example:

MRamX(3*x^2, x, 0, 2, 50) := 7.99920000000001
MRam(3*x^2, x, 0, 2) := 7.99920000000001

See Also :

TRAP(, , ,), TRAPX(, , , ,), SIMP(, , ,), SIMPX(, , , ,), MRAM(, , ,), LRAM(, , ,), LRAMX(, , , ,), RRAM(, , ,), RRAMX(, , , ,), INTEG(, , ,), INTEGX(, , , ,)

TRAP(ARG1, ARG2, ARG3, ARG4)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Trapezoid Rule. Limit is from Arg3 to Arg4 with N=50

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit

Example:

Trap(3*x^2, x, 0, 2) := 8.00160000000001
TrapX(3*x^2, x, 0, 2, 50) := 8.00160000000001

See Also :

TRAPX(, , , ,), SIMP(, , ,), SIMPX(, , , ,), MRAM(, , ,), MRAMX(, , , ,), LRAM(, , ,), LRAMX(, , , ,), RRAM(, , ,), RRAMX(, , , ,), INTEG(, , ,), INTEGX(, , , ,)

TRAPX(ARG1, ARG2, ARG3, ARG4, ARG5)

The Functions return the Approximate Area of Arg1 in respect to Arg2,
using Trapezoid Rule. Limit is from Arg3 to Arg4

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Lower Limit
Arg4 : Higher Limit
Arg5 : N value; the amount of segment for area.

Example:

TrapX(3*x^2, x, 0, 2, 50) := 8.00160000000001
Trap(3*x^2, x, 0, 2) := 8.00160000000001

See Also :

TRAP(, , ,), SIMP(, , ,), SIMPX(, , , ,), MRAM(, , ,), MRAMX(, , , ,), LRAM(, , ,), LRAMX(, , , ,), RRAM(, , ,), RRAMX(, , , ,), INTEG(, , ,), INTEGX(, , , ,)

DERIV(ARG1, ARG2, ARG3)

The Functions return the derivative of Arg1 in respect to Arg2.

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Floating point value use to evaluate answer

Example:

Deriv(2*x, x, 2) := 1.99999999999929

See Also :

DERIVX(, , ,)

DERIVX(ARG1, ARG2, ARG3, ARG4)

The Functions return the derivative of Arg1 in respect to Arg2.

Arg1 : Formula Variable or Formula Literal
Arg2 : Variable use as independent variable of Formula
Arg3 : Floating point value use to evaluate answer
Arg4 : derivative detail

Example:

Derivx(2*x, x, 2,.00045) := 1.99999999999929

See Also :

DERIV(, ,)

!(ARG1)

The Functions return the factorial of Arg1.

Arg1 : Integer value

Example:

!(6) := 720

PERCENT(ARG1) OR %(ARG1)

The Functions return one-hundredth of Arg1.

Arg1 : Floating point value

Example:

Percent(30) := 0.3
%150 := 1.5

ARG1 ^ ARG2

The Function Returns Arg1 to the Power of Arg2; It is exactly the Same as POWER( ) Function

Arg1 : Floating point value
Arg2 : Integer value or fractional value

Example :

8 := 2 ^ 3
2 := 4 ^ .5

See Also :

ROOT( ), POWER( ), SQR( ), CUBE( )

POWER(ARG1, ARG2)

The Function Returns Arg1 to the Power of Arg2; It is exactly the Same as Operator ^

Arg1 : Floating point value
Arg2 : Integer value or fractional value

Example :

8 := Power(2, 3)
2 := Power(4, 1/2)

See Also :

ROOT( ), ^, SQR( ), CUBE( )

ROOT(ARG1, ARG2)

The Function Returns Arg2 as a Root of Arg1

Arg1 : Integer value or fractional value
Arg2 : Floating point value

Example :

2 := Root(2, 4)
8 := Root(1/3, 2)
3 := Root(3, 27)

See Also :

POWER( ), ^, SQRT( ), CUBERT( )

MAX(ARG1, ARG2)

The Function Returns the Greater of the Two Arguments

Arg1 : Floating point value
Arg2 : Floating point value

Example :

78.9 := Max(78.9, 6.8)
34 := Max(7.8, 34)

See Also :

MIN(,), <, <=, <!, >, >=, >!

