GeoGebra Commands list

4.3.1 General commands

Relation

Relation[object a, object b]: shows a message box that tells us the relation of object *a* and object *b*.
Note: This command lets us find out whether two objects are equal, if a point lies on a line or conic, or if a line is tangent or a passing line to a conic.

Delete

Delete[object a]: Deletes an object *a* and all its dependants.

Element

Element[list L, number n]: n^{th} element of a list *L*

4.3.2. Boolean Commands

If[condition, a, b]: gives a copy of object *a* if *condition* evaluates to true, and a copy of object *b* if it evaluates to false.
If[condition, a]: gives a copy of object *a* if *condition* evaluates to true, and an undefined object if it evaluates to false.

4.3.3 Number

Length

Length[vector v]: Length of a vector v
Length[point A]: Length of the position vector of *A*
Length[function f, number x1, number x2]: Length of the function graph of *f* between numbers *x1* and *x2*
Length[function f, point A, point B]: Length of the function graph of *f* between two points *A* and *B* on the graph
Length[curve c, number t1, number t2]: Length of curve *c* between numbers *t1* and *t2*
Length[curve c, point A, point B]: Length of curve *c* between two points *A* and *B* on the curve
Length[list L]: Length of list *L* (number of elements in the list)

Area

Area[point A, point B, point C, ...]: Area of the polygon defined by the given points *A*, *B*, and *C*
Area[conic c]: Area of a conic section *c* (circle or ellipse)

Distance

Distance[point A, point B]: Distance of two points *A* and *B*
Distance[point A, line g]: Distance of a point *A* and a line *g*
Distance[line g, line h]: Distance of lines *g* and *h*. Note: The distance of intersecting lines is 0. This function is interesting for parallel lines.

Modulo Function

Mod[number a, number b]: Remainder when number *a* is divided by number *b*

Integer Division

Div[number a, number b]: Integer quotient when number *a* is divided by number *b*

Slope

Slope[line g]: Slope of a line *g*. Note: This command also draws the slope triangle whose size may be changed (see Properties dialog).

Curvature

Curvature[point A, function f]: Curvature of function *f* in point *A*
Curvature[point A, curve c]: Curvature of curve *c* in point *A*

Radius

Radius[circle c]: Radius of a circle *c*

Circumference

Circumference[conic c]: Returns the circumference of a conic section *c* (circle or ellipse)

Perimeter

Perimeter[polygon poly]: Perimeter of a polygon *poly*

Parameter

Parameter[parabola p]: Parameter of a parabola *p* (distance of directrix and focus)

FirstAxisLength

FirstAxisLength[conic c]: Length of a conic section *c*'s principal axis

SecondAxisLength

SecondAxisLength[conic c]: Length of a conic section *c*'s second axis

Excentricity

Excentricity[conic c]: Excentricity of a conic section *c*

Integral

Integral[function f, number a, number b]: Definite integral of function *f(x)* from number *a* to *b*. Note: This command also draws the area between the function graph of *f* and the *x*-axis.
Integral[function f, function g, number a, number b]: Definite integral of the difference of the functions *f(x) - g(x)* from number *a* to number *b*. Note: This command also draws the area between the function graphs of *f* and *g*.

LowerSum

LowerSum[function f, number a, number b, number n]: Lower sum of function *f* on the interval [*a, b*] with *n* rectangles. Note: This command draws the rectangles of the lower sum too.

Iteration

Iteration[function f, number x0, number n]: Iterates function *f n* times using the given start value *x0.*
Example: After defining f(x) = x^2 the command Iteration[f, 3, 2] gives you the result $(3^2)^2 = 81$

Minimum and Maximum

Min[number a, number b]: Minimum of the given numbers *a* and *b*
Max[number a, number b]: Maximum of the given numbers *a* and *b*

Affine Ratio

AffineRatio[point A, point B, point C]: Returns the affine ratio λ of three collinear points *A*, *B*, and *C*, where *C = A + λ * AB*

Cross Ratio

CrossRatio[point A, point B, point C, point D]: Cross ratio λ of four collinear points *A*, *B*, *C*, and *D*, where *λ = AffineRatio[B, C, D] / AffineRatio[A, C, D]*

