Skip to content

Curve Operators

bpy.ops.curve.**cyclic_toggle(*, direction='CYCLIC_U')**

Make active spline closed/opened loop

PARAMETERS:

direction (*enum in ['CYCLIC_U', 'CYCLIC_V'], (optional)*) – Direction, Direction to make surface cyclic in

bpy.ops.curve.**de_select_first()**

(De)select first of visible part of each NURBS

bpy.ops.curve.**de_select_last()**

(De)select last of visible part of each NURBS

bpy.ops.curve.**decimate(*, ratio=1.0)**

Simplify selected curves

PARAMETERS:

ratio (*float in [0, 1], (optional)*) – Ratio

bpy.ops.curve.**delete(*, type='VERT')**

Delete selected control points or segments

PARAMETERS:

type (*enum in ['VERT', 'SEGMENT'], (optional)*) – Type, Which elements to delete

bpy.ops.curve.**dissolve_verts()**

Delete selected control points, correcting surrounding handles

bpy.ops.curve.**draw(*, error_threshold=0.0, fit_method='REFIT', corner_angle=1.22173, use_cyclic=True, stroke=None, wait_for_input=True)**

Draw a freehand spline

PARAMETERS:

- **error_threshold** (*float in [0, 10], (optional)*) – Error, Error distance threshold (in object units)
- **fit_method** (enum in Curve Fit Method Items, (optional)) – Fit Method
- **corner_angle** (*float in [0, 3.14159], (optional)*) – Corner Angle
- **use_cyclic** (*boolean, (optional)*) – Cyclic
- **stroke** (`bpy_prop_collection` of `OperatorStrokeElement`, (optional)) – Stroke
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.curve.**duplicate()**

Duplicate selected control points

bpy.ops.curve.**duplicate_move(*, CURVE_OT_duplicate=None, TRANSFORM_OT_translate=None)**

Duplicate curve and move

PARAMETERS:

- **CURVE_OT_duplicate** (`CURVE_OT_duplicate`, (optional)) – Duplicate Curve, Duplicate selected control points
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.curve.**extrude(*, mode='TRANSLATION')**

Extrude selected control point(s)

PARAMETERS:

mode (enum in Transform Mode Type Items, (optional)) – Mode

bpy.ops.curve.**extrude_move(*, CURVE_OT_extrude=None, TRANSFORM_OT_translate=None)**

Extrude curve and move result

PARAMETERS:

- **CURVE_OT_extrude** (`CURVE_OT_extrude`, (optional)) – Extrude, Extrude selected control point(s)
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.curve.**handle_type_set(*, type='AUTOMATIC')**

Set type of handles for selected control points

PARAMETERS:

type (*enum in ['AUTOMATIC', 'VECTOR', 'ALIGNED', 'FREE_ALIGN', 'TOGGLE_FREE_ALIGN'], (optional)*) – Type, Spline type

bpy.ops.curve.**hide(*, unselected=False)**

Hide (un)selected control points

PARAMETERS:

unselected (*boolean, (optional)*) – Unselected, Hide unselected rather than selected

bpy.ops.curve.**make_segment()**

Join two curves by their selected ends

bpy.ops.curve.**match_texture_space()**

Match texture space to object's bounding box

bpy.ops.curve.**normals_make_consistent(*, calc_length=False)**

Recalculate the direction of selected handles

PARAMETERS:

calc_length (*boolean, (optional)*) – Length, Recalculate handle length

bpy.ops.curve.**pen(*, extend=False, deselect=False, toggle=False, deselect_all=False, select_passthrough=False, extrude_point=False, extrude_handle='VECTOR', delete_point=False, insert_point=False, move_segment=False, select_point=False, move_point=False close_spline=True, close_spline_method='OFF', toggle_vector=False, cycle_handle_type=False)**

Construct and edit splines

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **deselect** (*boolean, (optional)*) – Deselect, Remove from selection
- **toggle** (*boolean, (optional)*) – Toggle Selection, Toggle the selection
- **deselect_all** (*boolean, (optional)*) – Deselect On Nothing, Deselect all when nothing under the cursor
- **select_passthrough** (*boolean, (optional)*) – Only Select Unselected, Ignore the select action when the element is already selected
- **extrude_point** (*boolean, (optional)*) – Extrude Point, Add a point connected to the last selected point
- **extrude_handle** (*enum in ['AUTO', 'VECTOR'], (optional)*) – Extrude Handle Type, Type of the extruded handle
- **delete_point** (*boolean, (optional)*) – Delete Point, Delete an existing point
- **insert_point** (*boolean, (optional)*) – Insert Point, Insert Point into a curve segment
- **move_segment** (*boolean, (optional)*) – Move Segment, Delete an existing point
- **select_point** (*boolean, (optional)*) – Select Point, Select a point or its handles
- **move_point** (*boolean, (optional)*) – Move Point, Move a point or its handles
- **close_spline** (*boolean, (optional)*) – Close Spline, Make a spline cyclic by clicking endpoints
- **close_spline_method** (*enum in ['OFF', 'ON_PRESS', 'ON_CLICK'], (optional)*) –

  Close Spline Method, The condition for close spline to activate

Close Spline Method, The condition for close spline to activate

- `OFF` None.
- `ON_PRESS` On Press – Move handles after closing the spline.
- `ON_CLICK` On Click – Spline closes on release if not dragged.

