Mesh Operators

bpy.ops.mesh.**attribute_set(*, value_float=0.0, value_float_vector_2d=(0.0, 0.0), value_float_vector_3d=(0.0, 0.0, 0.0), value_int=0, value_int_vector_2d=(0, 0), value_color=(1.0, 1.0, 1.0, 1.0), value_bool=False)**

Set values of the active attribute for selected elements

PARAMETERS:

- **value_float** (*float in [-inf, inf], (optional)*) – Value
- **value_float_vector_2d** (*float array of 2 items in [-inf, inf], (optional)*) – Value
- **value_float_vector_3d** (*float array of 3 items in [-inf, inf], (optional)*) – Value
- **value_int** (*int in [-inf, inf], (optional)*) – Value
- **value_int_vector_2d** (*int array of 2 items in [-inf, inf], (optional)*) – Value
- **value_color** (*float array of 4 items in [-inf, inf], (optional)*) – Value
- **value_bool** (*boolean, (optional)*) – Value

bpy.ops.mesh.**average_normals(*, average_type='CUSTOM_NORMAL', weight=50, threshold=0.01)**

Average custom normals of selected vertices

PARAMETERS:

- **average_type** (*enum in ['CUSTOM_NORMAL', 'FACE_AREA', 'CORNER_ANGLE'], (optional)*) –
  Type, Averaging method
  - `CUSTOM_NORMAL` Custom Normal – Take average of vertex normals.
  - `FACE_AREA` Face Area – Set all vertex normals by face area.
  - `CORNER_ANGLE` Corner Angle – Set all vertex normals by corner angle.
- **weight** (*int in [1, 100], (optional)*) – Weight, Weight applied per face
- **threshold** (*float in [0, 10], (optional)*) – Threshold, Threshold value for different weights to be considered equal

bpy.ops.mesh.**beautify_fill(*, angle_limit=3.14159)**

Rearrange some faces to try to get less degenerated geometry

PARAMETERS:

angle_limit (*float in [0, 3.14159], (optional)*) – Max Angle, Angle limit

bpy.ops.mesh.**bevel(*, offset_type='OFFSET', offset=0.0, profile_type='SUPERELLIPSE', offset_pct=0.0, segments=1, profile=0.5, affect='EDGES', clamp_overlap=False, loop_slide=True, mark_seam=False, mark_sharp=False, material=-1, harden_normals=False, face_strength_mode='NONE', miter_outer='SHARP', miter_inner='SHARP', spread=0.1, vmesh_method='ADJ', release_confirm=False)**

Cut into selected items at an angle to create bevel or chamfer

PARAMETERS:

- **offset_type** (*enum in ['OFFSET', 'WIDTH', 'DEPTH', 'PERCENT', 'ABSOLUTE'], (optional)*) –
  Width Type, The method for determining the size of the bevel
  - `OFFSET` Offset – Amount is offset of new edges from original.
  - `WIDTH` Width – Amount is width of new face.
  - `DEPTH` Depth – Amount is perpendicular distance from original edge to bevel face.
  - `PERCENT` Percent – Amount is percent of adjacent edge length.
  - `ABSOLUTE` Absolute – Amount is absolute distance along adjacent edge.
- **offset** (*float in [0, 1e+06], (optional)*) – Width, Bevel amount
- **profile_type** (*enum in ['SUPERELLIPSE', 'CUSTOM'], (optional)*) –
  Profile Type, The type of shape used to rebuild a beveled section

- SUPERELLIPSE Superellipse – The profile can be a concave or convex curve.
- CUSTOM Custom – The profile can be any arbitrary path between its endpoints.

- **offset_pct** (*float in [0, 100], (optional)*) – Width Percent, Bevel amount for percentage method
- **segments** (*int in [1, 1000], (optional)*) – Segments, Segments for curved edge
- **profile** (*float in [0, 1], (optional)*) – Profile, Controls profile shape (0.5 = round)
- **affect** (*enum in ['VERTICES', 'EDGES'], (optional)*) –
  Affect, Affect edges or vertices
  - VERTICES Vertices – Affect only vertices.
  - EDGES Edges – Affect only edges.
- **clamp_overlap** (*boolean, (optional)*) – Clamp Overlap, Do not allow beveled edges/vertices to overlap each other
- **loop_slide** (*boolean, (optional)*) – Loop Slide, Prefer sliding along edges to even widths
- **mark_seam** (*boolean, (optional)*) – Mark Seams, Mark Seams along beveled edges
- **mark_sharp** (*boolean, (optional)*) – Mark Sharp, Mark beveled edges as sharp
- **material** (*int in [-1, inf], (optional)*) – Material Index, Material for bevel faces (-1 means use adjacent faces)
- **harden_normals** (*boolean, (optional)*) – Harden Normals, Match normals of new faces to adjacent faces
- **face_strength_mode** (*enum in ['NONE', 'NEW', 'AFFECTED', 'ALL'], (optional)*) –
  Face Strength Mode, Whether to set face strength, and which faces to set face strength on
  - NONE None – Do not set face strength.
  - NEW New – Set face strength on new faces only.
  - AFFECTED Affected – Set face strength on new and modified faces only.
  - ALL All – Set face strength on all faces.
- **miter_outer** (*enum in ['SHARP', 'PATCH', 'ARC'], (optional)*) –
  Outer Miter, Pattern to use for outside of miters
  - SHARP Sharp – Outside of miter is sharp.
  - PATCH Patch – Outside of miter is squared-off patch.
  - ARC Arc – Outside of miter is arc.
- **miter_inner** (*enum in ['SHARP', 'ARC'], (optional)*) –
  Inner Miter, Pattern to use for inside of miters
  - SHARP Sharp – Inside of miter is sharp.
  - ARC Arc – Inside of miter is arc.
- **spread** (*float in [0, 1e+06], (optional)*) – Spread, Amount to spread arcs for arc inner miters
- **vmesh_method** (*enum in ['ADJ', 'CUTOFF'], (optional)*) –
  Vertex Mesh Method, The method to use to create meshes at intersections
  - ADJ Grid Fill – Default patterned fill.
  - CUTOFF Cutoff – A cutoff at each profile's end before the intersection.
- **release_confirm** (*boolean, (optional)*) – Confirm on Release

bpy.ops.mesh.**bisect(*, plane_co=(0.0, 0.0, 0.0), plane_no=(0.0, 0.0, 0.0), use_fill=False, clear_inner=False, clear_outer=False, threshold=0.0001, xstart=0, xend=0, ystart=0, yend=0, flip=False, cursor=5)**

Cut geometry along a plane (click-drag to define plane)

PARAMETERS:

- **plane_co** (mathutils.Vector of 3 items in [-inf, inf], (optional)) – Plane Point, A point on the plane
- **plane_no** (mathutils.Vector of 3 items in [-1, 1], (optional)) – Plane Normal, The direction the plane points
- **use_fill** (*boolean, (optional)*) – Fill, Fill in the cut
- **clear_inner** (*boolean, (optional)*) – Clear Inner, Remove geometry behind the plane

- **clear_outer** (*boolean, (optional)*) – Clear Outer, Remove geometry in front of the plane
- **threshold** (*float in [0, 10], (optional)*) – Axis Threshold, Preserves the existing geometry along the cut plane
- **xstart** (*int in [-inf, inf], (optional)*) – X Start
- **xend** (*int in [-inf, inf], (optional)*) – X End
- **ystart** (*int in [-inf, inf], (optional)*) – Y Start
- **yend** (*int in [-inf, inf], (optional)*) – Y End
- **flip** (*boolean, (optional)*) – Flip
- **cursor** (*int in [0, inf], (optional)*) – Cursor, Mouse cursor style to use during the modal operator

bpy.ops.mesh.**blend_from_shape(*, shape='', blend=1.0, add=True)**

Blend in shape from a shape key

PARAMETERS:

- **shape** (*enum in [], (optional)*) – Shape, Shape key to use for blending
- **blend** (*float in [-1000, 1000], (optional)*) – Blend, Blending factor
- **add** (*boolean, (optional)*) – Add, Add rather than blend between shapes

bpy.ops.mesh.**bridge_edge_loops(*, type='SINGLE', use_merge=False, merge_factor=0.5, twist_offset=0, number_cuts=0, interpolation='PATH', smoothness=1.0, profile_shape_factor=0.0, profile_shape='SMOOTH')**

Create a bridge of faces between two or more selected edge loops

PARAMETERS:

- **type** (*enum in ['SINGLE', 'CLOSED', 'PAIRS'], (optional)*) – Connect Loops, Method of bridging multiple loops
- **use_merge** (*boolean, (optional)*) – Merge, Merge rather than creating faces
- **merge_factor** (*float in [0, 1], (optional)*) – Merge Factor
- **twist_offset** (*int in [-1000, 1000], (optional)*) – Twist, Twist offset for closed loops
- **number_cuts** (*int in [0, 1000], (optional)*) – Number of Cuts
- **interpolation** (*enum in ['LINEAR', 'PATH', 'SURFACE'], (optional)*) – Interpolation, Interpolation method
- **smoothness** (*float in [0, 1000], (optional)*) – Smoothness, Smoothness factor
- **profile_shape_factor** (*float in [-1000, 1000], (optional)*) – Profile Factor, How much intermediary new edges are shrunk/expanded
- **profile_shape** (enum in Proportional Falloff Curve Only Items, (optional)) – Profile Shape, Shape of the profile

bpy.ops.mesh.**colors_reverse()**

Flip direction of face corner color attribute inside faces

bpy.ops.mesh.**colors_rotate(*, use_ccw=False)**

Rotate face corner color attribute inside faces

PARAMETERS:

- **use_ccw** (*boolean, (optional)*) – Counter Clockwise

bpy.ops.mesh.**convex_hull(*, delete_unused=True, use_existing_faces=True, make_holes=False, join_triangles=True, face_threshold=0.698132, shape_threshold=0.698132, topology_influence=0.0, uvs=False, vcols=False, seam=False, sharp=False, materials=False, deselect_joined=False)**

Enclose selected vertices in a convex polyhedron

PARAMETERS:

- **delete_unused** (*boolean, (optional)*) – Delete Unused, Delete selected elements that are not used by the hull
- **use_existing_faces** (*boolean, (optional)*) – Use Existing Faces, Skip hull triangles that are covered by a pre-existing face
- **make_holes** (*boolean, (optional)*) – Make Holes, Delete selected faces that are used by the hull
- **join_triangles** (*boolean, (optional)*) – Join Triangles, Merge adjacent triangles into quads
- **face_threshold** (*float in [0, 3.14159], (optional)*) – Max Face Angle, Face angle limit
- **shape_threshold** (*float in [0, 3.14159], (optional)*) – Max Shape Angle, Shape angle limit

- **topology_influence** (*float in [0, 2], (optional)*) – Topology Influence, How much to prioritize regular grids of quads as well as quads that touch existing quads
- **uvs** (*boolean, (optional)*) – Compare UVs
- **vcols** (*boolean, (optional)*) – Compare Color Attributes
- **seam** (*boolean, (optional)*) – Compare Seam
- **sharp** (*boolean, (optional)*) – Compare Sharp
- **materials** (*boolean, (optional)*) – Compare Materials
- **deselect_joined** (*boolean, (optional)*) – Deselect Joined, Only select remaining triangles that were not merged

bpy.ops.mesh.**customdata_custom_splitnormals_add()**

Add a custom split normals layer, if none exists yet

bpy.ops.mesh.**customdata_custom_splitnormals_clear()**

Remove the custom split normals layer, if it exists

bpy.ops.mesh.**customdata_mask_clear()**

Clear vertex sculpt masking data from the mesh

bpy.ops.mesh.**customdata_skin_add()**

Add a vertex skin layer

bpy.ops.mesh.**customdata_skin_clear()**

Clear vertex skin layer

bpy.ops.mesh.**decimate(*, ratio=1.0, use_vertex_group=False, vertex_group_factor=1.0, invert_vertex_group=False, use_symmetry=Fals**
symmetry_axis='Y')

Simplify geometry by collapsing edges

PARAMETERS:

- **ratio** (*float in [0, 1], (optional)*) – Ratio
- **use_vertex_group** (*boolean, (optional)*) – Vertex Group, Use active vertex group as an influence
- **vertex_group_factor** (*float in [0, 1000], (optional)*) – Weight, Vertex group strength
- **invert_vertex_group** (*boolean, (optional)*) – Invert, Invert vertex group influence
- **use_symmetry** (*boolean, (optional)*) – Symmetry, Maintain symmetry on an axis
- **symmetry_axis** (enum in Axis Xyz Items, (optional)) – Axis, Axis of symmetry

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

Delete selected vertices, edges or faces

PARAMETERS:

type (*enum in ['VERT', 'EDGE', 'FACE', 'EDGE_FACE', 'ONLY_FACE'], (optional)*) – Type, Method used for deleting mesh data

bpy.ops.mesh.**delete_edgeloop(*, use_face_split=True)**

Delete an edge loop by merging the faces on each side

PARAMETERS:

use_face_split (*boolean, (optional)*) – Face Split, Split off face corners to maintain surrounding geometry

bpy.ops.mesh.**delete_loose(*, use_verts=True, use_edges=True, use_faces=False)**

Delete loose vertices, edges or faces

PARAMETERS:

- **use_verts** (*boolean, (optional)*) – Vertices, Remove loose vertices
- **use_edges** (*boolean, (optional)*) – Edges, Remove loose edges

- **use_faces** (*boolean, (optional)*) – Faces, Remove loose faces

bpy.ops.mesh.**dissolve_degenerate(*, threshold=0.0001)**

Dissolve zero area faces and zero length edges

PARAMETERS:

threshold (*float in [1e-06, 50], (optional)*) – Merge Distance, Maximum distance between elements to merge

bpy.ops.mesh.**dissolve_edges(*, use_verts=True, use_face_split=False)**

Dissolve edges, merging faces

PARAMETERS:

- **use_verts** (*boolean, (optional)*) – Dissolve Vertices, Dissolve remaining vertices
- **use_face_split** (*boolean, (optional)*) – Face Split, Split off face corners to maintain surrounding geometry

bpy.ops.mesh.**dissolve_faces(*, use_verts=False)**

Dissolve faces

PARAMETERS:

use_verts (*boolean, (optional)*) – Dissolve Vertices, Dissolve remaining vertices

bpy.ops.mesh.**dissolve_limited(*, angle_limit=0.0872665, use_dissolve_boundaries=False, delimit={'NORMAL'})**

Dissolve selected edges and vertices, limited by the angle of surrounding geometry

PARAMETERS:

- **angle_limit** (*float in [0, 3.14159], (optional)*) – Max Angle, Angle limit
- **use_dissolve_boundaries** (*boolean, (optional)*) – All Boundaries, Dissolve all vertices in between face boundaries
- **delimit** (enum set in Mesh Delimit Mode Items, (optional)) – Delimit, Delimit dissolve operation

bpy.ops.mesh.**dissolve_mode(*, use_verts=False, use_face_split=False, use_boundary_tear=False)**

Dissolve geometry based on the selection mode

PARAMETERS:

- **use_verts** (*boolean, (optional)*) – Dissolve Vertices, Dissolve remaining vertices
- **use_face_split** (*boolean, (optional)*) – Face Split, Split off face corners to maintain surrounding geometry
- **use_boundary_tear** (*boolean, (optional)*) – Tear Boundary, Split off face corners instead of merging faces

bpy.ops.mesh.**dissolve_verts(*, use_face_split=False, use_boundary_tear=False)**

Dissolve vertices, merge edges and faces

PARAMETERS:

- **use_face_split** (*boolean, (optional)*) – Face Split, Split off face corners to maintain surrounding geometry
- **use_boundary_tear** (*boolean, (optional)*) – Tear Boundary, Split off face corners instead of merging faces

bpy.ops.mesh.**dupli_extrude_cursor(*, rotate_source=True)**

Duplicate and extrude selected vertices, edges or faces towards the mouse cursor

PARAMETERS:

rotate_source (*boolean, (optional)*) – Rotate Source, Rotate initial selection giving better shape

bpy.ops.mesh.**duplicate(*, mode=1)**

Duplicate selected vertices, edges or faces

PARAMETERS:

mode (*int in [0, inf], (optional)*) – Mode

bpy.ops.mesh.**duplicate_move(*, MESH_OT_duplicate=None, TRANSFORM_OT_translate=None)**

Duplicate mesh and move

PARAMETERS:

- **MESH_OT_duplicate** (`MESH_OT_duplicate`, (optional)) – Duplicate, Duplicate selected vertices, edges or faces
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.mesh.**edge_collapse()**

Collapse isolated edge and face regions, merging data such as UVs and color attributes. This can collapse edge-rings as well as regions of connecte faces into vertices

bpy.ops.mesh.**edge_face_add()**

Add an edge or face to selected

bpy.ops.mesh.**edge_rotate(*, use_ccw=False)**

Rotate selected edge or adjoining faces

PARAMETERS:

use_ccw (*boolean, (optional)*) – Counter Clockwise

bpy.ops.mesh.**edge_split(*, type='EDGE')**

Split selected edges so that each neighbor face gets its own copy

PARAMETERS:

type (*enum in ['EDGE', 'VERT'], (optional)*) –

Type, Method to use for splitting

- `EDGE` Faces by Edges – Split faces along selected edges.
- `VERT` Faces & Edges by Vertices – Split faces and edges connected to selected vertices.

bpy.ops.mesh.**edgering_select(*, extend=False, deselect=False, toggle=False, ring=True)**

