Object Operators

bpy.ops.object.**add(*, radius=1.0, type='EMPTY', 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))**

Add an object to the scene

PARAMETERS:

- **radius** (*float in [0, inf], (optional)*) – Radius
- **type** (enum in Object Type Items, (optional)) – Type
- **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.object.**add_modifier_menu()**

Undocumented, consider contributing.

FILE:

startup/bl_ui/properties_data_modifier.py:283

bpy.ops.object.**add_named(*, linked=False, name='', session_uid=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)), drop_x=0, drop_y=0)**

Add named object

PARAMETERS:

- **linked** (*boolean, (optional)*) – Linked, Duplicate object but not object data, linking to the original data
- **name** (*string, (optional, never None)*) – Name, Name of the data-block to use by the operator
- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **matrix** (`mathutils.Matrix` of 4 * 4 items in [-inf, inf], (optional)) – Matrix
- **drop_x** (*int in [-inf, inf], (optional)*) – Drop X, X-coordinate (screen space) to place the new object under
- **drop_y** (*int in [-inf, inf], (optional)*) – Drop Y, Y-coordinate (screen space) to place the new object under

bpy.ops.object.**align(*, bb_quality=True, align_mode='OPT_2', relative_to='OPT_4', align_axis={})**

Align objects

PARAMETERS:

- **bb_quality** (*boolean, (optional)*) – High Quality, Enables high quality but slow calculation of the bounding box for perfect results on comple shape meshes with rotation/scale
- **align_mode** (*enum in ['OPT_1', 'OPT_2', 'OPT_3'], (optional)*) – Align Mode, Side of object to use for alignment
- **relative_to** (*enum in ['OPT_1', 'OPT_2', 'OPT_3', 'OPT_4'], (optional)*) –

  Relative To, Reference location to align to

  - `OPT_1` Scene Origin – Use the scene origin as the position for the selected objects to align to.
  - `OPT_2` 3D Cursor – Use the 3D cursor as the position for the selected objects to align to.
  - `OPT_3` Selection – Use the selected objects as the position for the selected objects to align to.
  - `OPT_4` Active – Use the active object as the position for the selected objects to align to.

- **align_axis** (*enum set in {'X', 'Y', 'Z'}, (optional)*) – Align, Align to axis

FILE:
startup/bl_operators/object_align.py:386

bpy.ops.object.**anim_transforms_to_deltas()**

Convert object animation for normal transforms to delta transforms

FILE:
startup/bl_operators/object.py:794

bpy.ops.object.**armature_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))**

Add an armature object to the scene

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.object.**assign_property_defaults(*, process_data=True, process_bones=True)**

Assign the current values of custom properties as their defaults, for use as part of the rest pose state in NLA track mixing

PARAMETERS:

- **process_data** (*boolean, (optional)*) – Process data properties
- **process_bones** (*boolean, (optional)*) – Process bone properties

FILE:
startup/bl_operators/object.py:947

bpy.ops.object.**bake(*, type='COMBINED', pass_filter={}, filepath='', width=512, height=512, margin=16, margin_type='EXTEND', use_selected_to_active=False, max_ray_distance=0.0, cage_extrusion=0.0, cage_object='', normal_space='TANGENT', normal_r='POS_X', normal_g='POS_Y', normal_b='POS_Z', target='IMAGE_TEXTURES', save_mode='INTERNAL', use_clear=False, use_cage=False, use_split_materials=False, use_automatic_name=False, uv_layer='')**

Bake image textures of selected objects

PARAMETERS:

- **type** (enum in Bake Pass Type Items, (optional)) – Type, Type of pass to bake, some of them may not be supported by the current render engine
- **pass_filter** (enum set in Bake Pass Filter Type Items, (optional)) – Pass Filter, Filter to combined, diffuse, glossy, transmission and subsurfac passes
- **filepath** (*string, (optional, never None)*) – File Path, Image filepath to use when saving externally
- **width** (*int in [1, inf], (optional)*) – Width, Horizontal dimension of the baking map (external only)
- **height** (*int in [1, inf], (optional)*) – Height, Vertical dimension of the baking map (external only)
- **margin** (*int in [0, inf], (optional)*) – Margin, Extends the baked result as a post process filter
- **margin_type** (enum in Bake Margin Type Items, (optional)) – Margin Type, Which algorithm to use to generate the margin
- **use_selected_to_active** (*boolean, (optional)*) – Selected to Active, Bake shading on the surface of selected objects to the active object

- **max_ray_distance** (*float in [0, inf], (optional)*) – Max Ray Distance, The maximum ray distance for matching points between the active ar selected objects. If zero, there is no limit
- **cage_extrusion** (*float in [0, inf], (optional)*) – Cage Extrusion, Inflate the active object by the specified distance for baking. This helps matching to points nearer to the outside of the selected object meshes
- **cage_object** (*string, (optional, never None)*) – Cage Object, Object to use as cage, instead of calculating the cage from the active object with cage extrusion
- **normal_space** (enum in Normal Space Items, (optional)) – Normal Space, Choose normal space for baking
- **normal_r** (enum in Normal Swizzle Items, (optional)) – R, Axis to bake in red channel
- **normal_g** (enum in Normal Swizzle Items, (optional)) – G, Axis to bake in green channel
- **normal_b** (enum in Normal Swizzle Items, (optional)) – B, Axis to bake in blue channel
- **target** (enum in Bake Target Items, (optional)) – Target, Where to output the baked map
- **save_mode** (enum in Bake Save Mode Items, (optional)) – Save Mode, Where to save baked image textures
- **use_clear** (*boolean, (optional)*) – Clear, Clear images before baking (only for internal saving)
- **use_cage** (*boolean, (optional)*) – Cage, Cast rays to active object from a cage
- **use_split_materials** (*boolean, (optional)*) – Split Materials, Split baked maps per material, using material name in output file (external only)
- **use_automatic_name** (*boolean, (optional)*) – Automatic Name, Automatically name the output file with the pass type
- **uv_layer** (*string, (optional, never None)*) – UV Layer, UV layer to override active

bpy.ops.object.**bake_image()**

Bake image textures of selected objects

bpy.ops.object.**camera_add(*, 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))**