4.3.4 Angle

Angle

Angle[vector v1, vector v2]: Angle between two vectors *v1* and *v2* (between 0 and 360°)
Angle[line g, line h]: Angle between the direction vectors of two lines *g* and *h* (between 0 and 360°)
Angle[point A, point B, point C]: Angle enclosed by *BA* and *BC* (between 0 and 360°). Point *B* is the apex.
Angle[point A, point B, angle alpha]: Angle of size α drawn from point *A* with vertex *B*. Note: Point *Rotate[A, alpha, B]* is created too.
Angle[conic c]: Angle of twist of a conic section *c*'s principle axis (see command Axes)
Angle[vector v]: Angle between *x*-axis and vector *v*
Angle[point A]: Angle between *x*-axis and position vector of point *A*
Angle[number n]: Converts a number *n* into an angle (result between 0 and 2pi)
Angle[polygon poly]: All inner angles of a polygon *poly*

4.3.5 Point

Point

Point[line g]: Point on line *g*
Point[conic c]: Point on conic section *c* (e.g. circle, ellipse, hyperbola)
Point[function f]: Point on function *f*
Point[polygon poly]: Point on polygon *poly*
Point[vector v]: Point on vector *v*
Point[point P, vector v]: Point *P* plus vector *v*

Midpoint and Center

Midpoint[point A, point B]: Midpoint of points *A* and *B*
Midpoint[segment s]: Midpoint of segment *s*
Center[conic c]: Center of a conic section *c* (e.g. circle, ellipse, hyperbola)

Focus

Focus[conic c]: (All) foci of a conic section *c*

Vertex

Vertex[conic c]: (All) vertices of a conic section *c*

Centroid

Centroid[polygon poly]: Centroid of a polygon *poly*

Intersect

Intersect[line g, line h]: Intersection point of lines *g* and *h*
Intersect[line g, conic c]: All Intersection points of line *g* and conic section *c* (max. 2)
Intersect[line g, conic c, number n]: n^{th} intersection point of line *g* and conic section *c*
Intersect[conic c1, conic c2]: All intersection points of conic sections *c1* and *c2* (max. 4)
Intersect[conic c1, conic c2, number n]: n^{th} intersection point of conic sections *c1* and *c2*
Intersect[polynomial f1, polynomial f2]: All intersection points of polynomials *f1* and *f2*
Intersect[polynomial f1, polynomial f2, number n]: n^{th} intersection point of polynomials *f1* and *f2*
Intersect[polynomial f, line g]: All intersection points of polynomial *f* and line *g*
Intersect[polynomial f, line g, number n]: n^{th} intersection point of polynomial *f* and line *g*
Intersect[function f, function g, point A]: Intersection point of functions *f* and *g* with initial point *A* (for Newton's method)
Intersect[function f, line g, point A]: Intersection point of function *f* and line *g* with initial point *A* (for Newton's method)

Root

Root[polynomial f]: All roots of polynomial *f* (as points)
Root[function f, number a]: One root of function *f* with initial value *a* (Newton's method)
Root[function f, number a, number b]: One root of function *f* on interval [*a*, *b*] (regula falsi)

Extremum

Extremum[polynomial f]: All local extrema of polynomial *f* (as points)

InflectionPoint

InflectionPoint[polynomial f]: All inflection points of polynomial *f*

4.3.6 Vector

Vector

Vector[point A, point B]: Vector from point *A* to point *B*
Vector[point A]: Position vector of point *A*

Direction

Direction[line g]: Direction vector of line *g*. Note: A line with equation *ax + by = c* has the direction vector *(b, - a)*.

UnitVector

UnitVector[line g]: Direction vector with length 1 of a line *g*
UnitVector[vector v]: Vector with length 1, same direction and orientation as the given vector *v*

PerpendicularVector

PerpendicularVector[line g]: Perpendicular vector of a line *g*. Note: A line with equation *ax + by = c* has the perpendicular vector *(a, b)*.
PerpendicularVector[vector v]: Perpendicular vector of a vector *v*. Note: A vector with coordinates *(a, b)* has the perpendicular vector *(- b, a)*.