MIN(ARG1, ARG2)

The Function Returns the Lowest Value of the Two Arguments

Arg1 : Floating point value
Arg2 : Floating point value

Example :
6.8 := Min(78.9, 6.8)
7.8 := Min(7.8, 34)

See Also :
MAX(,), >, >=, >!, <, <!, <=

ROUNDX(ARG1, ARG2)

The Function Returns Arg1 Rounded To The Precision of Arg2

Arg1 : Floating point value
Arg2 : Integer Value in Range [0..9] ;
If Argument is a floating Point Then it will be truncated

Example :
34.567 := RoundX(34.5673789, 3)
34 := RoundX(34.5673789, 0)

See Also :
ROUND( )

LOGX(ARG1, ARG2)

The Function Returns Log of Arg1 to Base Arg2; LogX(Arg1, 10) is exactly the same as LOG(Arg1)

Arg1 : Floating point value
Arg2 : Floating point value

See Also :

LOG( )

ARG1 <! ARG2

The Operator Returns The Smaller Value of The Two Arguments; If Arg1 is a Variable then the Answer is stored into it.

Arg1 : Floating point value; Usually a Variable
Arg2 : Floating point value

Example :

```
6 := 6 <! (5+3)
8 := 6 <! 5 + 3
If AA = 31 then
    10 := AA <! (AA-21)
    66 := AA + 56
```

See Also :

<<, <, <=, >!, >>, >, >=, MIN(,), MAX(,)

ARG1 >! ARG2

The Operator Returns The Greatest Value of The Two Arguments; If Arg1 is a Variable then the Answer is stored into it.

Arg1 : Floating point value; Usually a Variable
Arg2 : Floating point value

Example :

```
8 := 6 >! (5+3)
9 := 6 >! 5 + 3
If AA = 31 then
    40 := AA >! (AA+9)
    96 := AA + 56
```

See Also :

>>, >, >=, <!, <<, <, <=, MAX(,), MIN(,)

ARG1 << ARG2

The Operator Redirect Arg2 to Arg1

Arg1 : Floating point value; Usually a Variable
Arg2 : Floating point value

Example :

```
8 := 6 << (5+3);
8 := 6 << 5 + 3;
If AA = 6 then
    8 := AA << (5+3);
        Now AA := 8
If AA = 6 then
    8 := AA << 5 + 3;
        Now AA := 5
```

See Also :

<!, >>, >!

ARG1 >> ARG2

The Operator Redirect Arg1 to Arg2

Arg1 : Floating point value
Arg2 : Floating point value; Usually a Variable

Example :

```
8 := (5 + 3) >> 6;
8 := 5 + 3 >> 6;
If AA = 6 then
    8 := (5+3) >> AA;
        Now AA := 8
If AA = 6 then
    8 := 5 + 3 >> AA;
        Now AA := 3
```

See Also :

>!, <<, <!

ARG1 * ARG2

The Operator multiply Arg1 to Arg2

Arg1 : Floating point value
Arg2 : Floating point value

See Also :
/, MOD, DIV

ARG1 / ARG2

The Operator Divide Arg1 by Arg2

Arg1 : Floating point value
Arg2 : Floating point value; Not Zero

See Also :
*, MOD, DIV

ARG1 DIV ARG2

The Operator Divide Arg1 by Arg2 and returns An Integer Value that represents the Amount Arg2 divides into Arg1

Arg1 : Floating point value
Arg2 : Floating point value; Not Zero

See Also :
/, *, MOD

ARG1 MOD ARG2

The Operator Divide Arg1 by Arg2 and returns An Integer Value that represents the Remainder; The Answer takes the Sign of Arg1

Arg1 : Floating point value
Arg2 : Floating point value; Not Zero

See Also :
/, *, DIV

ARG1 + ARG2

The Operator Adds Arg1 to Arg2;

Arg1 : Floating point value
Arg2 : Floating point value

See Also :

-

ARG1 - ARG2

The Operator Subtract Arg1 by Arg2; Multiple Minus Signs changes the Signs of the Arguments