Add a camera object to the scene

PARAMETERS:

- **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.object.**clear_override_library()**

Delete the selected local overrides and relink their usages to the linked data-blocks if possible, else reset them and mark them as non editable

bpy.ops.object.**collection_add()**

Add an object to a new collection

bpy.ops.object.**collection_external_asset_drop(*, session_uid=0, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0), use_instance=True, drop_x=0, drop_y=0, collection='')**

Add the dragged collection to the scene

PARAMETERS:

- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **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
- **use_instance** (*boolean, (optional)*) – Instance, Add the dropped collection as collection instance
- **drop_x** (*int in [-inf, inf], (optional)*) – Drop X, X-coordinate (screen space) to place the new object under
- **drop_y** (*int in [-inf, inf], (optional)*) – Drop Y, Y-coordinate (screen space) to place the new object under
- **collection** (*enum in [], (optional)*) – Collection

bpy.ops.object.**collection_instance_add(*, name='Collection', collection='', align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0)**
scale=(0.0, 0.0, 0.0), session_uid=0, drop_x=0, drop_y=0)

Add a collection instance

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Collection name to add
- **collection** (*enum in [], (optional)*) – Collection
- **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
- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **drop_x** (*int in [-inf, inf], (optional)*) – Drop X, X-coordinate (screen space) to place the new object under
- **drop_y** (*int in [-inf, inf], (optional)*) – Drop Y, Y-coordinate (screen space) to place the new object under

bpy.ops.object.**collection_link(*, collection='')**

Add an object to an existing collection

PARAMETERS:

collection (*enum in [], (optional)*) – Collection

bpy.ops.object.**collection_objects_select()**

Select all objects in collection

bpy.ops.object.**collection_remove()**

Remove the active object from this collection

bpy.ops.object.**collection_unlink()**

Unlink the collection from all objects

bpy.ops.object.**constraint_add(*, type='')**

Add a constraint to the active object

PARAMETERS:

type (*enum in [], (optional)*) – Type

bpy.ops.object.**constraint_add_with_targets(*, type='')**

Add a constraint to the active object, with target (where applicable) set to the selected objects/bones

PARAMETERS:

type (*enum in [], (optional)*) – Type

bpy.ops.object.**constraints_clear()**

Clear all constraints from the selected objects

bpy.ops.object.**constraints_copy()**

Copy constraints to other selected objects

bpy.ops.object.**convert(*, target='MESH', keep_original=False, merge_customdata=True, thickness=5, faces=True, offset=0.01)**

Convert selected objects to another type

PARAMETERS:

- **target** (*enum in ['CURVE', 'MESH', 'CURVES', 'GREASEPENCIL'], (optional)*) –
  Target, Type of object to convert to
  - `CURVE` Curve – Curve from Mesh or Text objects.
  - `MESH` Mesh – Mesh from Curve, Surface, Metaball, or Text objects.
  - `CURVES` Curves – Curves from evaluated curve data.
  - `GREASEPENCIL` Grease Pencil – Grease Pencil from Curve or Mesh objects.
- **keep_original** (*boolean, (optional)*) – Keep Original, Keep original objects instead of replacing them
- **merge_customdata** (*boolean, (optional)*) – Merge UVs, Merge UV coordinates that share a vertex to account for imprecision in some modifiers
- **thickness** (*int in [1, 100], (optional)*) – Thickness
- **faces** (*boolean, (optional)*) – Export Faces, Export faces as filled strokes
- **offset** (*float in [0, inf], (optional)*) – Stroke Offset, Offset strokes from fill

bpy.ops.object.**correctivesmooth_bind(*, modifier='')**

Bind base pose in Corrective Smooth modifier

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**curves_empty_hair_add(*, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Add an empty curve object to the scene with the selected mesh as surface

PARAMETERS:

- **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.object.**curves_random_add(*, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Add a curves object with random curves to the scene

PARAMETERS:

- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –
  Align, The alignment of the new object
  - `WORLD` World – Align the new object to the world.

  - `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.object.**data_instance_add(*, name='', session_uid=0, type='ACTION', align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0), drop_x=0, drop_y=0)**

Add an object data instance

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the data-block to use by the operator
- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **type** (enum in Id Type Items, (optional)) – Type
- **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
- **drop_x** (*int in [-inf, inf], (optional)*) – Drop X, X-coordinate (screen space) to place the new object under
- **drop_y** (*int in [-inf, inf], (optional)*) – Drop Y, Y-coordinate (screen space) to place the new object under

bpy.ops.object.**data_transfer(*, use_reverse_transfer=False, use_freeze=False, data_type='', use_create=True, vert_mapping='NEAREST', edge_mapping='NEAREST', loop_mapping='NEAREST_POLYNOR', poly_mapping='NEAREST', use_auto_transform=False, use_object_transform=True, use_max_distance=False, max_distance=1.0, ray_radius=0.0, islands_precision=0.1, layers_select_src='ACTIVE', layers_select_dst='ACTIVE', mix_mode='REPLACE', mix_factor=1.0)**

Transfer data layer(s) (weights, edge sharp, etc.) from active to selected meshes

PARAMETERS:

- **use_reverse_transfer** (*boolean, (optional)*) – Reverse Transfer, Transfer from selected objects to active one
- **use_freeze** (*boolean, (optional)*) – Freeze Operator, Prevent changes to settings to re-run the operator, handy to change several things at once with heavy geometry
- **data_type** (*enum in ['VGROUP_WEIGHTS', 'BEVEL_WEIGHT_VERT', 'COLOR_VERTEX', 'SHARP_EDGE', 'SEAM', 'CREASE', 'BEVEL_WEIGHT_EDGE', 'FREESTYLE_EDGE', 'CUSTOM_NORMAL', 'COLOR_CORNER', 'UV', 'SMOOTH', 'FREESTYLE_FACE'], (optional)*) –

  Data Type, Which data to transfer

  - `VGROUP_WEIGHTS` Vertex Group(s) – Transfer active or all vertex groups.
  - `BEVEL_WEIGHT_VERT` Bevel Weight – Transfer bevel weights.
  - `COLOR_VERTEX` Colors – Color Attributes.
  - `SHARP_EDGE` Sharp – Transfer sharp mark.
  - `SEAM` UV Seam – Transfer UV seam mark.
  - `CREASE` Subdivision Crease – Transfer crease values.
  - `BEVEL_WEIGHT_EDGE` Bevel Weight – Transfer bevel weights.
  - `FREESTYLE_EDGE` Freestyle Mark – Transfer Freestyle edge mark.
  - `CUSTOM_NORMAL` Custom Normals – Transfer custom normals.
  - `COLOR_CORNER` Colors – Color Attributes.
  - `UV` UVs – Transfer UV layers.

- `SMOOTH` Smooth – Transfer flat/smooth mark.
- `FREESTYLE_FACE` Freestyle Mark – Transfer Freestyle face mark.

- **use_create** (*boolean, (optional)*) – Create Data, Add data layers on destination meshes if needed
- **vert_mapping** (enum in Dt Method Vertex Items, (optional)) – Vertex Mapping, Method used to map source vertices to destination ones
- **edge_mapping** (enum in Dt Method Edge Items, (optional)) – Edge Mapping, Method used to map source edges to destination ones
- **loop_mapping** (enum in Dt Method Loop Items, (optional)) – Face Corner Mapping, Method used to map source faces' corners to destination ones
- **poly_mapping** (enum in Dt Method Poly Items, (optional)) – Face Mapping, Method used to map source faces to destination ones
- **use_auto_transform** (*boolean, (optional)*) – Auto Transform, Automatically compute transformation to get the best possible match betwee source and destination meshes.Warning: Results will never be as good as manual matching of objects
- **use_object_transform** (*boolean, (optional)*) – Object Transform, Evaluate source and destination meshes in global space
- **use_max_distance** (*boolean, (optional)*) – Only Neighbor Geometry, Source elements must be closer than given distance from destination one
- **max_distance** (*float in [0, inf], (optional)*) – Max Distance, Maximum allowed distance between source and destination element, for non-topology mappings
- **ray_radius** (*float in [0, inf], (optional)*) – Ray Radius, 'Width' of rays (especially useful when raycasting against vertices or edges)
- **islands_precision** (*float in [0, 10], (optional)*) – Islands Precision, Factor controlling precision of islands handling (the higher, the better the results)
- **layers_select_src** (enum in Dt Layers Select Src Items, (optional)) – Source Layers Selection, Which layers to transfer, in case of multi-laye types
- **layers_select_dst** (enum in Dt Layers Select Dst Items, (optional)) – Destination Layers Matching, How to match source and destination layers
- **mix_mode** (enum in Dt Mix Mode Items, (optional)) – Mix Mode, How to affect destination elements with source values
- **mix_factor** (*float in [0, 1], (optional)*) – Mix Factor, Factor to use when applying data to destination (exact behavior depends on mix mod

bpy.ops.object.**datalayout_transfer(*, modifier='', data_type='', use_delete=False, layers_select_src='ACTIVE', layers_select_dst='ACTIVE')**

Transfer layout of data layer(s) from active to selected meshes

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **data_type** (*enum in ['VGROUP_WEIGHTS', 'BEVEL_WEIGHT_VERT', 'COLOR_VERTEX', 'SHARP_EDGE', 'SEAM', 'CREASE', 'BEVEL_WEIGHT_EDGE', 'FREESTYLE_EDGE', 'CUSTOM_NORMAL', 'COLOR_CORNER', 'UV', 'SMOOTH', 'FREESTYLE_FACE'], (optional)*) –

  Data Type, Which data to transfer

  - `VGROUP_WEIGHTS` Vertex Group(s) – Transfer active or all vertex groups.
  - `BEVEL_WEIGHT_VERT` Bevel Weight – Transfer bevel weights.
  - `COLOR_VERTEX` Colors – Color Attributes.
  - `SHARP_EDGE` Sharp – Transfer sharp mark.
  - `SEAM` UV Seam – Transfer UV seam mark.
  - `CREASE` Subdivision Crease – Transfer crease values.
  - `BEVEL_WEIGHT_EDGE` Bevel Weight – Transfer bevel weights.
  - `FREESTYLE_EDGE` Freestyle Mark – Transfer Freestyle edge mark.
  - `CUSTOM_NORMAL` Custom Normals – Transfer custom normals.
  - `COLOR_CORNER` Colors – Color Attributes.
  - `UV` UVs – Transfer UV layers.
  - `SMOOTH` Smooth – Transfer flat/smooth mark.
  - `FREESTYLE_FACE` Freestyle Mark – Transfer Freestyle face mark.

