Calcpad

Version 6.1 quick reference guide

How it works?

1. **Enter** text and formulas into the "**Code**" box on the left.
2. Press **F5** or click ▶ to **calculate**. The results will appear in the "**Output**" box on the right as a professionally formatted Html **report**.
3. Click 🖨 to **print** or 🗐 to **copy** the output.

   You can also **export** it to **Html** 💾, **PDF** 📄 or **MS Word** 📄 document.

The language

The Calcpad language includes the following elements (click an item to insert):

- Real numbers: digits **0** - **9** and decimal point "**.**";
- Complex numbers: **re ± im**i (e.g. **3** - **2**i);
- Variables:

  - Latin letters: a - z, A - Z;

  - Greek letters: α - ω, A - Ω;

  - digits: **0** - **9**;

  - comma: "**,**";

  - prime symbols: **′** , **″** , **‴** , **⁗** ;

  - superscripts: **⁰** , **¹** , **²** , **³** , **⁴** , **⁵** , **⁶** , **⁷** , **⁸** , **⁹** , **ⁿ** , **⁺** , **⁻** ;

  - special symbols: **‾** , **ø** , **Ø** , **°** , **∡** ;

  - " **_** " for subscript;

  A variable name must start with a letter. Names are case sensitive.

- Constants: π, e, φ, γ, g, G, M_E, M_S, c, h, μ_0, ε_0, k_e, e, m_e, m_p, m_n, N_A, σ, k_B, R, F, γ_c, γ_s, γ_a, γ_g, γ_w
- Operators:

  "**!**" - factorial;

  "**^**" - exponent;

  "**/**" - division;

  "**÷**" - force division bar;

  "****" - division;

  "**⊘**" - modulo (reminder);

  "*****" - multiplication;

  "**-**" - minus;

  "**+**" - plus;

  "**≡**" - equal to;

  "**≠**" - not equal to;

  "**<**" - less than;

  "**>**" - greater than;

  "**≤**" - less or equal;

  "**≥**" - greater or equal;

"**∧**" - logical "AND";

"**∨**" - logical "OR";

"**⊕**" - logical "XOR";

"**=**" - assignment;

- Custom functions of type $f(x; y; z; ...)$;
- Built-in functions:
  - Trigonometric:
    - **sin**(x) - sine;
    - **cos**(x) - cosine;
    - **tan**(x) - tangent;
    - **csc**(x) - cosecant;
    - **sec**(x) - secant;
    - **cot**(x) - cotangent;
  - Hyperbolic:
    - **sinh**(x) - hyperbolic sine;
    - **cosh**(x) - hyperbolic cosine;
    - **tanh**(x) - hyperbolic tangent;
    - **csch**(x) - hyperbolic cosecant;
    - **sech**(x) - hyperbolic secant;
    - **coth**(x) - hyperbolic cotangent;
  - Inverse trigonometric:
    - **asin**(x) - inverse sine;
    - **acos**(x) - inverse cosine;
    - **atan**(x) - inverse tangent;
    - **atan2**$(x; y)$ - the angle whose tangent is the quotient of y and x;
    - **acsc**(x) - inverse cosecant;
    - **asec**(x) - inverse secant;
    - **acot**(x) - inverse cotangent;
  - Inverse hyperbolic:
    - **asinh**(x) - inverse hyperbolic sine;
    - **acosh**(x) - inverse hyperbolic cosine;
    - **atanh**(x) - inverse hyperbolic tangent;
    - **acsch**(x) - inverse hyperbolic cosecant;
    - **asech**(x) - inverse hyperbolic secant;
    - **acoth**(x) - inverse hyperbolic cotangent;
  - Logarithmic, exponential and roots:
    - **log**(x) - decimal logarithm;
    - **ln**(x) - natural logarithm;
    - **log_2**(x) - binary logarithm;

exp(x) - exponential function;

sqr(x) or **sqrt**(x) - square root;

cbrt(x) - cubic root;

root(x; n) - n-th root;

◦ Rounding:

round(x) - round to the nearest integer;

floor(x) - round to the smaller integer (towards -∞);

ceiling(x) - round to the greater integer (towards +∞);

trunc(x) - round to the smaller integer (towards zero);