UnitPerpendicularVector

UnitPerpendicularVector[line g]: Perpendicular vector with length 1 of a line *g*
UnitPerpendicularVector[vector v]: Perpendicular vector with length 1 of a vector *v*

Curvature Vector

CurvatureVector[point A, function f]: Curvature vector of function *f* in point *A*
CurvatureVector[point A, curve c]: Curvature vector of curve *c* in point *A*

4.3.7 Segment

Segment

Segment[point A, point B]: Segment between two points *A* and *B*
Segment[point A, number a]: Segment with length *a* and starting point *A*. Note: The endpoint of the segment is created too.

4.3.8 Ray

Ray

Ray[point A, point B]: Ray starting at point *A* through point *B*
Ray[point A, vector v]: Ray starting at point *A* with direction vector *v*

4.3.9 Polygon

Polygon

Polygon[point A, point B, point C,...]: Polygon defined by the given points *A, B, C,…*
Polygon[point A, point B, number n]: Regular polygon with *n* vertices (including points *A* and *B*)

4.3.10 Line

Line

Line[point A, point B]: Line through two points *A* and *B*
Line[point A, line g]: Line through point *A* parallel to line *g*
Line[point A, vector v]: Line through point *A* with direction vector *v*

Perpendicular

Perpendicular[point A, line g]: Line through point *A* perpendicular to line *g*
Perpendicular[point A, vector v]: Line through point *A* perpendicular to vector *v*

LineBisector

LineBisector[point A, point B]: Line bisector of the line segment *AB*
LineBisector[segment s]: Line bisector of the segment *s*

AngularBisector

AngularBisector[point A, point B, point C]: Angular bisector of the angle defined by points *A, B,* and *C*. Note: Point *B* is apex of this angle.
AngularBisector[line g, line h]: Both angular bisectors of lines *g* and *h*.

Tangent

Tangent[point A, conic c]: (All) tangents through point *A* to conic section *c*

Tangent[line g, conic c]: (All) tangents to conic section *c* that are parallel to line *g*
Tangent[number a, function f]: Tangent to function *f(x)* at *x = a*
Tangent[point A, function f]: Tangent to function *f(x)* at *x = x(A)*
Tangent[point A, curve c]: Tangent to curve *c* in point *A*

Asymptote

Asymptote[hyperbola h]: Both asymptotes of a hyperbola *h*

Directrix

Directrix[parabola p]: Directrix of a parabola *p*

Axes

Axes[conic c]: Principal and second axis of a conic section *c*

FirstAxis

FirstAxis[conic c]: Principal axis of a conic section *c*

SecondAxis

SecondAxis[conic c]: Second axis of a conic section *c*

Polar

Polar[point A, conic c]: Polar line of point *A* relative to conic section *c*

Diameter

Diameter[line g , conic c]: Diameter parallel to line *g* relative to conic section *c*
Diameter[vector v, conic c]: Diameter with direction vector *v* relative to conic section *c*

4.3.11 Conic section

Circle

Circle[point M, number r]: Circle with midpoint *M* and radius *r*
Circle[point M, segment s]: Circle with midpoint *M* and radius equal to *Length[s]*
Circle[point M, point A]: Circle with midpoint *M* through point *A*
Circle[point A, point B, point C]: Circle through three points *A*, *B* and *C*

Osculating Circle

OsculatingCircle[point A, function f]: Osculating circle of function *f* in point *A*
OsculatingCircle[point A, curve c]: Osculating circle of curve *c* in point *A*

Ellipse

Ellipse[point F, point G, number a]: Ellipse with focal points *F* and *G* and principal axis length *a*. Note: Condition: $2a > Distance[F, G]$
Ellipse[point F, point G, segment s]: Ellipse with focal points *F* and *G* where the length of the principal axis equals the length of segment *s* (*a = Length[s]*).

Hyperbola

Hyperbola[point F, point G, number a]: Hyperbola with focal points *F* and *G* and principal axis length *a*. Note: Condition: $0 < 2a < Distance[F, G]$
Hyperbola[point F, point G, segment s]: Hyperbola with focal points *F* and *G* where the length of the principal axis equals the length of segment *s* (*a = Length[s]*)

Parabola

Parabola[point F, line g]: Parabola with focal point *F* and directrix *g*

Conic

Conic[point A, point B, point C, point D, point E]: Conic section through five points *A*, *B*, *C*, *D*, and *C*. Note: No four of the points lie on one line.