- **use_delete** (*boolean, (optional)*) – Exact Match, Also delete some data layers from destination if necessary, so that it matches exactly sourc

- **layers_select_src** (enum in Dt Layers Select Src Items, (optional)) – Source Layers Selection, Which layers to transfer, in case of multi-laye types
- **layers_select_dst** (enum in Dt Layers Select Dst Items, (optional)) – Destination Layers Matching, How to match source and destination layers

bpy.ops.object.**delete(*, use_global=False, confirm=True)**

Delete selected objects

PARAMETERS:

- **use_global** (*boolean, (optional)*) – Delete Globally, Remove object from all scenes
- **confirm** (*boolean, (optional)*) – Confirm, Prompt for confirmation

bpy.ops.object.**drop_geometry_nodes(*, session_uid=0, show_datablock_in_modifier=True)**

Undocumented, consider contributing.

PARAMETERS:

- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the geometry node group being dropped
- **show_datablock_in_modifier** (*boolean, (optional)*) – Show the datablock selector in the modifier

bpy.ops.object.**drop_named_material(*, name='', session_uid=0)**

Undocumented, consider contributing.

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the data-block to use by the operator
- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator

bpy.ops.object.**duplicate(*, linked=False, mode='TRANSLATION')**

Duplicate selected objects

PARAMETERS:

- **linked** (*boolean, (optional)*) – Linked, Duplicate object but not object data, linking to the original data
- **mode** (enum in Transform Mode Type Items, (optional)) – Mode

bpy.ops.object.**duplicate_move(*, OBJECT_OT_duplicate=None, TRANSFORM_OT_translate=None)**

Duplicate the selected objects and move them

PARAMETERS:

- **OBJECT_OT_duplicate** (`OBJECT_OT_duplicate`, (optional)) – Duplicate Objects, Duplicate selected objects
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.object.**duplicate_move_linked(*, OBJECT_OT_duplicate=None, TRANSFORM_OT_translate=None)**

Duplicate the selected objects, but not their object data, and move them

PARAMETERS:

- **OBJECT_OT_duplicate** (`OBJECT_OT_duplicate`, (optional)) – Duplicate Objects, Duplicate selected objects
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.object.**duplicates_make_real(*, use_base_parent=False, use_hierarchy=False)**

Make instanced objects attached to this object real

PARAMETERS:

- **use_base_parent** (*boolean, (optional)*) – Parent, Parent newly created objects to the original instancer
- **use_hierarchy** (*boolean, (optional)*) – Keep Hierarchy, Maintain parent child relationships

bpy.ops.object.**editmode_toggle()**

Toggle object's edit mode

bpy.ops.object.**effector_add(*, type='FORCE', 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))**

Add an empty object with a physics effector to the scene

PARAMETERS:

- **type** (*enum in ['FORCE', 'WIND', 'VORTEX', 'MAGNET', 'HARMONIC', 'CHARGE', 'LENNARDJ', 'TEXTURE', 'GUIDE', 'BOID', 'TURBULENCE', 'DRAG', 'FLUID'], (optional)*) – Type
- **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.object.**empty_add(*, type='PLAIN_AXES', radius=1.0, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.(0.0, 0.0))**

Add an empty object to the scene

PARAMETERS:

- **type** (enum in Object Empty Drawtype Items, (optional)) – Type
- **radius** (*float in [0, inf], (optional)*) – Radius
- **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.object.**empty_image_add(*, filepath='', hide_props_region=True, check_existing=False, filter_blender=False, filter_backup=False filter_image=True, filter_movie=True, filter_python=False, filter_font=False, filter_sound=False, filter_text=False, filter_archive=False, filter_btx=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=9, relative_path=True, show_multiview=False, use_multiview=False, display_type='DEFAULT', sort_method='', name='', session_uid=0, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0), background=False)**

Add an empty image type to scene with data

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **hide_props_region** (*boolean, (optional)*) – Hide Operator Properties, Collapse the region displaying the operator settings
- **check_existing** (*boolean, (optional)*) – Check Existing, Check and warn on overwriting existing files
- **filter_blender** (*boolean, (optional)*) – Filter .blend files
- **filter_backup** (*boolean, (optional)*) – Filter .blend files
- **filter_image** (*boolean, (optional)*) – Filter image files
- **filter_movie** (*boolean, (optional)*) – Filter movie files

- **filter_python** (*boolean, (optional)*) – Filter Python files
- **filter_font** (*boolean, (optional)*) – Filter font files
- **filter_sound** (*boolean, (optional)*) – Filter sound files
- **filter_text** (*boolean, (optional)*) – Filter text files
- **filter_archive** (*boolean, (optional)*) – Filter archive files
- **filter_btx** (*boolean, (optional)*) – Filter btx files
- **filter_collada** (*boolean, (optional)*) – Filter COLLADA files
- **filter_alembic** (*boolean, (optional)*) – Filter Alembic files
- **filter_usd** (*boolean, (optional)*) – Filter USD files
- **filter_obj** (*boolean, (optional)*) – Filter OBJ files
- **filter_volume** (*boolean, (optional)*) – Filter OpenVDB volume files
- **filter_folder** (*boolean, (optional)*) – Filter folders
- **filter_blenlib** (*boolean, (optional)*) – Filter Blender IDs
- **filemode** (*int in [1, 9], (optional)*) – File Browser Mode, The setting for the file browser mode to load a .blend file, a library or a special fil
- **relative_path** (*boolean, (optional)*) – Relative Path, Select the file relative to the blend file
- **show_multiview** (*boolean, (optional)*) – Enable Multi-View
- **use_multiview** (*boolean, (optional)*) – Use Multi-View
- **display_type** (*enum in ['DEFAULT', 'LIST_VERTICAL', 'LIST_HORIZONTAL', 'THUMBNAIL'], (optional)*) –