◦ Integer:

mod(x; y) - the reminder of an integer division;

gcd(x; y) - the greatest common divisor of two integers;

lcm(x; y) - the least common multiple of two integers;

◦ Complex:

abs(x) - absolute value/magnitude;

re(x) - the real part of a complex number;

im(x) - the imaginary part of a complex number;

phase(x) - the phase of a complex number;

◦ Aggregate and interpolation:

min(x; y; z...) - minimum of multiple values;

max(x; y; z...) - maximum of multiple values;

sum(x; y; z...) - sum of multiple values = $x + y + z$...;

sumsq(x; y; z...) - sum of squares = $x^2 + y^2 + z^2$...;

srss(x; y; z...) - square root of sum of squares = **sqrt**($x^2 + y^2 + z^2$...);

average(x; y; z...) - average of multiple values = $(x + y + z...)/n$;

product(x; y; z...) - product of multiple values = $x{\cdot}y{\cdot}z$...;

mean(x; y; z...) - geometric mean = **n-th root**($x{\cdot}y{\cdot}z$...);

take(n; a; b; c...) - returns the n-th element from the list;

line(x; a; b; c...) - linear interpolation;

spline(x; a; b; c...) - Hermite spline interpolation;

◦ Conditional and logical:

if(*cond*; *value-if-true*; *value-if-false*) - conditional evaluation;

switch(*cond1*; *value1*; *cond2*; *value2*; ...; *default*) - selective evaluation;

not(x) - logical "NOT";

and(x; y; z...) - logical "AND";

or(x; y; z...) - logical "OR";

xor(x; y; z...) - logical "XOR";

◦ Other:

sign(x) - sign of a number;

random(x) - random number between 0 and x.

Comments: **"Title"** or **'text'** in double or single quotes, respectively.

HTML, **CSS**, **JS** and **SVG** are allowed.

- Graphing and plotting:

$Plot$\{ f(x) \ @ \ x = a : b \}$ - simple plot;

$Plot$\{ x(t) \mid y(t) \ @ \ t = a : b \}$ - parametric;

$Plot$\{ f_1(x) \ \& \ f_2(x) \ \& \ ... \ @ \ x = a : b \}$ - multiple;

$Plot$\{ x_1(t) \mid y_1(t) \ \& \ x_2(t) \mid y_2(t) \ \& \ ... \ @ \ x = a : b \}$ - multiple parametric;

Map\{ f(x; \ y) \ @ \ x = a : b \ \& \ y = c : d \}$ - 2D color map of a 3D surface;

PlotHeight - height of plot area in pixels;

PlotWidth - width of plot area in pixels;

PlotStep - grid size for map plotting.

- Iterative and numerical methods:

$Root$\{ f(x) = const \ @ \ x = a : b \}$ - root finding for $f(x) = const$;

$Root$\{ f(x) \ @ \ x = a : b \}$ - root finding for $f(x) = 0$;

$Find$\{ f(x) \ @ \ x = a : b \}$ - similar to above, but x is not required to be a precise solution;

Sup\{ f(x) \ @ \ x = a : b \}$ - local maximum of a function;

Inf\{ f(x) \ @ \ x = a : b \}$ - local minimum of a function;

$Area$\{ f(x) \ @ \ x = a : b \}$ - adaptive Gauss-Lobatto numerical integration;

$Integral$\{ f(x) \ @ \ x = a : b \}$ - Tanh-Sinh numerical integration;

$Slope$\{ f(x) \ @ \ x = a \}$ - numerical differentiation;

Sum\{ f(x) \ @ \ k = a : b \}$ - iterative sum;

$Product$\{ f(k) \ @ \ k = a : b \}$ - iterative product;

$Repeat$\{ f(k) \ @ \ k = a : b \}$ - general inline iterative procedure;

Precision - relative precision for numerical methods [10^{-2}; 10^{-16}] (default is 10^{-12})

- Program flow control:

Simple:

```
#if condition
    your code goes here
#end if
```

Alternative:

```
#if condition
    your code goes here
#else
    some other code
#end if
```