Select an edge ring

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend the selection
- **deselect** (*boolean, (optional)*) – Deselect, Remove from the selection
- **toggle** (*boolean, (optional)*) – Toggle Select, Toggle the selection
- **ring** (*boolean, (optional)*) – Select Ring, Select ring

bpy.ops.mesh.**edges_select_sharp(*, sharpness=0.523599)**

Select all sharp enough edges

PARAMETERS:

sharpness (*float in [0.000174533, 3.14159], (optional)*) – Sharpness

bpy.ops.mesh.**extrude_context(*, use_normal_flip=False, use_dissolve_ortho_edges=False, mirror=False)**

Extrude selection

PARAMETERS:

- **use_normal_flip** (*boolean, (optional)*) – Flip Normals
- **use_dissolve_ortho_edges** (*boolean, (optional)*) – Dissolve Orthogonal Edges
- **mirror** (*boolean, (optional)*) – Mirror Editing

bpy.ops.mesh.**extrude_context_move(*, MESH_OT_extrude_context=None, TRANSFORM_OT_translate=None)**

Extrude region together along the average normal

PARAMETERS:

- **MESH_OT_extrude_context** (`MESH_OT_extrude_context` , (optional)) – Extrude Context, Extrude selection
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate` , (optional)) – Move, Move selected items

bpy.ops.mesh.**extrude_edges_indiv(*, use_normal_flip=False, mirror=False)**

Extrude individual edges only

PARAMETERS:

- **use_normal_flip** (*boolean, (optional)*) – Flip Normals
- **mirror** (*boolean, (optional)*) – Mirror Editing

bpy.ops.mesh.**extrude_edges_move(*, MESH_OT_extrude_edges_indiv=None, TRANSFORM_OT_translate=None)**

Extrude edges and move result

PARAMETERS:

- **MESH_OT_extrude_edges_indiv** (`MESH_OT_extrude_edges_indiv` , (optional)) – Extrude Only Edges, Extrude individual edges only
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate` , (optional)) – Move, Move selected items

bpy.ops.mesh.**extrude_faces_indiv(*, mirror=False)**

Extrude individual faces only

PARAMETERS:

mirror (*boolean, (optional)*) – Mirror Editing

bpy.ops.mesh.**extrude_faces_move(*, MESH_OT_extrude_faces_indiv=None, TRANSFORM_OT_shrink_fatten=None)**

Extrude each individual face separately along local normals

PARAMETERS:

- **MESH_OT_extrude_faces_indiv** (`MESH_OT_extrude_faces_indiv` , (optional)) – Extrude Individual Faces, Extrude individua faces only
- **TRANSFORM_OT_shrink_fatten** (`TRANSFORM_OT_shrink_fatten` , (optional)) – Shrink/Fatten, Shrink/fatten selected vertic along normals

bpy.ops.mesh.**extrude_manifold(*, MESH_OT_extrude_region=None, TRANSFORM_OT_translate=None)**

Extrude, dissolves edges whose faces form a flat surface and intersect new edges

PARAMETERS:

- **MESH_OT_extrude_region** (`MESH_OT_extrude_region` , (optional)) – Extrude Region, Extrude region of faces
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate` , (optional)) – Move, Move selected items

bpy.ops.mesh.**extrude_region(*, use_normal_flip=False, use_dissolve_ortho_edges=False, mirror=False)**

Extrude region of faces

PARAMETERS:

- **use_normal_flip** (*boolean, (optional)*) – Flip Normals
- **use_dissolve_ortho_edges** (*boolean, (optional)*) – Dissolve Orthogonal Edges
- **mirror** (*boolean, (optional)*) – Mirror Editing

bpy.ops.mesh.**extrude_region_move(*, MESH_OT_extrude_region=None, TRANSFORM_OT_translate=None)**

Extrude region and move result

PARAMETERS:

- **MESH_OT_extrude_region** (`MESH_OT_extrude_region` , (optional)) – Extrude Region, Extrude region of faces
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate` , (optional)) – Move, Move selected items

bpy.ops.mesh.**extrude_region_shrink_fatten(*, MESH_OT_extrude_region=None, TRANSFORM_OT_shrink_fatten=None)**

Extrude region together along local normals

PARAMETERS:

- **MESH_OT_extrude_region** (`MESH_OT_extrude_region` , (optional)) – Extrude Region, Extrude region of faces
- **TRANSFORM_OT_shrink_fatten** (`TRANSFORM_OT_shrink_fatten` , (optional)) – Shrink/Fatten, Shrink/fatten selected vertic along normals

bpy.ops.mesh.**extrude_repeat(*, steps=10, offset=(0.0, 0.0, 0.0), scale_offset=1.0)**

Extrude selected vertices, edges or faces repeatedly

PARAMETERS:

- **steps** (*int in [0, 1000000], (optional)*) – Steps
- **offset** (`mathutils.Vector` of 3 items in [-100000, 100000], (optional)) – Offset, Offset vector
- **scale_offset** (*float in [0, inf], (optional)*) – Scale Offset

bpy.ops.mesh.**extrude_vertices_move(*, MESH_OT_extrude_verts_indiv=None, TRANSFORM_OT_translate=None)**

Extrude vertices and move result

PARAMETERS:

- **MESH_OT_extrude_verts_indiv** (`MESH_OT_extrude_verts_indiv` , (optional)) – Extrude Only Vertices, Extrude individual vertices only
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate` , (optional)) – Move, Move selected items

bpy.ops.mesh.**extrude_verts_indiv(*, mirror=False)**

Extrude individual vertices only

PARAMETERS:

mirror (*boolean, (optional)*) – Mirror Editing

bpy.ops.mesh.**face_make_planar(*, factor=1.0, repeat=1)**

Flatten selected faces

PARAMETERS:

- **factor** (*float in [-10, 10], (optional)*) – Factor
- **repeat** (*int in [1, 10000], (optional)*) – Iterations

bpy.ops.mesh.**face_set_extract(*, add_boundary_loop=True, smooth_iterations=4, apply_shrinkwrap=True, add_solidify=True)**

Create a new mesh object from the selected Face Set

PARAMETERS:

- **add_boundary_loop** (*boolean, (optional)*) – Add Boundary Loop, Add an extra edge loop to better preserve the shape when applying a subdivision surface modifier
- **smooth_iterations** (*int in [0, inf], (optional)*) – Smooth Iterations, Smooth iterations applied to the extracted mesh
- **apply_shrinkwrap** (*boolean, (optional)*) – Project to Sculpt, Project the extracted mesh into the original sculpt
- **add_solidify** (*boolean, (optional)*) – Extract as Solid, Extract the mask as a solid object with a solidify modifier

bpy.ops.mesh.**face_split_by_edges()**

Weld loose edges into faces (splitting them into new faces)

bpy.ops.mesh.**faces_mirror_uv(*, direction='POSITIVE', precision=3)**

Copy mirror UV coordinates on the X axis based on a mirrored mesh

PARAMETERS:

- **direction** (*enum in ['POSITIVE', 'NEGATIVE'], (optional)*) – Axis Direction
- **precision** (*int in [1, 16], (optional)*) – Precision, Tolerance for finding vertex duplicates

FILE:

startup/bl_operators/mesh.py:134

bpy.ops.mesh.**faces_select_linked_flat(*, sharpness=0.0174533)**

Select linked faces by angle

PARAMETERS:

sharpness (*float in [0.000174533, 3.14159], (optional)*) – Sharpness

bpy.ops.mesh.**faces_shade_flat()**

Display faces flat

bpy.ops.mesh.**faces_shade_smooth()**

Display faces smooth (using vertex normals)

bpy.ops.mesh.**fill(*, use_beauty=True)**

Fill a selected edge loop with faces

PARAMETERS:

use_beauty (*boolean, (optional)*) – Beauty, Use best triangulation division

bpy.ops.mesh.**fill_grid(*, span=1, offset=0, use_interp_simple=False)**

Fill grid from two loops

PARAMETERS:

- **span** (*int in [1, 1000], (optional)*) – Span, Number of grid columns
- **offset** (*int in [-1000, 1000], (optional)*) – Offset, Vertex that is the corner of the grid
- **use_interp_simple** (*boolean, (optional)*) – Simple Blending, Use simple interpolation of grid vertices

bpy.ops.mesh.**fill_holes(*, sides=4)**

Fill in holes (boundary edge loops)

PARAMETERS:

sides (*int in [0, 1000], (optional)*) – Sides, Number of sides in hole required to fill (zero fills all holes)

bpy.ops.mesh.**flip_normals(*, only_clnors=False)**

Flip the direction of selected faces' normals (and of their vertices)

PARAMETERS:

only_clnors (*boolean, (optional)*) – Custom Normals Only, Only flip the custom loop normals of the selected elements

bpy.ops.mesh.**flip_quad_tessellation()**

Flips the tessellation of selected quads

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

Hide (un)selected vertices, edges or faces

PARAMETERS:

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

bpy.ops.mesh.**inset(*, use_boundary=True, use_even_offset=True, use_relative_offset=False, use_edge_rail=False, thickness=0.0, depth=0.0, use_outset=False, use_select_inset=False, use_individual=False, use_interpolate=True, release_confirm=False)**

Inset new faces into selected faces

PARAMETERS:

- **use_boundary** (*boolean, (optional)*) – Boundary, Inset face boundaries

- **use_even_offset** (*boolean, (optional)*) – Offset Even, Scale the offset to give more even thickness
- **use_relative_offset** (*boolean, (optional)*) – Offset Relative, Scale the offset by surrounding geometry
- **use_edge_rail** (*boolean, (optional)*) – Edge Rail, Inset the region along existing edges
- **thickness** (*float in [0, inf], (optional)*) – Thickness
- **depth** (*float in [-inf, inf], (optional)*) – Depth
- **use_outset** (*boolean, (optional)*) – Outset, Outset rather than inset
- **use_select_inset** (*boolean, (optional)*) – Select Outer, Select the new inset faces
- **use_individual** (*boolean, (optional)*) – Individual, Individual face inset
- **use_interpolate** (*boolean, (optional)*) – Interpolate, Blend face data across the inset
- **release_confirm** (*boolean, (optional)*) – Confirm on Release

bpy.ops.mesh.**intersect(*, mode='SELECT_UNSELECT', separate_mode='CUT', threshold=1e-06, solver='EXACT')**

Cut an intersection into faces

PARAMETERS:

- **mode** (*enum in ['SELECT', 'SELECT_UNSELECT'], (optional)*) –
  Source
  - `SELECT` Self Intersect – Self intersect selected faces.
  - `SELECT_UNSELECT` Selected/Unselected – Intersect selected with unselected faces.
- **separate_mode** (*enum in ['ALL', 'CUT', 'NONE'], (optional)*) –
  Separate Mode
  - `ALL` All – Separate all geometry from intersections.
  - `CUT` Cut – Cut into geometry keeping each side separate (Selected/Unselected only).
  - `NONE` Merge – Merge all geometry from the intersection.
- **threshold** (*float in [0, 0.01], (optional)*) – Merge Threshold
- **solver** (*enum in ['FAST', 'EXACT'], (optional)*) –
  Solver, Which Intersect solver to use
  - `FAST` Fast – Faster solver, some limitations.
  - `EXACT` Exact – Exact solver, slower, handles more cases.

bpy.ops.mesh.**intersect_boolean(*, operation='DIFFERENCE', use_swap=False, use_self=False, threshold=1e-06, solver='EXACT')**

Cut solid geometry from selected to unselected

PARAMETERS:

- **operation** (*enum in ['INTERSECT', 'UNION', 'DIFFERENCE'], (optional)*) – Boolean Operation, Which boolean operation to apply
- **use_swap** (*boolean, (optional)*) – Swap, Use with difference intersection to swap which side is kept
- **use_self** (*boolean, (optional)*) – Self Intersection, Do self-union or self-intersection
- **threshold** (*float in [0, 0.01], (optional)*) – Merge Threshold
- **solver** (*enum in ['FAST', 'EXACT'], (optional)*) –
  Solver, Which Boolean solver to use
  - `FAST` Fast – Faster solver, some limitations.
  - `EXACT` Exact – Exact solver, slower, handles more cases.

bpy.ops.mesh.**knife_project(*, cut_through=False)**

Use other objects outlines and boundaries to project knife cuts

PARAMETERS:

cut_through (*boolean, (optional)*) – Cut Through, Cut through all faces, not just visible ones

bpy.ops.mesh.**knife_tool(*, use_occlude_geometry=True, only_selected=False, xray=True, visible_measurements='NONE',**

angle_snapping='NONE', angle_snapping_increment=0.523599, wait_for_input=True)

Cut new topology

PARAMETERS:

- **use_occlude_geometry** (*boolean, (optional)*) – Occlude Geometry, Only cut the front most geometry
- **only_selected** (*boolean, (optional)*) – Only Selected, Only cut selected geometry
- **xray** (*boolean, (optional)*) – X-Ray, Show cuts hidden by geometry
- **visible_measurements** (*enum in ['NONE', 'BOTH', 'DISTANCE', 'ANGLE'], (optional)*) – Measurements, Visible distance and angle measurements
  - `NONE` None – Show no measurements.
  - `BOTH` Both – Show both distances and angles.
  - `DISTANCE` Distance – Show just distance measurements.
  - `ANGLE` Angle – Show just angle measurements.
- **angle_snapping** (*enum in ['NONE', 'SCREEN', 'RELATIVE'], (optional)*) – Angle Snapping, Angle snapping mode
  - `NONE` None – No angle snapping.
  - `SCREEN` Screen – Screen space angle snapping.
  - `RELATIVE` Relative – Angle snapping relative to the previous cut edge.
- **angle_snapping_increment** (*float in [0, 3.14159], (optional)*) – Angle Snap Increment, The angle snap increment used when in constrain angle mode
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.mesh.**loop_multi_select(*, ring=False)**

Select a loop of connected edges by connection type

PARAMETERS:

ring (*boolean, (optional)*) – Ring

bpy.ops.mesh.**loop_select(*, extend=False, deselect=False, toggle=False, ring=False)**

Select a loop of connected edges

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend Select, Extend the selection
- **deselect** (*boolean, (optional)*) – Deselect, Remove from the selection
- **toggle** (*boolean, (optional)*) – Toggle Select, Toggle the selection
- **ring** (*boolean, (optional)*) – Select Ring, Select ring

bpy.ops.mesh.**loop_to_region(*, select_bigger=False)**

Select region of faces inside of a selected loop of edges

PARAMETERS:

select_bigger (*boolean, (optional)*) – Select Bigger, Select bigger regions instead of smaller ones

bpy.ops.mesh.**loopcut(*, number_cuts=1, smoothness=0.0, falloff='INVERSE_SQUARE', object_index=-1, edge_index=-1, mesh_select_mode_init=(False, False, False))**

Add a new loop between existing loops

PARAMETERS:

- **number_cuts** (*int in [1, 1000000], (optional)*) – Number of Cuts
- **smoothness** (*float in [-1000, 1000], (optional)*) – Smoothness, Smoothness factor
- **falloff** (enum in Proportional Falloff Curve Only Items, (optional)) – Falloff, Falloff type of the feather
- **object_index** (*int in [-1, inf], (optional)*) – Object Index

- **edge_index** (*int in [-1, inf], (optional)*) – Edge Index

bpy.ops.mesh.**loopcut_slide(*, MESH_OT_loopcut=None, TRANSFORM_OT_edge_slide=None)**

Cut mesh loop and slide it

PARAMETERS:

- **MESH_OT_loopcut** (`MESH_OT_loopcut`, (optional)) – Loop Cut, Add a new loop between existing loops
- **TRANSFORM_OT_edge_slide** (`TRANSFORM_OT_edge_slide`, (optional)) – Edge Slide, Slide an edge loop along a mesh

bpy.ops.mesh.**mark_freestyle_edge(*, clear=False)**

(Un)mark selected edges as Freestyle feature edges

PARAMETERS:

clear (*boolean, (optional)*) – Clear

bpy.ops.mesh.**mark_freestyle_face(*, clear=False)**

(Un)mark selected faces for exclusion from Freestyle feature edge detection

PARAMETERS:

clear (*boolean, (optional)*) – Clear

bpy.ops.mesh.**mark_seam(*, clear=False)**

(Un)mark selected edges as a seam

PARAMETERS:

clear (*boolean, (optional)*) – Clear

bpy.ops.mesh.**mark_sharp(*, clear=False, use_verts=False)**

(Un)mark selected edges as sharp

PARAMETERS:

- **clear** (*boolean, (optional)*) – Clear
- **use_verts** (*boolean, (optional)*) – Vertices, Consider vertices instead of edges to select which edges to (un)tag as sharp

bpy.ops.mesh.**merge(*, type='CENTER', uvs=False)**

Merge selected vertices

PARAMETERS:

- **type** (*enum in ['CENTER', 'CURSOR', 'COLLAPSE', 'FIRST', 'LAST'], (optional)*) – Type, Merge method to use
- **uvs** (*boolean, (optional)*) – UVs, Move UVs according to merge

bpy.ops.mesh.**merge_normals()**

Merge custom normals of selected vertices

bpy.ops.mesh.**mod_weighted_strength(*, set=False, face_strength='MEDIUM')**

Set/Get strength of face (used in Weighted Normal modifier)

PARAMETERS:

- **set** (*boolean, (optional)*) – Set Value, Set value of faces
- **face_strength** (*enum in ['WEAK', 'MEDIUM', 'STRONG'], (optional)*) – Face Strength, Strength to use for assigning or selecting face influence for weighted normal modifier

bpy.ops.mesh.**normals_make_consistent(*, inside=False)**

Make face and vertex normals point either outside or inside the mesh

PARAMETERS:

inside (*boolean, (optional)*) – Inside

bpy.ops.mesh.**normals_tools(*, mode='COPY', absolute=False)**

Custom normals tools using Normal Vector of UI

PARAMETERS:

- **mode** (*enum in ['COPY', 'PASTE', 'ADD', 'MULTIPLY', 'RESET'], (optional)*) –
  Mode, Mode of tools taking input from interface
  - `COPY` Copy Normal – Copy normal to the internal clipboard.
  - `PASTE` Paste Normal – Paste normal from the internal clipboard.
  - `ADD` Add Normal – Add normal vector with selection.
  - `MULTIPLY` Multiply Normal – Multiply normal vector with selection.
  - `RESET` Reset Normal – Reset the internal clipboard and/or normal of selected element.
- **absolute** (*boolean, (optional)*) – Absolute Coordinates, Copy Absolute coordinates of Normal vector

bpy.ops.mesh.**offset_edge_loops(*, use_cap_endpoint=False)**

Create offset edge loop from the current selection

PARAMETERS:

use_cap_endpoint (*boolean, (optional)*) – Cap Endpoint, Extend loop around end-points

bpy.ops.mesh.**offset_edge_loops_slide(*, MESH_OT_offset_edge_loops=None, TRANSFORM_OT_edge_slide=None)**

Offset edge loop slide

PARAMETERS:

- **MESH_OT_offset_edge_loops** (`MESH_OT_offset_edge_loops`, (optional)) – Offset Edge Loop, Create offset edge loop from the current selection
- **TRANSFORM_OT_edge_slide** (`TRANSFORM_OT_edge_slide`, (optional)) – Edge Slide, Slide an edge loop along a mesh

bpy.ops.mesh.**paint_mask_extract(*, mask_threshold=0.5, add_boundary_loop=True, smooth_iterations=4, apply_shrinkwrap=True, add_solidify=True)**

Create a new mesh object from the current paint mask

PARAMETERS:

- **mask_threshold** (*float in [0, 1], (optional)*) – Threshold, Minimum mask value to consider the vertex valid to extract a face from the origin mesh
- **add_boundary_loop** (*boolean, (optional)*) – Add Boundary Loop, Add an extra edge loop to better preserve the shape when applying a subdivision surface modifier
- **smooth_iterations** (*int in [0, inf], (optional)*) – Smooth Iterations, Smooth iterations applied to the extracted mesh
- **apply_shrinkwrap** (*boolean, (optional)*) – Project to Sculpt, Project the extracted mesh into the original sculpt
- **add_solidify** (*boolean, (optional)*) – Extract as Solid, Extract the mask as a solid object with a solidify modifier

bpy.ops.mesh.**paint_mask_slice(*, mask_threshold=0.5, fill_holes=True, new_object=True)**

Slices the paint mask from the mesh

PARAMETERS:

- **mask_threshold** (*float in [0, 1], (optional)*) – Threshold, Minimum mask value to consider the vertex valid to extract a face from the origin mesh
- **fill_holes** (*boolean, (optional)*) – Fill Holes, Fill holes after slicing the mask
- **new_object** (*boolean, (optional)*) – Slice to New Object, Create a new object from the sliced mask

bpy.ops.mesh.**point_normals(*, mode='COORDINATES', invert=False, align=False, target_location=(0.0, 0.0, 0.0), spherize=False, spherize_strength=0.1)**

Point selected custom normals to specified Target

PARAMETERS:

- **mode** (*enum in ['COORDINATES', 'MOUSE'], (optional)*) –
  Mode, How to define coordinates to point custom normals to
  - `COORDINATES` Coordinates – Use static coordinates (defined by various means).
  - `MOUSE` Mouse – Follow mouse cursor.
- **invert** (*boolean, (optional)*) – Invert, Invert affected normals
- **align** (*boolean, (optional)*) – Align, Make all affected normals parallel
- **target_location** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Target, Target location to which normals will point
- **spherize** (*boolean, (optional)*) – Spherize, Interpolate between original and new normals
- **spherize_strength** (*float in [0, 1], (optional)*) – Spherize Strength, Ratio of spherized normal to original normal

bpy.ops.mesh.**poke(*, offset=0.0, use_relative_offset=False, center_mode='MEDIAN_WEIGHTED')**

Split a face into a fan

PARAMETERS:

- **offset** (*float in [-1000, 1000], (optional)*) – Poke Offset, Poke Offset
- **use_relative_offset** (*boolean, (optional)*) – Offset Relative, Scale the offset by surrounding geometry
- **center_mode** (*enum in ['MEDIAN_WEIGHTED', 'MEDIAN', 'BOUNDS'], (optional)*) –
  Poke Center, Poke face center calculation
  - `MEDIAN_WEIGHTED` Weighted Median – Weighted median face center.
  - `MEDIAN` Median – Median face center.
  - `BOUNDS` Bounds – Face bounds center.

bpy.ops.mesh.**polybuild_delete_at_cursor(*, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, release_confirm=False, use_accurate=False)**

Undocumented, consider contributing.

PARAMETERS:

- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.mesh.**polybuild_dissolve_at_cursor()**

Undocumented, consider contributing.

bpy.ops.mesh.**polybuild_extrude_at_cursor_move(*, MESH_OT_polybuild_transform_at_cursor=None, MESH_OT_extrude_edges_indiv=None, TRANSFORM_OT_translate=None)**

Undocumented, consider contributing.

PARAMETERS:

- **MESH_OT_polybuild_transform_at_cursor** (`MESH_OT_polybuild_transform_at_cursor`, (optional)) – Poly Build Transform at Cursor
- **MESH_OT_extrude_edges_indiv** (`MESH_OT_extrude_edges_indiv`, (optional)) – Extrude Only Edges, Extrude individual edges only
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.mesh.**polybuild_face_at_cursor(*, create_quads=True, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=Fals release_confirm=False, use_accurate=False)**

Undocumented, consider contributing.

PARAMETERS:

- **create_quads** (*boolean, (optional)*) – Create Quads, Automatically split edges in triangles to maintain quad topology
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.mesh.**polybuild_face_at_cursor_move(*, MESH_OT_polybuild_face_at_cursor=None, TRANSFORM_OT_translate=None)**

Undocumented, consider contributing.

PARAMETERS:

- **MESH_OT_polybuild_face_at_cursor** (`MESH_OT_polybuild_face_at_cursor`, (optional)) – Poly Build Face at Cursor
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.mesh.**polybuild_split_at_cursor(*, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, release_confirm=False, use_accurate=False)**

Undocumented, consider contributing.

PARAMETERS:

- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.mesh.**polybuild_split_at_cursor_move(*, MESH_OT_polybuild_split_at_cursor=None, TRANSFORM_OT_translate=None)**

Undocumented, consider contributing.

PARAMETERS:

- **MESH_OT_polybuild_split_at_cursor** (`MESH_OT_polybuild_split_at_cursor`, (optional)) – Poly Build Split at Cursor
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.mesh.**polybuild_transform_at_cursor(*, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, release_confirm=False, use_accurate=False)**

Undocumented, consider contributing.

PARAMETERS:

- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode

- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.mesh.**polybuild_transform_at_cursor_move(*, MESH_OT_polybuild_transform_at_cursor=None, TRANSFORM_OT_translate=None)**

Undocumented, consider contributing.

PARAMETERS:

- **MESH_OT_polybuild_transform_at_cursor** (`MESH_OT_polybuild_transform_at_cursor`, (optional)) – Poly Build Transform at Cursor
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.mesh.**primitive_circle_add(*, vertices=32, radius=1.0, fill_type='NOTHING', calc_uvs=True, 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 circle mesh

PARAMETERS:

- **vertices** (*int in [3, 10000000], (optional)*) – Vertices
- **radius** (*float in [0, inf], (optional)*) – Radius
- **fill_type** (*enum in ['NOTHING', 'NGON', 'TRIFAN'], (optional)*) –
  Fill Type
  - `NOTHING` Nothing – Don't fill at all.
  - `NGON` N-Gon – Use n-gons.
  - `TRIFAN` Triangle Fan – Use triangle fans.
- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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.mesh.**primitive_cone_add(*, vertices=32, radius1=1.0, radius2=0.0, depth=2.0, end_fill_type='NGON', calc_uvs=True, 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 conic mesh

PARAMETERS:

- **vertices** (*int in [3, 10000000], (optional)*) – Vertices
- **radius1** (*float in [0, inf], (optional)*) – Radius 1
- **radius2** (*float in [0, inf], (optional)*) – Radius 2
- **depth** (*float in [0, inf], (optional)*) – Depth
- **end_fill_type** (*enum in ['NOTHING', 'NGON', 'TRIFAN'], (optional)*) –
  Base Fill Type
  - `NOTHING` Nothing – Don't fill at all.

- NGON N-Gon – Use n-gons.
- TRIFAN Triangle Fan – Use triangle fans.

- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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.mesh.**primitive_cube_add(*, size=2.0, calc_uvs=True, 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 cube mesh that consists of six square faces

PARAMETERS:

- **size** (*float in [0, inf], (optional)*) – Size
- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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.mesh.**primitive_cube_add_gizmo(*, calc_uvs=True, 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), matrix=((0.0, 0.0, 0.0, 0.0), (0.0, 0.0, 0.0, 0.0), (0.0, 0.0, 0.0, 0.0), (0.0, 0.0, 0.0, 0.0)))**

Construct a cube mesh

PARAMETERS:

- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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
- **matrix** (mathutils.Matrix of 4 * 4 items in [-inf, inf], (optional)) – Matrix

bpy.ops.mesh.**primitive_cylinder_add(*, vertices=32, radius=1.0, depth=2.0, end_fill_type='NGON', calc_uvs=True, 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 cylinder mesh

PARAMETERS:

- **vertices** (*int in [3, 10000000], (optional)*) – Vertices
- **radius** (*float in [0, inf], (optional)*) – Radius
- **depth** (*float in [0, inf], (optional)*) – Depth
- **end_fill_type** (*enum in ['NOTHING', 'NGON', 'TRIFAN'], (optional)*) –
  Cap Fill Type
  - `NOTHING` Nothing – Don't fill at all.
  - `NGON` N-Gon – Use n-gons.
  - `TRIFAN` Triangle Fan – Use triangle fans.
- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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.mesh.**primitive_grid_add(*, x_subdivisions=10, y_subdivisions=10, size=2.0, calc_uvs=True, 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 subdivided plane mesh

PARAMETERS:

- **x_subdivisions** (*int in [1, 10000000], (optional)*) – X Subdivisions
- **y_subdivisions** (*int in [1, 10000000], (optional)*) – Y Subdivisions
- **size** (*float in [0, inf], (optional)*) – Size
- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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.mesh.**primitive_ico_sphere_add(*, subdivisions=2, radius=1.0, calc_uvs=True, 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 spherical mesh that consists of equally sized triangles

PARAMETERS:

- **subdivisions** (*int in [1, 10], (optional)*) – Subdivisions
- **radius** (*float in [0, inf], (optional)*) – Radius
- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map

- **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.mesh.**primitive_monkey_add(*, size=2.0, calc_uvs=True, 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 Suzanne mesh

PARAMETERS:

- **size** (*float in [0, inf], (optional)*) – Size
- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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.mesh.**primitive_plane_add(*, size=2.0, calc_uvs=True, 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 filled planar mesh with 4 vertices

PARAMETERS:

- **size** (*float in [0, inf], (optional)*) – Size
- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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.mesh.**primitive_torus_add(*, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), major_segments=48, minor_segments=12, mode='MAJOR_MINOR', major_radius=1.0, minor_radius=0.25, abso_major_rad=1.25, abso_minor_rad=0.75, generate_uvs=True)**

Construct a torus mesh

PARAMETERS:

- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –
  Align
  - `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
- **rotation** (`mathutils.Euler` rotation of 3 items in [-inf, inf], (optional)) – Rotation
- **major_segments** (*int in [3, 256], (optional)*) – Major Segments, Number of segments for the main ring of the torus
- **minor_segments** (*int in [3, 256], (optional)*) – Minor Segments, Number of segments for the minor ring of the torus
- **mode** (*enum in ['MAJOR_MINOR', 'EXT_INT'], (optional)*) –
  Dimensions Mode
  - `MAJOR_MINOR` Major/Minor – Use the major/minor radii for torus dimensions.
  - `EXT_INT` Exterior/Interior – Use the exterior/interior radii for torus dimensions.
- **major_radius** (*float in [0, 10000], (optional)*) – Major Radius, Radius from the origin to the center of the cross sections
- **minor_radius** (*float in [0, 10000], (optional)*) – Minor Radius, Radius of the torus' cross section
- **abso_major_rad** (*float in [0, 10000], (optional)*) – Exterior Radius, Total Exterior Radius of the torus
- **abso_minor_rad** (*float in [0, 10000], (optional)*) – Interior Radius, Total Interior Radius of the torus
- **generate_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map

FILE:

startup/bl_operators/add_mesh_torus.py:222

bpy.ops.mesh.**primitive_uv_sphere_add(*, segments=32, ring_count=16, radius=1.0, calc_uvs=True, 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 spherical mesh with quad faces, except for triangle faces at the top and bottom

PARAMETERS:

- **segments** (*int in [3, 100000], (optional)*) – Segments
- **ring_count** (*int in [3, 100000], (optional)*) – Rings
- **radius** (*float in [0, inf], (optional)*) – Radius
- **calc_uvs** (*boolean, (optional)*) – Generate UVs, Generate a default UV map
- **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.mesh.**quads_convert_to_tris(*, quad_method='BEAUTY', ngon_method='BEAUTY')**

Triangulate selected faces

PARAMETERS:

- **quad_method** (enum in Modifier Triangulate Quad Method Items, (optional)) – Quad Method, Method for splitting the quads into triangles
- **ngon_method** (enum in Modifier Triangulate Ngon Method Items, (optional)) – N-gon Method, Method for splitting the n-gons into triangles

bpy.ops.mesh.**region_to_loop()**

Select boundary edges around the selected faces

bpy.ops.mesh.**remove_doubles(*, threshold=0.0001, use_unselected=False, use_sharp_edge_from_normals=False)**

Merge vertices based on their proximity

PARAMETERS:

- **threshold** (*float in [1e-06, 50], (optional)*) – Merge Distance, Maximum distance between elements to merge
- **use_unselected** (*boolean, (optional)*) – Unselected, Merge selected to other unselected vertices
- **use_sharp_edge_from_normals** (*boolean, (optional)*) – Sharp Edges, Calculate sharp edges using custom normal data (when available)

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

Reveal all hidden vertices, edges and faces

PARAMETERS:

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

bpy.ops.mesh.**rip(*, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, release_confirm=False, use_accurate=False, use_fill=False)**

Disconnect vertex or edges from connected geometry

PARAMETERS:

- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation
- **use_fill** (*boolean, (optional)*) – Fill, Fill the ripped region

bpy.ops.mesh.**rip_edge(*, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, release_confirm=False, use_accurate=False)**

Extend vertices along the edge closest to the cursor

PARAMETERS:

- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.mesh.**rip_edge_move(*, MESH_OT_rip_edge=None, TRANSFORM_OT_translate=None)**

Extend vertices and move the result

PARAMETERS:

- **MESH_OT_rip_edge** (`MESH_OT_rip_edge`, (optional)) – Extend Vertices, Extend vertices along the edge closest to the cursor
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.mesh.**rip_move(*, MESH_OT_rip=None, TRANSFORM_OT_translate=None)**

Rip polygons and move the result

PARAMETERS:

- **MESH_OT_rip** (`MESH_OT_rip` , (optional)) – Rip, Disconnect vertex or edges from connected geometry
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate` , (optional)) – Move, Move selected items

bpy.ops.mesh.**screw(*, steps=9, turns=1, center=(0.0, 0.0, 0.0), axis=(0.0, 0.0, 0.0))**

Extrude selected vertices in screw-shaped rotation around the cursor in indicated viewport

PARAMETERS:

- **steps** (*int in [1, 100000], (optional)*) – Steps, Steps
- **turns** (*int in [1, 100000], (optional)*) – Turns, Turns
- **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.mesh.**select_all(*, action='TOGGLE')**

(De)select all vertices, edges or faces

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.mesh.**select_axis(*, orientation='LOCAL', sign='POS', axis='X', threshold=0.0001)**

Select all data in the mesh on a single axis

PARAMETERS:

- **orientation** (enum in Transform Orientation Items, (optional)) – Axis Mode, Axis orientation
- **sign** (*enum in ['POS', 'NEG', 'ALIGN'], (optional)*) – Axis Sign, Side to select
- **axis** (enum in Axis Xyz Items, (optional)) – Axis, Select the axis to compare each vertex on
- **threshold** (*float in [1e-06, 50], (optional)*) – Threshold

bpy.ops.mesh.**select_by_attribute()**

Select elements based on the active boolean attribute

bpy.ops.mesh.**select_by_pole_count(*, pole_count=4, type='NOTEQUAL', extend=False, exclude_nonmanifold=True)**

Select vertices at poles by the number of connected edges. In edge and face mode the geometry connected to the vertices is selected

PARAMETERS:

- **pole_count** (*int in [0, inf], (optional)*) – Pole Count
- **type** (*enum in ['LESS', 'EQUAL', 'GREATER', 'NOTEQUAL'], (optional)*) – Type, Type of comparison to make
- **extend** (*boolean, (optional)*) – Extend, Extend the selection
- **exclude_nonmanifold** (*boolean, (optional)*) – Exclude Non Manifold, Exclude non-manifold poles

bpy.ops.mesh.**select_face_by_sides(*, number=4, type='EQUAL', extend=True)**

Select vertices or faces by the number of face sides

PARAMETERS:

- **number** (*int in [3, inf], (optional)*) – Number of Vertices
- **type** (*enum in ['LESS', 'EQUAL', 'GREATER', 'NOTEQUAL'], (optional)*) – Type, Type of comparison to make
- **extend** (*boolean, (optional)*) – Extend, Extend the selection

bpy.ops.mesh.**select_interior_faces()**

Select faces where all edges have more than 2 face users

bpy.ops.mesh.**select_less(*, use_face_step=True)**

Deselect vertices, edges or faces at the boundary of each selection region

PARAMETERS:

use_face_step (*boolean, (optional)*) – Face Step, Connected faces (instead of edges)

bpy.ops.mesh.**select_linked(*, delimit={'SEAM'})**

Select all vertices connected to the current selection

PARAMETERS:

delimit (enum set in Mesh Delimit Mode Items, (optional)) – Delimit, Delimit selected region

bpy.ops.mesh.**select_linked_pick(*, deselect=False, delimit={'SEAM'}, object_index=-1, index=-1)**

(De)select all vertices linked to the edge under the mouse cursor

PARAMETERS:

- **deselect** (*boolean, (optional)*) – Deselect
- **delimit** (enum set in Mesh Delimit Mode Items, (optional)) – Delimit, Delimit selected region

bpy.ops.mesh.**select_loose(*, extend=False)**

Select loose geometry based on the selection mode

PARAMETERS:

extend (*boolean, (optional)*) – Extend, Extend the selection

bpy.ops.mesh.**select_mirror(*, axis={'X'}, extend=False)**

Select mesh items at mirrored locations

PARAMETERS:

- **axis** (enum set in Axis Flag Xyz Items, (optional)) – Axis
- **extend** (*boolean, (optional)*) – Extend, Extend the existing selection

bpy.ops.mesh.**select_mode(*, use_extend=False, use_expand=False, type='VERT', action='TOGGLE')**

Change selection mode

PARAMETERS:

- **use_extend** (*boolean, (optional)*) – Extend
- **use_expand** (*boolean, (optional)*) – Expand
- **type** (enum in Mesh Select Mode Items, (optional)) – Type
- **action** (*enum in ['DISABLE', 'ENABLE', 'TOGGLE'], (optional)*) –

  Action, Selection action to execute

  - `DISABLE` Disable – Disable selected markers.
  - `ENABLE` Enable – Enable selected markers.
  - `TOGGLE` Toggle – Toggle disabled flag for selected markers.

bpy.ops.mesh.**select_more(*, use_face_step=True)**

Select more vertices, edges or faces connected to initial selection

PARAMETERS:

use_face_step (*boolean, (optional)*) – Face Step, Connected faces (instead of edges)

bpy.ops.mesh.**select_next_item()**

Select the next element (using selection order)

FILE:

startup/bl_operators/mesh.py:206

bpy.ops.mesh.**select_non_manifold(*, extend=True, use_wire=True, use_boundary=True, use_multi_face=True, use_non_contiguous=True, use_verts=True)**

Select all non-manifold vertices or edges

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend the selection
- **use_wire** (*boolean, (optional)*) – Wire, Wire edges
- **use_boundary** (*boolean, (optional)*) – Boundaries, Boundary edges
- **use_multi_face** (*boolean, (optional)*) – Multiple Faces, Edges shared by more than two faces
- **use_non_contiguous** (*boolean, (optional)*) – Non Contiguous, Edges between faces pointing in alternate directions
- **use_verts** (*boolean, (optional)*) – Vertices, Vertices connecting multiple face regions

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

Deselect every Nth element starting from the active vertex, edge or face

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.mesh.**select_prev_item()**

Select the previous element (using selection order)

FILE:

startup/bl_operators/mesh.py:231

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

Randomly select vertices

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.mesh.**select_similar(*, type='VERT_NORMAL', compare='EQUAL', threshold=0.0)**

Select similar vertices, edges or faces by property types

PARAMETERS:

- **type** (*enum in ['VERT_NORMAL', 'VERT_FACES', 'VERT_GROUPS', 'VERT_EDGES', 'VERT_CREASE', 'EDGE_LENGTH', 'EDGE_DIR', 'EDGE_FACES', 'EDGE_FACE_ANGLE', 'EDGE_CREASE', 'EDGE_BEVEL', 'EDGE_SEAM', 'EDGE_SHARP', 'EDGE_FREESTYLE', 'FACE_MATERIAL', 'FACE_AREA', 'FACE_SIDES', 'FACE_PERIMETER', 'FACE_NORMAL', 'FACE_COPLANAR', 'FACE_SMOOTH', 'FACE_FREESTYLE'], (optional)*) – Type
- **compare** (*enum in ['EQUAL', 'GREATER', 'LESS'], (optional)*) – Compare
- **threshold** (*float in [0, 100000], (optional)*) – Threshold

bpy.ops.mesh.**select_similar_region()**

Select similar face regions to the current selection

Select similar face regions to the current selection

bpy.ops.mesh.**select_ungrouped(*, extend=False)**

Select vertices without a group

PARAMETERS:

extend (*boolean, (optional)*) – Extend, Extend the selection

bpy.ops.mesh.**separate(*, type='SELECTED')**

Separate selected geometry into a new mesh

PARAMETERS:

type (*enum in ['SELECTED', 'MATERIAL', 'LOOSE'], (optional)*) – Type

bpy.ops.mesh.**set_normals_from_faces(*, keep_sharp=False)**

Set the custom normals from the selected faces ones

PARAMETERS:

keep_sharp (*boolean, (optional)*) – Keep Sharp Edges, Do not set sharp edges to face

bpy.ops.mesh.**set_sharpness_by_angle(*, angle=0.523599, extend=False)**

Set edge sharpness based on the angle between neighboring faces

PARAMETERS:

- **angle** (*float in [0.000174533, 3.14159], (optional)*) – Angle
- **extend** (*boolean, (optional)*) – Extend, Add new sharp edges without clearing existing sharp edges

bpy.ops.mesh.**shape_propagate_to_all()**

Apply selected vertex locations to all other shape keys

bpy.ops.mesh.**shortest_path_pick(*, edge_mode='SELECT', use_face_step=False, use_topology_distance=False, use_fill=False, skip=0, nth=1, offset=0, index=-1)**

Select shortest path between two selections

PARAMETERS:

- **edge_mode** (*enum in ['SELECT', 'SEAM', 'SHARP', 'CREASE', 'BEVEL', 'FREESTYLE'], (optional)*) – Edge Tag, The edge flag to tag when selecting the shortest path
- **use_face_step** (*boolean, (optional)*) – Face Stepping, Traverse connected faces (includes diagonals and edge-rings)
- **use_topology_distance** (*boolean, (optional)*) – Topology Distance, Find the minimum number of steps, ignoring spatial distance
- **use_fill** (*boolean, (optional)*) – Fill Region, Select all paths between the source/destination elements
- **skip** (*int in [0, 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.mesh.**shortest_path_select(*, edge_mode='SELECT', use_face_step=False, use_topology_distance=False, use_fill=False, skip=0, nth=1, offset=0)**

Selected shortest path between two vertices/edges/faces

PARAMETERS:

- **edge_mode** (*enum in ['SELECT', 'SEAM', 'SHARP', 'CREASE', 'BEVEL', 'FREESTYLE'], (optional)*) – Edge Tag, The edge flag to tag when selecting the shortest path
- **use_face_step** (*boolean, (optional)*) – Face Stepping, Traverse connected faces (includes diagonals and edge-rings)
- **use_topology_distance** (*boolean, (optional)*) – Topology Distance, Find the minimum number of steps, ignoring spatial distance
- **use_fill** (*boolean, (optional)*) – Fill Region, Select all paths between the source/destination elements
- **skip** (*int in [0, 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.mesh.**smooth_normals(*, factor=0.5)**

Smooth custom normals based on adjacent vertex normals

PARAMETERS:

factor (*float in [0, 1], (optional)*) – Factor, Specifies weight of smooth vs original normal

bpy.ops.mesh.**solidify(*, thickness=0.01)**

Create a solid skin by extruding, compensating for sharp angles

PARAMETERS:

thickness (*float in [-10000, 10000], (optional)*) – Thickness

bpy.ops.mesh.**sort_elements(*, type='VIEW_ZAXIS', elements={'VERT'}, reverse=False, seed=0)**

The order of selected vertices/edges/faces is modified, based on a given method

PARAMETERS:

- **type** (*enum in ['VIEW_ZAXIS', 'VIEW_XAXIS', 'CURSOR_DISTANCE', 'MATERIAL', 'SELECTED', 'RANDOMIZE', 'REVERSE'], (optional)*) –

  Type, Type of reordering operation to apply

  - `VIEW_ZAXIS` View Z Axis – Sort selected elements from farthest to nearest one in current view.
  - `VIEW_XAXIS` View X Axis – Sort selected elements from left to right one in current view.
  - `CURSOR_DISTANCE` Cursor Distance – Sort selected elements from nearest to farthest from 3D cursor.
  - `MATERIAL` Material – Sort selected faces from smallest to greatest material index.
  - `SELECTED` Selected – Move all selected elements in first places, preserving their relative order. Warning: This will affect unselected elements' indices as well.
  - `RANDOMIZE` Randomize – Randomize order of selected elements.
  - `REVERSE` Reverse – Reverse current order of selected elements.
- **elements** (*enum set in {'VERT', 'EDGE', 'FACE'}, (optional)*) – Elements, Which elements to affect (vertices, edges and/or faces)
- **reverse** (*boolean, (optional)*) – Reverse, Reverse the sorting effect
- **seed** (*int in [0, inf], (optional)*) – Seed, Seed for random-based operations

bpy.ops.mesh.**spin(*, steps=12, dupli=False, angle=1.5708, use_auto_merge=True, use_normal_flip=False, center=(0.0, 0.0, 0.0), axis=(0.0, 0.0, 0.0))**