  Display Type

  - `DEFAULT` Default – Automatically determine display type for files.
  - `LIST_VERTICAL` Short List – Display files as short list.
  - `LIST_HORIZONTAL` Long List – Display files as a detailed list.
  - `THUMBNAIL` Thumbnails – Display files as thumbnails.
- **sort_method** (*enum in ['DEFAULT', 'FILE_SORT_ALPHA', 'FILE_SORT_EXTENSION', 'FILE_SORT_TIME', 'FILE_SORT_SIZE', 'ASSET_CATALOG'], (optional)*) –

  File sorting mode

  - `DEFAULT` Default – Automatically determine sort method for files.
  - `FILE_SORT_ALPHA` Name – Sort the file list alphabetically.
  - `FILE_SORT_EXTENSION` Extension – Sort the file list by extension/type.
  - `FILE_SORT_TIME` Modified Date – Sort files by modification time.
  - `FILE_SORT_SIZE` Size – Sort files by size.
  - `ASSET_CATALOG` Asset Catalog – Sort the asset list so that assets in the same catalog are kept together. Within a single catalog, asse are ordered by name. The catalogs are in order of the flattened catalog hierarchy..
- **name** (*string, (optional, never None)*) – Name, Name of the data-block to use by the operator
- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **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
- **background** (*boolean, (optional)*) – Put in Background, Make the image render behind all objects

bpy.ops.object.**explode_refresh(*, modifier='')**

Refresh data in the Explode modifier

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**forcefield_toggle()**

Toggle object's force field

bpy.ops.object.**geometry_node_bake_delete_single(*, session_uid=0, modifier_name='', bake_id=0)**

Delete baked data of a single bake node or simulation

PARAMETERS:

- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **modifier_name** (*string, (optional, never None)*) – Modifier Name, Name of the modifier that contains the node
- **bake_id** (*int in [0, inf], (optional)*) – Bake ID, Nested node id of the node

bpy.ops.object.**geometry_node_bake_pack_single(*, session_uid=0, modifier_name='', bake_id=0)**

Pack baked data from disk into the .blend file

PARAMETERS:

- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **modifier_name** (*string, (optional, never None)*) – Modifier Name, Name of the modifier that contains the node
- **bake_id** (*int in [0, inf], (optional)*) – Bake ID, Nested node id of the node

bpy.ops.object.**geometry_node_bake_single(*, session_uid=0, modifier_name='', bake_id=0)**

Bake a single bake node or simulation

PARAMETERS:

- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **modifier_name** (*string, (optional, never None)*) – Modifier Name, Name of the modifier that contains the node
- **bake_id** (*int in [0, inf], (optional)*) – Bake ID, Nested node id of the node

bpy.ops.object.**geometry_node_bake_unpack_single(*, session_uid=0, modifier_name='', bake_id=0, method='USE_LOCAL')**

Unpack baked data from the .blend file to disk

PARAMETERS:

- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **modifier_name** (*string, (optional, never None)*) – Modifier Name, Name of the modifier that contains the node
- **bake_id** (*int in [0, inf], (optional)*) – Bake ID, Nested node id of the node
- **method** (*enum in ['USE_LOCAL', 'WRITE_LOCAL', 'USE_ORIGINAL', 'WRITE_ORIGINAL'], (optional)*) – Method, How to unpack

bpy.ops.object.**geometry_node_tree_copy_assign()**

Copy the active geometry node group and assign it to the active modifier

bpy.ops.object.**geometry_nodes_input_attribute_toggle(*, input_name='', modifier_name='')**

Switch between an attribute and a single value to define the data for every element

PARAMETERS:

- **input_name** (*string, (optional, never None)*) – Input Name
- **modifier_name** (*string, (optional, never None)*) – Modifier Name

bpy.ops.object.**geometry_nodes_move_to_nodes(*, use_selected_objects=False)**

Move inputs and outputs from in the modifier to a new node group

PARAMETERS:

use_selected_objects (*boolean, (optional)*) – Selected Objects, Affect all selected objects instead of just the active object

FILE:

startup/bl_operators/geometry_nodes.py:260

bpy.ops.object.**grease_pencil_add(*, type='EMPTY', use_in_front=True, stroke_depth_offset=0.05, use_lights=False, stroke_depth_order='3D', radius=1.0, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Add a Grease Pencil object to the scene

PARAMETERS:

- **type** (enum in Object Gpencil Type Items, (optional)) – Type
- **use_in_front** (*boolean, (optional)*) – Show In Front, Show Line Art Grease Pencil in front of everything
- **stroke_depth_offset** (*float in [0, inf], (optional)*) – Stroke Offset, Stroke offset for the Line Art modifier
- **use_lights** (*boolean, (optional)*) – Use Lights, Use lights for this Grease Pencil object
- **stroke_depth_order** (*enum in ['2D', '3D'], (optional)*) – Stroke Depth Order, Defines how the strokes are ordered in 3D space (for objects not displayed 'In Front')
    - `2D` 2D Layers – Display strokes using Grease Pencil layers to define order.
    - `3D` 3D Location – Display strokes using real 3D position in 3D space.
- **radius** (*float in [0, inf], (optional)*) – Radius
- **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.object.**grease_pencil_dash_modifier_segment_add(*, modifier='')**

Add a segment to the dash modifier

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**grease_pencil_dash_modifier_segment_move(*, modifier='', type='UP')**

Move the active dash segment up or down

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **type** (*enum in ['UP', 'DOWN'], (optional)*) – Type

bpy.ops.object.**grease_pencil_dash_modifier_segment_remove(*, modifier='', index=0)**

Remove the active segment from the dash modifier

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **index** (*int in [0, inf], (optional)*) – Index, Index of the segment to remove

bpy.ops.object.**grease_pencil_time_modifier_segment_add(*, modifier='')**

Add a segment to the time modifier

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**grease_pencil_time_modifier_segment_move(*, modifier='', type='UP')**

