

Calculator

1 Usage

1.1 Quick Examples

Enter the input into the upper window and press Alt+e to evaluate it. The return values appear in the window below. White spaces in the input are treated as multiplication where appropriate.

To calculate $2 + \frac{3!}{1+\pi}$:

```
2+3!/(1+Pi)
```

To define $f(x) = x^2 + \sin(x)$ and calculate $f(2)$:

```
f[x]=x^2+Sin[x];f[2]
```

To display π in base 3:

```
OutputBase=3;Pi
```

To define f to be the factorial function in three different ways: (local variables in function definitions appear after a “.”)

```
f[x]=x!  
f[x]=If[x<=1,1,x*f[x-1]]  
f[x:t]=(t=1;Loop[t=t*x;x=x-1>1];t)
```

To load definitions for all the Trigonometric functions:

```
Examples > 4: Elementary Functions
```

To make a plot of $z = \sin(xy)$:

```
Plot[{x,y,Sin[x*y]},{x,-4,4},{y,-4,4}]
```

To make a plot of $\rho = 2 + \frac{1}{5}\phi^2(\pi - \phi)^2 \sin(6\theta) \sin(7\phi)$ in spherical coordinates:

```
Plot[ r = 2. + 0.2 (phi (Pi - phi))^2 Sin[6.th] Sin[7.phi];  
{r Cos[th] Sin[phi], r Sin[th] Sin[phi], r Cos[phi]},  
{phi, 0, Pi}, {th, 0, 2Pi}]
```

1.2 Main Menu Commands

```
File > Exit
```

closes the calculator.

```
Evaluate
```

evaluates the input window. Syntax errors are displayed in the status bar.

```
Debug > View Symbols
```

prints out all symbols recognized by the calculator, including constants, variables, and functions.

```
Examples > n
```

loads an example input.

2 Data Types

Values are 256 bits in length. These can be:

Complex: stored as two 80-bit reals in the lower 160 bits and the highest bit reset to 0.

Integer: a 255-bit signed integer in the lower 255 bits and the highest bit set to 1.

True: represented by any value with a non-zero lowest byte.

False: represented by any value with a zero lowest byte.

3 Built-in Functions

3.1 Basic Functions

Divide $[z_1, \dots, z_n]$ or $z_1 / \dots / z_n$: returns $(\dots ((z_1 / z_2) / z_3) \dots) / z_n$.

Divide $[z_1]$ or $/z_1$: returns $1/z_1$.

Equal $[z_1, \dots, z_n]$ or $z_1 == \dots == z_n$: returns $z_1 = \dots = z_n$.

Greater $[z_1, \dots, z_n]$ or $z_1 > \dots > z_n$: returns $z_1 > \dots > z_n$.

GreaterEqual $[z_1, \dots, z_n]$ or $z_1 >= \dots >= z_n$: returns $z_1 \geq \dots \geq z_n$.

Less $[z_1, \dots, z_n]$ or $z_1 < \dots < z_n$: returns $z_1 < \dots < z_n$.

LessEqual $[z_1, \dots, z_n]$ or $z_1 <= \dots <= z_n$: returns $z_1 \leq \dots \leq z_n$.

Minus $[z_1, \dots, z_n]$ or $z_1 - \dots - z_n$: returns $(\dots ((z_1 - z_2) - z_3) \dots) - z_n$.

Minus $[z_1]$ or $-z_1$: returns $-z_1$.

Plus $[z_1, \dots, z_n]$ or $z_1 + \dots + z_n$: returns $z_1 + \dots + z_n$.

Plus $[z_1]$ or $+z_1$: returns z_1 .

Power $[z_1, \dots, z_n]$ or $z_1 \wedge \dots \wedge z_n$: returns $z_1^{\dots^{z_n}}$.

Power $[z_1]$ or $\wedge z_1$: returns z_1 .

Sqrt $[z_1]$: returns $\sqrt{z_1}$.

Times $[z_1, \dots, z_n]$ or $z_1 * \dots * z_n$ or $z_1 z_2 \dots z_n$: returns $z_1 \times \dots \times z_n$.

Times $[z_1]$ or $*z_1$: returns z_1 .