Arg1 : Floating point value
Arg2 : Floating point value

Example :

```
11 := 5 - -6
-1 := 5 - --6
```

See Also :

NEG( ), +

ARG1 < ARG2

This Boolean Operation Returns 1 if Arg1 is Less then Arg2, else 0 is return

Arg1 : Floating point value
Arg2 : Floating point value

See Also :
<=, >, >=, >!, <!, MIN(,), MAX(,)

ARG1 <= ARG2

This Boolean Operation Returns 1 if Arg1 is Less or equal to Arg2, else 0 is return

Arg1 : Floating point value
Arg2 : Floating point value

See Also :
<, >, >=, >!, <!, MIN(,), MAX(,), =, NOT( )

ARG1 > ARG2

This Boolean Operation Returns 1 if Arg1 is Greater then Arg2, else 0 is return

Arg1 : Floating point value
Arg2 : Floating point value

See Also :
>=, <, <=, >!, <!, MIN(,), MAX(,)

ARG1 >= ARG2

This Boolean Operation Returns 1 if Arg1 is Greater or equal to Arg2, else 0 is return

Arg1 : Floating point value
Arg2 : Floating point value

See Also :
>, <, <=, >!, <!, MIN(,), MAX(,), =, NOT( )

ARG1 = ARG2

This Boolean Operation Returns 1 if Arg1 is equal to Arg2, else 0 is return

Arg1 : Floating point value
Arg2 : Floating point value

See Also :
<>, NOT( ), <=, >=

ARG1 <> ARG2

This Boolean Operation Returns 1 if Arg1 is Not equal to Arg2, else 0 is return

Arg1 : Floating point value
Arg2 : Floating point value

See Also :
NOT( ), =, <=, >=

ARG1 AND ARG2

This is a Bit-Wise Operator; Arg1 and Arg2 gets truncated to integer if they are Floats;
Operation is done at the Bit Level;

Arg1 : Floating point value
Arg2 : Floating point value

sample :

```
4 := 5 And 6
```

See Also :
NOT( ), OR, XOR

ARG1 OR ARG2

This is a Bit-Wise Operator; Arg1 and Arg2 gets truncated to integer if they are Floats
Operation is done at the Bit Level;

Arg1 : Floating point value
Arg2 : Floating point value

sample:

```
7 := 5 Or 6
```

See Also :
NOT( ), AND, XOR

ARG1 XOR ARG2

This is a Bit-Wise Operator; Arg1 and Arg2 gets truncated to integer if they are Floats
Operation is done at the Bit Level;

Arg1 : Floating point value
Arg2 : Floating point value

sample:

```
3 := 5 Xor 6
```

See Also :
NOT( ), AND, OR

INC(ARG1)

The Function adds 1 to Arg1; If Arg1 is a Variable, answer is Stored into Arg1;

Arg1 : Floating point value

Example :

```
7 := Inc(6)
If AA = 7 then
    8 := Inc(AA)
    Now AA := 8
```

See Also :

DEC( ), <<, <!, >>, >!

DEC(ARG1)

The Function Subtract 1 from Arg1; If Arg1 is a Variable, answer is Stored into Arg1;

Arg1 : Floating point value

Example :

```
5 := Dec(6)
If AA = 7 then
    6 := Dec(AA)
    Now AA := 6
```

See Also :

INC( ), <<, <!, >>, >!

NOT(ARG1)

This is a Bit-Wise Function; Arg1 gets truncated to an integer if it is a Floats
Operation is done at the Bit Level;

Arg1 : Floating point value

sample :

-10 := Not(9)
9 = Not(-10)

See Also

AND, OR, XOR, <>

INV(ARG1)

The Function return the inverse of Arg1; Same as : 1/Arg1

Arg1 : Floating point value

Example :

0.25 := Inv(4)

See Also

/, DIV

DEG(ARG1)

The Function return the Degree representation of Arg1; Same as : Arg1 * 180 / Pi

Arg1 : Floating point value

Example :

360 := Deg(2*pi)

See Also

RAD( )

RAD(ARG1)

The Function return the Radian representation of Arg1; Same as : Arg1 * Pi / 180

Arg1 : Floating point value

Example :