Move the active time segment up or down

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **type** (*enum in ['UP', 'DOWN'], (optional)*) – Type

bpy.ops.object.**grease_pencil_time_modifier_segment_remove(*, modifier='', index=0)**

Remove the active segment from the time modifier

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **index** (*int in [0, inf], (optional)*) – Index, Index of the segment to remove

bpy.ops.object.**hide_collection(*, collection_index=-1, toggle=False, extend=False)**

Show only objects in collection (Shift to extend)

PARAMETERS:

- **collection_index** (*int in [-1, inf], (optional)*) – Collection Index, Index of the collection to change visibility
- **toggle** (*boolean, (optional)*) – Toggle, Toggle visibility
- **extend** (*boolean, (optional)*) – Extend, Extend visibility

bpy.ops.object.**hide_render_clear_all()**

Reveal all render objects by setting the hide render flag

FILE:

startup/bl_operators/object.py:701

bpy.ops.object.**hide_view_clear(*, select=True)**

Reveal temporarily hidden objects

PARAMETERS:

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

bpy.ops.object.**hide_view_set(*, unselected=False)**

Temporarily hide objects from the viewport

PARAMETERS:

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

bpy.ops.object.**hook_add_newob()**

Hook selected vertices to a newly created object

bpy.ops.object.**hook_add_selob(*, use_bone=False)**

Hook selected vertices to the first selected object

PARAMETERS:

use_bone (*boolean, (optional)*) – Active Bone, Assign the hook to the hook object's active bone

bpy.ops.object.**hook_assign(*, modifier='')**

Assign the selected vertices to a hook

PARAMETERS:

modifier (*enum in [], (optional)*) – Modifier, Modifier number to assign to

bpy.ops.object.**hook_recenter(*, modifier='')**

Set hook center to cursor position

PARAMETERS:

modifier (*enum in [], (optional)*) – Modifier, Modifier number to assign to

bpy.ops.object.**hook_remove(*, modifier='')**

Remove a hook from the active object

PARAMETERS:

modifier (*enum in [], (optional)*) – Modifier, Modifier number to remove

bpy.ops.object.**hook_reset(*, modifier='')**

Recalculate and clear offset transformation

PARAMETERS:

modifier (*enum in [], (optional)*) – Modifier, Modifier number to assign to

bpy.ops.object.**hook_select(*, modifier='')**

Select affected vertices on mesh

PARAMETERS:

modifier (*enum in [], (optional)*) – Modifier, Modifier number to remove

bpy.ops.object.**instance_offset_from_cursor()**

Set offset used for collection instances based on cursor position

FILE:

startup/bl_operators/object.py:882

bpy.ops.object.**instance_offset_from_object()**

Set offset used for collection instances based on the active object position

FILE:

startup/bl_operators/object.py:914

bpy.ops.object.**instance_offset_to_cursor()**

Set cursor position to the offset used for collection instances

FILE:

startup/bl_operators/object.py:897

bpy.ops.object.**isolate_type_render()**

Hide unselected render objects of same type as active by setting the hide render flag

FILE:

startup/bl_operators/object.py:681

bpy.ops.object.**join()**

Join selected objects into active object

bpy.ops.object.**join_shapes()**

Copy the current resulting shape of another selected object to this one

bpy.ops.object.**join_uvs()**

Transfer UV Maps from active to selected objects (needs matching geometry)

FILE:

startup/bl_operators/object.py:582

bpy.ops.object.**laplaciandeform_bind(*, modifier='')**

Bind mesh to system in laplacian deform modifier

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**light_add(*, type='POINT', radius=1.0, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Add a light object to the scene

PARAMETERS:

- **type** (enum in Light Type Items, (optional)) – Type
- **radius** (*float in [0, inf], (optional)*) – Radius
- **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.object.**light_linking_blocker_collection_new()**

Create new light linking collection used by the active emitter

bpy.ops.object.**light_linking_blockers_link(*, link_state='INCLUDE')**

Light link selected blockers to the active emitter object

PARAMETERS:

link_state (*enum in ['INCLUDE', 'EXCLUDE'], (optional)*) –

Link State, State of the shadow linking

- `INCLUDE` Include – Include selected blockers to cast shadows from the active emitter.
- `EXCLUDE` Exclude – Exclude selected blockers from casting shadows from the active emitter.

bpy.ops.object.**light_linking_blockers_select()**

Select all objects which block light from this emitter

bpy.ops.object.**light_linking_receiver_collection_new()**

Create new light linking collection used by the active emitter

bpy.ops.object.**light_linking_receivers_link(*, link_state='INCLUDE')**

Light link selected receivers to the active emitter object

PARAMETERS:

link_state (*enum in ['INCLUDE', 'EXCLUDE'], (optional)*) –

Link State, State of the light linking

- `INCLUDE` Include – Include selected receivers to receive light from the active emitter.
- `EXCLUDE` Exclude – Exclude selected receivers from receiving light from the active emitter.

bpy.ops.object.**light_linking_receivers_select()**

Select all objects which receive light from this emitter

bpy.ops.object.**light_linking_unlink_from_collection()**

Remove this object or collection from the light linking collection

bpy.ops.object.**lightprobe_add(*, type='SPHERE', 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))**

Add a light probe object

PARAMETERS:

- **type** (*enum in ['SPHERE', 'PLANE', 'VOLUME'], (optional)*) –

  Type

  - `SPHERE` Sphere – Light probe that captures precise lighting from all directions at a single point in space.
  - `PLANE` Plane – Light probe that captures incoming light from a single direction on a plane.
  - `VOLUME` Volume – Light probe that captures low frequency lighting inside a volume.