3.2 Bit Operations

And $[z_1, \dots, z_n]$ or $z_1 \& \dots \& z_n$: returns the bitwise and of the z_i .

And $[z_1]$ or $\&z_1$: returns z_1 .

Or $[z_1, \dots, z_n]$ or $z_1 | \dots | z_n$: returns the bitwise or of the z_i .

Or $[z_1]$ or $|z_1$: returns z_1 .

Xor $[z_1, \dots, z_n]$ or $z_1 \sim \dots \sim z_n$: returns the bitwise xor of the z_i with the highest bit set to 1.

Xor $[z_1]$ or $\sim z_1$: returns the bitwise not of z_1 with the highest bit set to 1.

3.3 Transcendental Functions

Cos $[z_1]$: returns $\cos(z_1) = (e^{iz_1} + e^{-iz_1})/2$.

Exp $[z_1]$: returns $e^{z_1} = \sum z_1^n / n!$.

Factorial $[z_1]$ or $z_1!$: returns $z_1! = \prod \frac{(1+1/n)^{z_1}}{1+z_1/n}$.

Log $[z_1]$: returns the inverse of e^z where $-\pi < \text{Im}(\log(z)) \leq \pi$ and has a branch cut along the negative real axis.

Sin $[z_1]$: returns $\sin(z_1) = (e^{iz_1} - e^{-iz_1})/2i$.

Tan $[z_1]$: returns $\tan(z_1)$

3.4 Programming Constructs

CmpdExpr $[z_1, \dots, z_n]$ or $z_1; \dots; z_n$: evaluates the z_i in turn and returns the result from z_n .

If $[z_1, z_2]$: If z_1 returns True, evaluates z_2 . return value is undefined.

If $[z_1, z_2, z_3]$: If z_1 returns True, returns z_2 , else z_3 is returned.

Loop $[z_1]$: evaluates z_1 until it returns False. return value is undefined.

Set $[v_1, \dots, v_n, z_1]$ or $v_1 = \dots = v_n = z_1$: evaluates z_1 and assigns the return value to the v_i .

Set $[f[v_1, \dots, v_m : t_1, \dots, t_n], z_1]$ or $f[v_1, \dots, v_m : t_1, \dots, t_n] = z_1$: compiles z_1 as a function of the m arguments v_i and n local variables t_i . This function may then be called via $f[z_1, \dots, z_m]$. A function may only be defined in the top level of an input expression.

While $[z_1, z_2]$: while z_1 returns True, evaluates z_2 . return value is undefined.

3.5 Graphing

Plot $[F, \{u, u_{min}, u_{max}, u_{div}\}, \{v, v_{min}, v_{max}, v_{div}\}]$: Draws a plot of the surface $F([u_{min}, u_{max}] \times [v_{min}, v_{max}])$. F should return three values (the coordinate functions $\{x[u, v], y[u, v], z[u, v]\}$).

Plot $[f, \{x, x_{min}, x_{max}, x_{div}\}, \{y, y_{min}, y_{max}, y_{div}\}, \{z, z_{min}, z_{max}, z_{div}\}]$: Draws a plot of the surface $f(x, y, z) = 0$. f should return one value. CURRENTLY NOT IMPLEMENTED.

plot window controls:

w, a, s, d: movement

i, j, k, l: look

c: change surface color

n: toggle normals

b: toggle box

3.6 Global Variables

InputBase: the base in which input numbers are interpreted. It can be as high as 16; 0-9, $a-f$ are used

OutputBase: the base in which numbers are printed. It can be as high as 16; 0-9, $a-f$ are used

OutputBits: the number of bit of accuracy in printed numbers

Out $[n]$: the n^{th} output

4 Fine Points

1. Precision in complex numbers is limited to 80-bits.
2. Mixing floating point numbers and integers in calculations incurs a costly conversion.
3. The number of functions and variables that can be stored in the calculator is limited only by the memory available.
4. The calculator uses a stack consisting of 1024 entries, which is sufficient for most calculations. If this overflows, an error will be returned and the calculation will be stopped.
5. Support for aborting calculations is currently not implemented.