Extrude selected vertices in a circle around the cursor in indicated viewport

PARAMETERS:

- **steps** (*int in [0, 1000000], (optional)*) – Steps, Steps
- **dupli** (*boolean, (optional)*) – Use Duplicates
- **angle** (*float in [-inf, inf], (optional)*) – Angle, Rotation for each step
- **use_auto_merge** (*boolean, (optional)*) – Auto Merge, Merge first/last when the angle is a full revolution
- **use_normal_flip** (*boolean, (optional)*) – Flip Normals
- **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.mesh.**split()**

Split off selected geometry from connected unselected geometry

bpy.ops.mesh.**split_normals()**

Split custom normals of selected vertices

bpy.ops.mesh.**subdivide(*, number_cuts=1, smoothness=0.0, ngon=True, quadcorner='STRAIGHT_CUT', fractal=0.0,**

fractal_along_normal=0.0, seed=0)

Subdivide selected edges

PARAMETERS:

- **number_cuts** (*int in [1, 100], (optional)*) – Number of Cuts
- **smoothness** (*float in [0, 1000], (optional)*) – Smoothness, Smoothness factor
- **ngon** (*boolean, (optional)*) – Create N-Gons, When disabled, newly created faces are limited to 3 and 4 sided faces
- **quadcorner** (*enum in ['INNERVERT', 'PATH', 'STRAIGHT_CUT', 'FAN'], (optional)*) – Quad Corner Type, How to subdivide quad corners (anything other than Straight Cut will prevent n-gons)
- **fractal** (*float in [0, 1e+06], (optional)*) – Fractal, Fractal randomness factor
- **fractal_along_normal** (*float in [0, 1], (optional)*) – Along Normal, Apply fractal displacement along normal only
- **seed** (*int in [0, inf], (optional)*) – Random Seed, Seed for the random number generator

bpy.ops.mesh.**subdivide_edgering(*, number_cuts=10, interpolation='PATH', smoothness=1.0, profile_shape_factor=0.0, profile_shape='SMOOTH')**

Subdivide perpendicular edges to the selected edge-ring

PARAMETERS:

- **number_cuts** (*int in [0, 1000], (optional)*) – Number of Cuts
- **interpolation** (*enum in ['LINEAR', 'PATH', 'SURFACE'], (optional)*) – Interpolation, Interpolation method
- **smoothness** (*float in [0, 1000], (optional)*) – Smoothness, Smoothness factor
- **profile_shape_factor** (*float in [-1000, 1000], (optional)*) – Profile Factor, How much intermediary new edges are shrunk/expanded
- **profile_shape** (enum in Proportional Falloff Curve Only Items, (optional)) – Profile Shape, Shape of the profile

bpy.ops.mesh.**symmetrize(*, direction='NEGATIVE_X', threshold=0.0001)**

Enforce symmetry (both form and topological) across an axis

PARAMETERS:

- **direction** (enum in Symmetrize Direction Items, (optional)) – Direction, Which sides to copy from and to
- **threshold** (*float in [0, 10], (optional)*) – Threshold, Limit for snap middle vertices to the axis center

bpy.ops.mesh.**symmetry_snap(*, direction='NEGATIVE_X', threshold=0.05, factor=0.5, use_center=True)**

Snap vertex pairs to their mirrored locations

PARAMETERS:

- **direction** (enum in Symmetrize Direction Items, (optional)) – Direction, Which sides to copy from and to
- **threshold** (*float in [0, 10], (optional)*) – Threshold, Distance within which matching vertices are searched
- **factor** (*float in [0, 1], (optional)*) – Factor, Mix factor of the locations of the vertices
- **use_center** (*boolean, (optional)*) – Center, Snap middle vertices to the axis center

bpy.ops.mesh.**tris_convert_to_quads(*, face_threshold=0.698132, shape_threshold=0.698132, topology_influence=0.0, uvs=False, vcols=False, seam=False, sharp=False, materials=False, deselect_joined=False)**

Join triangles into quads

PARAMETERS:

- **face_threshold** (*float in [0, 3.14159], (optional)*) – Max Face Angle, Face angle limit
- **shape_threshold** (*float in [0, 3.14159], (optional)*) – Max Shape Angle, Shape angle limit
- **topology_influence** (*float in [0, 2], (optional)*) – Topology Influence, How much to prioritize regular grids of quads as well as quads that touch existing quads
- **uvs** (*boolean, (optional)*) – Compare UVs
- **vcols** (*boolean, (optional)*) – Compare Color Attributes
- **seam** (*boolean, (optional)*) – Compare Seam
- **sharp** (*boolean, (optional)*) – Compare Sharp

- **materials** (*boolean, (optional)*) – Compare Materials
- **deselect_joined** (*boolean, (optional)*) – Deselect Joined, Only select remaining triangles that were not merged

bpy.ops.mesh.**unsubdivide(*, iterations=2)**

Un-subdivide selected edges and faces

PARAMETERS:

iterations (*int in [1, 1000], (optional)*) – Iterations, Number of times to un-subdivide

bpy.ops.mesh.**uv_texture_add()**

Add UV map

bpy.ops.mesh.**uv_texture_remove()**

Remove UV map

bpy.ops.mesh.**uvs_reverse()**

Flip direction of UV coordinates inside faces

bpy.ops.mesh.**uvs_rotate(*, use_ccw=False)**

Rotate UV coordinates inside faces

PARAMETERS:

use_ccw (*boolean, (optional)*) – Counter Clockwise

bpy.ops.mesh.**vert_connect()**

Connect selected vertices of faces, splitting the face

bpy.ops.mesh.**vert_connect_concave()**

Make all faces convex

bpy.ops.mesh.**vert_connect_nonplanar(*, angle_limit=0.0872665)**

Split non-planar faces that exceed the angle threshold

PARAMETERS:

angle_limit (*float in [0, 3.14159], (optional)*) – Max Angle, Angle limit

bpy.ops.mesh.**vert_connect_path()**

Connect vertices by their selection order, creating edges, splitting faces

bpy.ops.mesh.**vertices_smooth(*, factor=0.0, repeat=1, xaxis=True, yaxis=True, zaxis=True, wait_for_input=True)**

Flatten angles of selected vertices

PARAMETERS:

- **factor** (*float in [-10, 10], (optional)*) – Smoothing, Smoothing factor
- **repeat** (*int in [1, 1000], (optional)*) – Repeat, Number of times to smooth the mesh
- **xaxis** (*boolean, (optional)*) – X-Axis, Smooth along the X axis
- **yaxis** (*boolean, (optional)*) – Y-Axis, Smooth along the Y axis
- **zaxis** (*boolean, (optional)*) – Z-Axis, Smooth along the Z axis
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.mesh.**vertices_smooth_laplacian(*, repeat=1, lambda_factor=1.0, lambda_border=5e-05, use_x=True, use_y=True, use_z=True, preserve_volume=True)**

Laplacian smooth of selected vertices

PARAMETERS:

- **repeat** (*int in [1, 1000], (optional)*) – Number of iterations to smooth the mesh

- **lambda_factor** (*float in [1e-07, 1000], (optional)*) – Lambda factor
- **lambda_border** (*float in [1e-07, 1000], (optional)*) – Lambda factor in border
- **use_x** (*boolean, (optional)*) – Smooth X Axis, Smooth object along X axis
- **use_y** (*boolean, (optional)*) – Smooth Y Axis, Smooth object along Y axis
- **use_z** (*boolean, (optional)*) – Smooth Z Axis, Smooth object along Z axis
- **preserve_volume** (*boolean, (optional)*) – Preserve Volume, Apply volume preservation after smooth

bpy.ops.mesh.**wireframe(*, use_boundary=True, use_even_offset=True, use_relative_offset=False, use_replace=True, thickness=0.01, offset=0.01, use_crease=False, crease_weight=0.01)**

Create a solid wireframe from faces

PARAMETERS:

- **use_boundary** (*boolean, (optional)*) – Boundary, Inset face boundaries
- **use_even_offset** (*boolean, (optional)*) – Offset Even, Scale the offset to give more even thickness
- **use_relative_offset** (*boolean, (optional)*) – Offset Relative, Scale the offset by surrounding geometry
- **use_replace** (*boolean, (optional)*) – Replace, Remove original faces
- **thickness** (*float in [0, 10000], (optional)*) – Thickness
- **offset** (*float in [0, 10000], (optional)*) – Offset
- **use_crease** (*boolean, (optional)*) – Crease, Crease hub edges for an improved subdivision surface
- **crease_weight** (*float in [0, 1000], (optional)*) – Crease Weight