- **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.object.**lightprobe_cache_bake(*, subset='ALL')**

Bake irradiance volume light cache

PARAMETERS:

subset (*enum in ['ALL', 'SELECTED', 'ACTIVE'], (optional)*) –

Subset, Subset of probes to update

- `ALL` All Volumes – Bake all light probe volumes.
- `SELECTED` Selected Only – Only bake selected light probe volumes.
- `ACTIVE` Active Only – Only bake the active light probe volume.

bpy.ops.object.**lightprobe_cache_free(*, subset='SELECTED')**

Delete cached indirect lighting

PARAMETERS:

subset (*enum in ['ALL', 'SELECTED', 'ACTIVE'], (optional)*) –

Subset, Subset of probes to update

- `ALL` All Light Probes – Delete all light probes' baked lighting data.
- `SELECTED` Selected Only – Only delete selected light probes' baked lighting data.
- `ACTIVE` Active Only – Only delete the active light probe's baked lighting data.

bpy.ops.object.**lineart_bake_strokes(*, bake_all=False)**

Bake Line Art for current Grease Pencil object

PARAMETERS:

bake_all (*boolean, (optional)*) – Bake All, Bake all Line Art modifiers

bpy.ops.object.**lineart_clear(*, clear_all=False)**

Clear all strokes in current Grease Pencil object

Clear all strokes in current Grease Pencil object

PARAMETERS:

clear_all (*boolean, (optional)*) – Clear All, Clear all Line Art modifier bakes

bpy.ops.object.**link_to_collection(*, collection_index=-1, is_new=False, new_collection_name='')**

Link objects to a collection

PARAMETERS:

- **collection_index** (*int in [-1, inf], (optional)*) – Collection Index, Index of the collection to move to
- **is_new** (*boolean, (optional)*) – New, Move objects to a new collection
- **new_collection_name** (*string, (optional, never None)*) – Name, Name of the newly added collection

bpy.ops.object.**location_clear(*, clear_delta=False)**

Clear the object's location

PARAMETERS:

clear_delta (*boolean, (optional)*) – Clear Delta, Clear delta location in addition to clearing the normal location transform

bpy.ops.object.**make_dupli_face()**

Convert objects into instanced faces

FILE:

startup/bl_operators/object.py:664

bpy.ops.object.**make_links_data(*, type='OBDATA')**

Transfer data from active object to selected objects

PARAMETERS:

type (*enum in ['OBDATA', 'MATERIAL', 'ANIMATION', 'GROUPS', 'DUPLICOLLECTION', 'FONTS', 'MODIFIERS', 'EFFECTS'], (optional)*) –

Type

- `OBDATA` Link Object Data – Replace assigned Object Data.
- `MATERIAL` Link Materials – Replace assigned Materials.
- `ANIMATION` Link Animation Data – Replace assigned Animation Data.
- `GROUPS` Link Collections – Replace assigned Collections.
- `DUPLICOLLECTION` Link Instance Collection – Replace assigned Collection Instance.
- `FONTS` Link Fonts to Text – Replace Text object Fonts.
- `MODIFIERS` Copy Modifiers – Replace Modifiers.
- `EFFECTS` Copy Grease Pencil Effects – Replace Grease Pencil Effects.

bpy.ops.object.**make_links_scene(*, scene='')**

Link selection to another scene

PARAMETERS:

scene (*enum in [], (optional)*) – Scene

bpy.ops.object.**make_local(*, type='SELECT_OBJECT')**

Make library linked data-blocks local to this file

PARAMETERS:

type (*enum in ['SELECT_OBJECT', 'SELECT_OBDATA', 'SELECT_OBDATA_MATERIAL', 'ALL'], (optional)*) – Type

bpy.ops.object.**make_override_library(*, collection=0)**

Create a local override of the selected linked objects, and their hierarchy of dependencies

PARAMETERS:

collection (*int in [-inf, inf], (optional)*) – Override Collection, Session UID of the directly linked collection containing the selected object, t
make an override from

bpy.ops.object.**make_single_user(*, type='SELECTED_OBJECTS', object=False, obdata=False, material=False, animation=False, obdata_animation=False)**

Make linked data local to each object

PARAMETERS:

- **type** (*enum in ['SELECTED_OBJECTS', 'ALL'], (optional)*) – Type
- **object** (*boolean, (optional)*) – Object, Make single user objects
- **obdata** (*boolean, (optional)*) – Object Data, Make single user object data
- **material** (*boolean, (optional)*) – Materials, Make materials local to each data-block
- **animation** (*boolean, (optional)*) – Object Animation, Make object animation data local to each object
- **obdata_animation** (*boolean, (optional)*) – Object Data Animation, Make object data (mesh, curve etc.) animation data local to each objec

bpy.ops.object.**material_slot_add()**

Add a new material slot

bpy.ops.object.**material_slot_assign()**

Assign active material slot to selection

bpy.ops.object.**material_slot_copy()**

Copy material to selected objects

bpy.ops.object.**material_slot_deselect()**

Deselect by active material slot

bpy.ops.object.**material_slot_move(*, direction='UP')**

Move the active material up/down in the list

PARAMETERS:

direction (*enum in ['UP', 'DOWN'], (optional)*) – Direction, Direction to move the active material towards

bpy.ops.object.**material_slot_remove()**

Remove the selected material slot

bpy.ops.object.**material_slot_remove_unused()**

Remove unused material slots

bpy.ops.object.**material_slot_select()**

Select by active material slot

bpy.ops.object.**meshdeform_bind(*, modifier='')**

Bind mesh to cage in mesh deform modifier

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**metaball_add(*, type='BALL', radius=2.0, 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))**

Add an metaball object to the scene

PARAMETERS:

- **type** (enum in Metaelem Type Items, (optional)) – Primitive

- **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.object.**mode_set(*, mode='OBJECT', toggle=False)**