4.3.12 Function

Derivative

Derivative[function f]: Derivative of function *f(x)*
Derivative[function f, number n]: n^{th} derivative of function *f(x)*

Integral

Integral[function f]: Indefinite integral for function *f(x)*

Integral

Integral[function f, number a, number b]: Definite integral of function *f(x)* from number *a* to *b*. Note: This command also draws the area between the function graph of *f* and the *x*-axis.
Integral[function f, function g, number a, number b]: Definite integral of the difference of the functions *f(x) - g(x)* from number *a* to number *b*. Note: This command also draws the area between the function graphs of *f* and *g*.

Polynomial

Polynomial[function f]: Expanded polynomial function *f*.
Example: Polynomial[(x - 3)^2] yields $x^2 - 6x + 9$

TaylorPolynomial

TaylorPolynomial[function f, number a, number n]: Power series expansion for function *f* about the point $x = a$ to order *n*

Function

Function[function f, number a, number b]: Function, that is equal to *f* on the interval [*a*, *b*] and not defined outside of [*a*, *b*]

Conditional Function

You can use the Boolean command If (see command If) in order to create a conditional function.
Note: You can use derivatives and integrals of such functions and intersect them like "normal" functions.

Example:
f(x) = If[x < 3, sin(x), x^2] gives you a function that equals

- sin(x) for x < 3 and
- x^2 for $x \geq 3$.

4.3.13. Parametric Curves

Curve[expression e1, expression e2, parameter t, number a, number b]: Cartesian parametric curve for the given *x*-expression *e1* and *y*-expression *e2* (using parameter *t*) within the given interval [*a*, *b*]
Example: c = Curve[2 cos(t), 2 sin(t), t, 0, 2 pi]

Derivative[curve c]: Derivative of the curve *c*

Note: Parametric curves can be used like functions in arithmetic expressions.
Example: Input c(3) returns the point at parameter position 3 on curve *c*.

Note: Using the mouse you can also place a point on a curve using the mode *New point* (see mode New point; also see command Point). Since the parameters *a* and *b* are dynamic you could use slider variables there (see mode Slider).

4.3.14 Arc and sector

Semicircle

Semicircle[point A, point B]: Semicircle above the segment AB.

CircularArc

CircularArc[point M, point A, point B]: Circular arc with midpoint *M* between points *A* and *B*. Note: Point *B* does not have to lie on the arc.

CircumcircularArc

CircumcircularArc[point A, point B, point C]: Circular arc through three points *A*, *B*, and *C*

Arc

Arc[conic c, point A, point B]: Conic section arc between two points *A* and *B* on the conic section *c* (circle or ellipse)
Arc[conic c, number t1, number t2]: Conic section arc between two parameter values *t1* and *t2* on the conic section *c* for the following parameter forms:

- Circle: *(r cos(t), r sin(t))* where *r* is the circle's radius
- Ellipse: *(a cos(t), b sin(t))* where *a* and *b* are the lengths of the first and second axis

CircularSector

CircularSector[point M, point A, point B]: Circular sector with midpoint *M* between two points *A* and *B*. Note: point *B* does not have to lie on the arc.

CircumcircularSector

CircumcircularSector[point A, point B, point C]: Circular sector through three points *A*, *B*, and *C*

Sector

Sector[conic c, point A, point B]: Conic section sector between two points *A* and *B* on the conic section *c* (circle or ellipse)
Sector[conic c, number t1, number t2]: Conic section sector between two parameter values *t1* and *t2* on the conic section *c* for the following parameter forms:

- Circle: *(r cos(t), r sin(t))* where *r* is the circle's radius
- Ellipse: *(a cos(t), b sin(t))* where *a* and *b* are the lengths of the first and second axis

4.3.15 Image

Corner

Corner[image pic, number n]: n^{th} corner of an image pic with a maximum of 4 corners

4.3.16 Text

Name

Name[object]: Text showing the name of the given object
Note: Use this command in dynamic texts for objects that might be renamed

4.3.17 Locus

Locus

Locus[point Q, point P]: Locus line of point *Q* that depends on point *P*.
Note: Point *P* has to be a point on an object (e.g. line, segment, circle).