- **toggle_vector** (*boolean, (optional)*) – Toggle Vector, Toggle between Vector and Auto handles
- **cycle_handle_type** (*boolean, (optional)*) – Cycle Handle Type, Cycle between all four handle types

bpy.ops.curve.**primitive_bezier_circle_add(*, radius=1.0, enter_editmode=False, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Construct a Bézier Circle

PARAMETERS:

- **radius** (*float in [0, inf], (optional)*) – Radius
- **enter_editmode** (*boolean, (optional)*) – Enter Edit Mode, Enter edit mode when adding this object
- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –
  Align, The alignment of the new object
  - `WORLD` World – Align the new object to the world.
  - `VIEW` View – Align the new object to the view.
  - `CURSOR` 3D Cursor – Use the 3D cursor orientation for the new object.
- **location** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Location, Location for the newly added object
- **rotation** (`mathutils.Euler` rotation of 3 items in [-inf, inf], (optional)) – Rotation, Rotation for the newly added object
- **scale** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Scale, Scale for the newly added object

bpy.ops.curve.**primitive_bezier_curve_add(*, radius=1.0, enter_editmode=False, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Construct a Bézier Curve

PARAMETERS:

- **radius** (*float in [0, inf], (optional)*) – Radius
- **enter_editmode** (*boolean, (optional)*) – Enter Edit Mode, Enter edit mode when adding this object
- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –
  Align, The alignment of the new object
  - `WORLD` World – Align the new object to the world.
  - `VIEW` View – Align the new object to the view.
  - `CURSOR` 3D Cursor – Use the 3D cursor orientation for the new object.
- **location** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Location, Location for the newly added object
- **rotation** (`mathutils.Euler` rotation of 3 items in [-inf, inf], (optional)) – Rotation, Rotation for the newly added object
- **scale** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Scale, Scale for the newly added object

bpy.ops.curve.**primitive_nurbs_circle_add(*, radius=1.0, enter_editmode=False, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Construct a Nurbs Circle

PARAMETERS:

- **radius** (*float in [0, inf], (optional)*) – Radius
- **enter_editmode** (*boolean, (optional)*) – Enter Edit Mode, Enter edit mode when adding this object
- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –
  Align, The alignment of the new object
  - `WORLD` World – Align the new object to the world.
  - `VIEW` View – Align the new object to the view.

- CURSOR 3D Cursor – Use the 3D cursor orientation for the new object.

- **location** (mathutils.Vector of 3 items in [-inf, inf], (optional)) – Location, Location for the newly added object
- **rotation** (mathutils.Euler rotation of 3 items in [-inf, inf], (optional)) – Rotation, Rotation for the newly added object
- **scale** (mathutils.Vector of 3 items in [-inf, inf], (optional)) – Scale, Scale for the newly added object

bpy.ops.curve.**primitive_nurbs_curve_add(*, radius=1.0, enter_editmode=False, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Construct a Nurbs Curve

PARAMETERS:

- **radius** (*float in [0, inf], (optional)*) – Radius
- **enter_editmode** (*boolean, (optional)*) – Enter Edit Mode, Enter edit mode when adding this object
- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –
  Align, The alignment of the new object
  - WORLD World – Align the new object to the world.
  - VIEW View – Align the new object to the view.
  - CURSOR 3D Cursor – Use the 3D cursor orientation for the new object.
- **location** (mathutils.Vector of 3 items in [-inf, inf], (optional)) – Location, Location for the newly added object
- **rotation** (mathutils.Euler rotation of 3 items in [-inf, inf], (optional)) – Rotation, Rotation for the newly added object
- **scale** (mathutils.Vector of 3 items in [-inf, inf], (optional)) – Scale, Scale for the newly added object

bpy.ops.curve.**primitive_nurbs_path_add(*, radius=1.0, enter_editmode=False, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Construct a Path

PARAMETERS:

- **radius** (*float in [0, inf], (optional)*) – Radius
- **enter_editmode** (*boolean, (optional)*) – Enter Edit Mode, Enter edit mode when adding this object
- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –
  Align, The alignment of the new object
  - WORLD World – Align the new object to the world.
  - VIEW View – Align the new object to the view.
  - CURSOR 3D Cursor – Use the 3D cursor orientation for the new object.
- **location** (mathutils.Vector of 3 items in [-inf, inf], (optional)) – Location, Location for the newly added object
- **rotation** (mathutils.Euler rotation of 3 items in [-inf, inf], (optional)) – Rotation, Rotation for the newly added object
- **scale** (mathutils.Vector of 3 items in [-inf, inf], (optional)) – Scale, Scale for the newly added object

bpy.ops.curve.**radius_set(*, radius=1.0)**

Set per-point radius which is used for bevel tapering

PARAMETERS:

radius (*float in [0, inf], (optional)*) – Radius

bpy.ops.curve.**reveal(*, select=True)**

Reveal hidden control points

PARAMETERS:

select (*boolean, (optional)*) – Select

bpy.ops.curve.**select_all(*, action='TOGGLE')**

(De)select all control points

PARAMETERS:

action (*enum in ['TOGGLE', 'SELECT', 'DESELECT', 'INVERT'], (optional)*) –

Action, Selection action to execute

- `TOGGLE` Toggle – Toggle selection for all elements.
- `SELECT` Select – Select all elements.
- `DESELECT` Deselect – Deselect all elements.
- `INVERT` Invert – Invert selection of all elements.