Complete:

```
#if condition1
    your code goes here
```

    #else if *condition2*

      your code goes here

    #else

      some other code

    #end if

You can add or omit as many "#else ifs" as needed. Only one "#else" is allowed. You can omit this too.
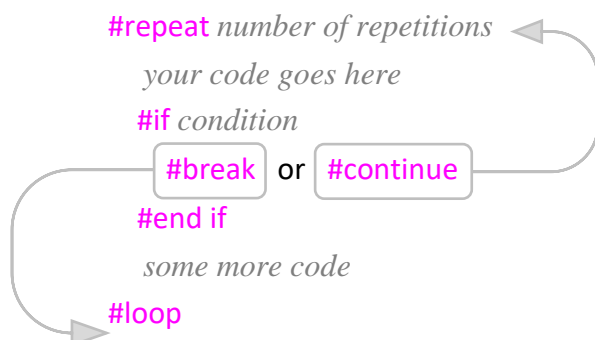
- Iteration blocks:

Simple:

    #repeat *number of repetitions*

      your code goes here

    #loop

With conditional break/continue:

    #repeat *number of repetitions*

      your code goes here

    #if *condition*

      #break or **#continue**

    #end if

      some more code

    #loop

- Modules and macros/string variables:

Modules:

    #include *filename* - include external file (module);

    #local - start local section (not to be included);

    #global - start global section (to be included);

Inline string variable:

    #def *variable_name*$ *= content*

Multiline string variable:

    #def *variable_name*$

      content line 1

      content line 2

      ...

    #end def

Inline macro:

    #def *macro_name*$(*param1*$; *param2*$; ...) *= content*

Multiline macro:

    #def *macro_name*$(*param1*$; *param2*$; ...)

      content line 1

      content line 2

      ...

    #end def

- Output control:

  #hide - hide the report contents;

  #show - always show the contents (default);

  #pre - show the next contents only before calculations;

  #post - show the next contents only after calculations;

  #val - show only the result, without the equation;

  #equ - show complete equations and results (default);

  #noc - show only equations without results (no calculations);

  #round *n* - rounds the output to *n* digits after the decimal point.

- Breakpoints for step-by-step execution:

  #pause - calculates down to the current line and waits for the user to resume manually;

  #input - renders an input form to the current line and waits for user input.

  Each of the above commands is effective after the current line until the end of the report or another command that overwrites it.

- Units for trigonometric functions: #deg - degrees, #rad - radians, #gra - gradians;
- Separator for target units: |;
- Return angles with units: *ReturnAngleUnits* = 1;
- Dimensionless: *%*, *‰*;
- Angle: *°*, *′*, *″*, *deg*, *rad*, *grad*, *rev*;
- Metric units (SI and compatible):

  Mass: *g*, *hg*, *kg*, *t*, *kt*, *Mt*, *Gt*, *dg*, *cg*, *mg*, *µg*, *Da* (or *u*);

  Length: *m*, *km*, *dm*, *cm*, *mm*, *µm*, *nm*, *pm*, *AU*, *ly*;

  Time: *s*, *ms*, *µs*, *ns*, *ps*, *min*, *h*, *d*, *w*, *y*;

  Frequency: *Hz*, *kHz*, *MHz*, *GHz*, *THz*, *mHz*, *µHz*, *nHz*, *pHz*, *rpm*;

  Speed: *kmh*;

  Electric current: *A*, *kA*, *MA*, *GA*, *TA*, *mA*, *µA*, *nA*, *pA*;

  Temperature: *°C*, *Δ°C*, *K*;

  Amount of substance: *mol*;

  Luminous intensity: *cd*;

  Area: *a*, *daa*, *ha*;

  Volume: *L*, *daL*, *hL*, *dL*, *cL*, *mL*, *µL*, *nL*, *pL*;

  Force: *N*, *daN*, *hN*, *kN*, *MN*, *GN*, *TN*, *gf*, *kgf*, *tf*, *dyn*;

  Moment: *Nm*, *kNm*;