4.3.18 Sequence

Sequence

Sequence[expression e, variable i, number a, number b]: List of objects created using expression *e* and index *i* that ranges from number *a* to number *b*.
Example: L = Sequence[(2, i), i, 1, 5] creates a list of points whose *y*-coordinates range from 1 to 5
Sequence[expression e, variable i, number a, number b, number s]: List of objects created using expression *e* and index *i* that ranges from number *a* to number *b* with given step size *s*.
Example: L = Sequence[(2, i), i, 1, 5, 0.5] creates a list of points whose *y*-coordinates range from 1 to 5 with a step size of 0.5.
Note: Since the parameters *a* and *b* are dynamic you could use slider variables there.

Other Sequence Commands

Element[list L, number n]: n^{th} element of a list *L*
Length[list L]: Length of a list *L*
Min[list L]: Minimal element of a list *L*
Max[list L]: Maximal element of a list *L*

Iteration

IterationList[function f, number x0, number n]:
List *L* of length *n*+1 whose elements are iterations of function *f* starting with the value *x0*.
Example: After defining function f(x) = x^2 the command L = IterationList[f, 3, 2] gives you the list
$L = \{3, 3^2, (3^2)^2\} = \{3, 9, 81\}$

4.3.19 Geometric transformations

Translate

Translate[point A, vector v]: Translates point *A* by vector *v*
Translate[line g, vector v]: Translates line *g* by vector *v*
Translate[conic c, vector v]: Translates conic *c* by vector *v*
Translate[function c, vector v]: Translates function *f* by the vector *v*
Translate[polygon poly, vector v]: Translates polygon *poly* by vector *v*. Note: New vertices and segments are created too.
Translate[image pic, vector v]: Translates image *pic* by vector *v*
Translate[vector v, Point P]: Translates vector *v* to point *P*

Rotate

Rotate[point A, angle phi]: Rotates point *A* by angle *φ* around the axis origin
Rotate[vector v, angle phi]: Rotates vector *v* by angle *φ*
Rotate[line g, angle phi]: Rotates line *g* by angle *φ* around the axis origin
Rotate[conic c, angle phi]: Rotates conic section *c* by angle *φ* around the axis origin
Rotate[polygon poly, angle phi]: Rotates polygon *poly* by angle *φ* around the axis origin. Note: New vertices and segments are created too.
Rotate[image pic, angle phi]: Rotates image *pic* by angle *φ* around the axis origin
Rotate[point A, angle phi, point B]: Rotates point *A* by angle *φ* around point *B*
Rotate[line g, angle phi, point B]: Rotates line *g* by angle *φ* around point *B*
Rotate[conic c, angle phi, point B]: Rotates conic section *c* by angle *φ* around point *B*
Rotate[polygon poly, angle phi, point B]: Rotates polygon *poly* by angle *φ* around point *B*. Note: New vertices and segments are created too.
Rotate[image pic, angle phi, point B]: Rotates image *pic* by angle *φ* around point *B*

Mirror

Mirror[point A, point B]: Mirror point *A* at point *B*
Mirror[line g, point B]: Mirror line *g* at point *B*
Mirror[conic c, point B]: Mirror conic section *c* at point *B*
Mirror[polygon poly, point B]: Mirror polygon *poly* at point B. Note: New vertices and segments are created too.
Mirror[image pic, point B]: Mirror image *pic* at point *B*
Mirror[point A, line h]: Mirror point *A* at line *h*
Mirror[line g, line h]: Mirror line *g* at line *h*
Mirror[conic c, line h]: Mirror conic *c* at line *h*
Mirror[polygon poly, line h]: Mirror polygon *poly* at line *h*. Note: New vertices and segments are created too.
Mirror[image pic, line h]: Mirror image *pic* at line *h*

Note: Also see mode  Mirror object at point; mode  Mirror object at line

Dilate

Dilate[point A, number f, point S]: Dilates point *A* from point *S* using factor *f*
Dilate[line h, number f, point S]: Dilates line *h* from point *S* using factor *f*
Dilate[conic c, number f, point S]: Dilates conic section *c* from point *S* using factor *f*
Dilate[polygon poly, number f, point S]: Dilates polygon *poly* from point *S* using factor *f*. Note: New vertices and segments are created too.
Dilate[image pic, number f, point S]: Dilates image *pic* from point *S* using factor *f*

Note: Also see mode  Dilate object from point by vector