3.14159265358979 := Rad(180)

See Also

DEG( )

SQR(ARG1)

The Function return the Square of Arg1

Arg1 : Floating point value

Example :

25 := Sqr(5)

See Also

SQRT( ), POWER( ), ^

SQRT(ARG1)

The Function return the Square-Root of Arg1

Arg1 : Floating point value

Example :

4 := Sqrt(16)

See Also

SQR( ), ROOT( ), ^

CUBE(ARG1)

The Function return Arg1 to the third Power

Arg1 : Floating point value

Example :

8 := Cube(2)

See Also

CUBERT( ), POWER( ), ^

CUBERT(ARG1)

The Function return the Cube-Root of Arg1

Arg1 : Floating point value

Example :

2 := Cubert(8)

See Also

CUBE( ), ROOT( ), ^

LOG(ARG1)

The Function return the Log base 10 of Arg1

Arg1 : Floating point value

Example :

0.903089986991944 := Log(8)

See Also

LOGX(,), LN( ), EXP( )

LN(ARG1)

The Function return the Natural Log of Arg1

Arg1 : Floating point value

Example :

103 := Ln(Exp(103))

See Also

LOG( ), LOGX(,), EXP( )

EXP(ARG1)

The Function return the Exponent of Arg1

Arg1 : Floating point value

Example :

103 := Exp(Ln(103))

See Also

LN( ), LOG( ), LOGX(,)

SIN(ARG1)

The Function returns Sine of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

COS( ), TAN( ), CSC( ), SEC( ), COT( ), ARCSIN( )

COS(ARG1)

The Function returns Cosine of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

SIN( ), TAN( ), CSC( ), SEC( ), COT( ), ARCCOS( )

TAN(ARG1)

The Function returns Tangent of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

SIN( ), COS( ), CSC( ), SEC( ), COT( ), ARCTAN( )

CSC(ARG1)

The Function returns Cosecant of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

SIN( ), COS( ), TAN( ), SEC( ), COT( )

SEC(ARG1)

The Function returns Secant of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

SIN( ), COS( ), TAN( ), CSC( ), COT( )

COT(ARG1)

The Function returns Cotangent of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

SIN( ), COS( ), TAN( ), CSC( ), SEC( )

ARCSIN(ARG1)

The Function returns The Inverse-Sine of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

SIN( ), ARCCOS( ), ARCTAN( )

ARCCOS(ARG1)

The Function returns The Inverse-Cosine of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

COS( ), ARCSIN( ), ARCTAN( )

ARCTAN(ARG1)

The Function returns The Inverse-Tangent of Arg1; Answer Depends if current Mode is Set to Radian or Degree

Arg1 : Floating point value

See Also

TAN( ), ARCSIN( ), ARCCOS( )

SINH(ARG1)

The Function returns The Hyperbolic-Sine of Arg1

Arg1 : Floating point value

See Also

COSH( ), TANH( )

COSH(ARG1)

The Function returns The Hyperbolic-Cosine of Arg1

Arg1 : Floating point value

See Also

SINH( ), TANH( )

TANH(ARG1)

The Function returns The Hyperbolic-Tangent of Arg1

Arg1 : Floating point value

See Also

SINH( ), COSH( )

ROUND(ARG1)

The Function round Arg1 to nearest Whole Number

Arg1 : Floating point value

See Also

ROUNDX(,)

INT(ARG1)

The Function returns the Integer Part of Arg1

Arg1 : Floating point value

See Also

IPART( ), FPART( )

IPART(ARG1)

The Function returns the Integer Part of Arg1

Arg1 : Floating point value

See Also

INT( ), FPART( )

FPART(ARG1)

The Function returns the Fractional Part of Arg1

Arg1 : Floating point value

See Also

INT( ), IPART( )

ABS(ARG1)

The Function returns the Absolute value of Arg1

Arg1 : Floating point value

See Also

NEG( )

NEG(ARG1)

The Function Negates Arg1

Arg1 : Floating point value

Example :

-5 := Neg(-5)
-5 := Neg(5)

See Also

ABS( )

RAND(ARG1)

The Function returns a Random Number in the Range of [0..(Arg1-1)]

Arg1 : Floating point value