  Pressure: *Pa*, *daPa*, *hPa*, *kPa*, *MPa*, *GPa*, *TPa*, *dPa*, *cPa*, *mPa*, *µPa*, *nPa*, *pPa*,
  bar, *mbar*, *µbar*, *atm*, *at*, *Torr*, *mmHg*;

  Viscosity: *P*, *cP*, *St*, *cSt*;

  Energy work: *J*, *kJ*, *MJ*, *GJ*, *TJ*, *mJ*, *µJ*, *nJ*, *pJ*,
  Wh, *kWh*, *MWh*, *GWh*, *TWh*, *mWh*, *µWh*, *nWh*, *pWh*,
  eV, *keV*, *MeV*, *GeV*, *TeV*, *PeV*, *EeV*, *cal*, *kcal*, *erg*;

  Power: *W*, *kW*, *MW*, *GW*, *TW*, *mW*, *µW*, *nW*, *pW*, *hpM*, *ks*,
  VA, *kVA*, *MVA*, *GVA*, *TVA*, *mVA*, *µVA*, *nVA*, *pVA*,

VAR, kVAR, MVAR, GVAR, TVAR, mVAR, µVAR, nVAR, pVAR;

Electric charge: *C, kC, MC, GC, TC, mC, µC, nC, pC, Ah, mAh;*

Potential: *V, kV, MV, GV, TV, mV, µV, nV, pV;*

Capacitance: *F, kF, MF, GF, TF, mF, µF, nF, pF;*

Resistance: *Ω, kΩ, MΩ, GΩ, TΩ, mΩ, µΩ, nΩ, pΩ;*

Conductance: *S, kS, MS, GS, TS, mS, µS, nS, pS, Ʊ, kƱ, MƱ, GƱ, TƱ, mƱ, µƱ, nƱ, pƱ;*

Magnetic flux: *Wb , kWb, MWb, GWb, TWb, mWb, µWb, nWb, pWb;*

Magnetic flux density: *T, kT, MT, GT, TT, mT, µT, nT, pT;*

Inductance: *H, kH, MH, GH, TH, mH, µH, nH, pH;*

Luminous flux: *lm;*

Illuminance: *lx;*

Radioactivity: *Bq, kBq, MBq, GBq, TBq, mBq, µBq, nBq, pBq, Ci, Rd;*

Absorbed dose: *Gy, kGy, MGy, GGy, TGy, mGy, µGy, nGy, pGy;*

Equivalent dose: *Sv, kSv, MSv, GSv, TSv, mSv, µSv, nSv, pSv;*

Catalytic activity: *kat;*

- Non-metric units (Imperial/US):

  Mass: *gr, dr, oz, lb* (or *lbm, lb_m*), *kipm* (or *kip_m*), *st, qr,*
  cwt (or *cwt_UK, cwt_US*), *ton* (or *ton_UK, ton_US*), *slug;*

  Length: *th, in, ft, yd, ch, fur, mi, ftm* (or *ftm_UK, ftm_US*),
  cable (or *cable _UK, cable _US*), *nmi, li, rod, pole, perch, lea;*

  Speed: *mph, knot;*

  Temperature: *°F, Δ°F, °R;*

  Area: *rood, ac;*

  Volume, fluid: *fl_oz, gi, pt, qt, gal, bbl,* or:
  fl_oz_UK, gi_UK, pt_UK, qt_UK, gal_UK, bbl_UK,
  fl_oz_US, gi_US, pt_US, qt_US, gal_US, bbl_US;

  Volume, dry: (US) *pt_dry*, (US) *qt_dry*, (US) *gal_dry* (US) *bbl_dry*,
  pk (or *pk_UK, pk_US*), *bu* (or *bu_UK, bu_US*);

  Force: *ozf* (or *oz_f*), *lbf* (or *lb_f*), *kip* (or *kipf, kip_f*), *tonf* (or *ton_f*), *pdl;*

  Pressure: *osi, osf psi, psf, ksi, ksf, tsi, tsf, inHg;*

  Energy/work: *BTU, therm* (or *therm_UK, therm_US*), *quad;*

  Power: *hp, hpE, hpS;*

- Custom units - *.Name* = expression.

  Names can include currency symbols: €, £, £, ¥, ¢, ₽, ₹, ₩, ₪.