bpy.ops.curve.**select_less()**

Deselect control points at the boundary of each selection region

bpy.ops.curve.**select_linked()**

Select all control points linked to the current selection

bpy.ops.curve.**select_linked_pick(*, deselect=False)**

Select all control points linked to already selected ones

PARAMETERS:

deselect (*boolean, (optional)*) – Deselect, Deselect linked control points rather than selecting them

bpy.ops.curve.**select_more()**

Select control points at the boundary of each selection region

bpy.ops.curve.**select_next()**

Select control points following already selected ones along the curves

bpy.ops.curve.**select_nth(*, skip=1, nth=1, offset=0)**

Deselect every Nth point starting from the active one

PARAMETERS:

- **skip** (*int in [1, inf], (optional)*) – Deselected, Number of deselected elements in the repetitive sequence
- **nth** (*int in [1, inf], (optional)*) – Selected, Number of selected elements in the repetitive sequence
- **offset** (*int in [-inf, inf], (optional)*) – Offset, Offset from the starting point

bpy.ops.curve.**select_previous()**

Select control points preceding already selected ones along the curves

bpy.ops.curve.**select_random(*, ratio=0.5, seed=0, action='SELECT')**

Randomly select some control points

PARAMETERS:

- **ratio** (*float in [0, 1], (optional)*) – Ratio, Portion of items to select randomly
- **seed** (*int in [0, inf], (optional)*) – Random Seed, Seed for the random number generator
- **action** (*enum in ['SELECT', 'DESELECT'], (optional)*) –

  Action, Selection action to execute

  - `SELECT` Select – Select all elements.
  - `DESELECT` Deselect – Deselect all elements.

bpy.ops.curve.**select_row()**

Select a row of control points including active one. Successive use on the same point switches between U/V directions

bpy.ops.curve.**select_similar(*, type='WEIGHT', compare='EQUAL', threshold=0.1)**

Select similar curve points by property type

PARAMETERS:

- **type** (*enum in ['TYPE', 'RADIUS', 'WEIGHT', 'DIRECTION'], (optional)*) – Type
- **compare** (*enum in ['EQUAL', 'GREATER', 'LESS'], (optional)*) – Compare
- **threshold** (*float in [0, inf], (optional)*) – Threshold

bpy.ops.curve.**separate()**

Separate selected points from connected unselected points into a new object

bpy.ops.curve.**shade_flat()**

Set shading to flat

bpy.ops.curve.**shade_smooth()**

Set shading to smooth

bpy.ops.curve.**shortest_path_pick()**

Select shortest path between two selections

bpy.ops.curve.**smooth()**

Flatten angles of selected points

bpy.ops.curve.**smooth_radius()**

Interpolate radii of selected points

bpy.ops.curve.**smooth_tilt()**

Interpolate tilt of selected points

bpy.ops.curve.**smooth_weight()**

Interpolate weight of selected points

bpy.ops.curve.**spin(*, center=(0.0, 0.0, 0.0), axis=(0.0, 0.0, 0.0))**

Extrude selected boundary row around pivot point and current view axis

PARAMETERS:

- **center** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center, Center in global view space
- **axis** (`mathutils.Vector` of 3 items in [-1, 1], (optional)) – Axis, Axis in global view space

bpy.ops.curve.**spline_type_set(*, type='POLY', use_handles=False)**

Set type of active spline

PARAMETERS:

- **type** (*enum in ['POLY', 'BEZIER', 'NURBS'], (optional)*) – Type, Spline type
- **use_handles** (*boolean, (optional)*) – Handles, Use handles when converting Bézier curves into polygons

bpy.ops.curve.**spline_weight_set(*, weight=1.0)**

Set softbody goal weight for selected points

PARAMETERS:

weight (*float in [0, 1], (optional)*) – Weight

bpy.ops.curve.**split()**

Split off selected points from connected unselected points

bpy.ops.curve.**subdivide(*, number_cuts=1)**

Subdivide selected segments

PARAMETERS:

number_cuts (*int in [1, 1000], (optional)*) – Number of Cuts

bpy.ops.curve.**switch_direction()**

Switch direction of selected splines

bpy.ops.curve.**tilt_clear()**

Clear the tilt of selected control points

bpy.ops.curve.**vertex_add(*, location=(0.0, 0.0, 0.0))**

Add a new control point (linked to only selected end-curve one, if any)

PARAMETERS:

location (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Location, Location to add new vertex at