Sets the object interaction mode

PARAMETERS:

- **mode** (enum in Object Mode Items, (optional)) – Mode
- **toggle** (*boolean, (optional)*) – Toggle

bpy.ops.object.**mode_set_with_submode(*, mode='OBJECT', toggle=False, mesh_select_mode={})**

Sets the object interaction mode

PARAMETERS:

- **mode** (enum in Object Mode Items, (optional)) – Mode
- **toggle** (*boolean, (optional)*) – Toggle
- **mesh_select_mode** (enum set in Mesh Select Mode Items, (optional)) – Mesh Mode

bpy.ops.object.**modifier_add(*, type='SUBSURF', use_selected_objects=False)**

Add a procedural operation/effect to the active object

PARAMETERS:

- **type** (enum in Object Modifier Type Items, (optional)) – Type
- **use_selected_objects** (*boolean, (optional)*) – Selected Objects, Affect all selected objects instead of just the active object

bpy.ops.object.**modifier_add_node_group(*, asset_library_type='LOCAL', asset_library_identifier='', relative_asset_identifier='', session_uid=0, use_selected_objects=False)**

Add a procedural operation/effect to the active object

PARAMETERS:

- **asset_library_type** (enum in Asset Library Type Items, (optional)) – Asset Library Type
- **asset_library_identifier** (*string, (optional, never None)*) – Asset Library Identifier
- **relative_asset_identifier** (*string, (optional, never None)*) – Relative Asset Identifier
- **session_uid** (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator
- **use_selected_objects** (*boolean, (optional)*) – Selected Objects, Affect all selected objects instead of just the active object

bpy.ops.object.**modifier_apply(*, modifier='', report=False, merge_customdata=True, single_user=False, all_keyframes=False, use_selected_objects=False)**

Apply modifier and remove from the stack

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **report** (*boolean, (optional)*) – Report, Create a notification after the operation
- **merge_customdata** (*boolean, (optional)*) – Merge UVs, For mesh objects, merge UV coordinates that share a vertex to account for imprecision in some modifiers

- **single_user** (*boolean, (optional)*) – Make Data Single User, Make the object's data single user if needed
- **all_keyframes** (*boolean, (optional)*) – Apply to all keyframes, For Grease Pencil objects, apply the modifier to all the keyframes
- **use_selected_objects** (*boolean, (optional)*) – Selected Objects, Affect all selected objects instead of just the active object

bpy.ops.object.**modifier_apply_as_shapekey(*, keep_modifier=False, modifier='', report=False, use_selected_objects=False)**

Apply modifier as a new shape key and remove from the stack

PARAMETERS:

- **keep_modifier** (*boolean, (optional)*) – Keep Modifier, Do not remove the modifier from stack
- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **report** (*boolean, (optional)*) – Report, Create a notification after the operation
- **use_selected_objects** (*boolean, (optional)*) – Selected Objects, Affect all selected objects instead of just the active object

bpy.ops.object.**modifier_convert(*, modifier='')**

Convert particles to a mesh object

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**modifier_copy(*, modifier='', use_selected_objects=False)**

Duplicate modifier at the same position in the stack

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **use_selected_objects** (*boolean, (optional)*) – Selected Objects, Affect all selected objects instead of just the active object

bpy.ops.object.**modifier_copy_to_selected(*, modifier='')**

Copy the modifier from the active object to all selected objects

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**modifier_move_down(*, modifier='')**

Move modifier down in the stack

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**modifier_move_to_index(*, modifier='', index=0, use_selected_objects=False)**

Change the modifier's index in the stack so it evaluates after the set number of others

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **index** (*int in [0, inf], (optional)*) – Index, The index to move the modifier to
- **use_selected_objects** (*boolean, (optional)*) – Selected Objects, Affect all selected objects instead of just the active object

bpy.ops.object.**modifier_move_up(*, modifier='')**

Move modifier up in the stack

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**modifier_remove(*, modifier='', report=False, use_selected_objects=False)**

Remove a modifier from the active object

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **report** (*boolean, (optional)*) – Report, Create a notification after the operation
- **use_selected_objects** (*boolean, (optional)*) – Selected Objects, Affect all selected objects instead of just the active object

bpy.ops.object.**modifier_set_active(*, modifier='')**

Activate the modifier to use as the context

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**modifiers_clear()**

Clear all modifiers from the selected objects

bpy.ops.object.**modifiers_copy_to_selected()**

Copy modifiers to other selected objects

bpy.ops.object.**move_to_collection(*, collection_index=-1, is_new=False, new_collection_name='')**

Move objects to a collection

PARAMETERS:

- **collection_index** (*int in [-1, inf], (optional)*) – Collection Index, Index of the collection to move to
- **is_new** (*boolean, (optional)*) – New, Move objects to a new collection
- **new_collection_name** (*string, (optional, never None)*) – Name, Name of the newly added collection

bpy.ops.object.**multires_base_apply(*, modifier='')**

Modify the base mesh to conform to the displaced mesh

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**multires_external_pack()**

Pack displacements from an external file

bpy.ops.object.**multires_external_save(*, filepath='', hide_props_region=True, check_existing=True, filter_blender=False, filter_backup=False, filter_image=False, filter_movie=False, filter_python=False, filter_font=False, filter_sound=False, filter_text=False, filter_archive=False, filter_btx=True, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=Fals filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=9, relative_path=True, display_type='DEFAULT', sort_method='', modifier='')**

Save displacements to an external file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **hide_props_region** (*boolean, (optional)*) – Hide Operator Properties, Collapse the region displaying the operator settings
- **check_existing** (*boolean, (optional)*) – Check Existing, Check and warn on overwriting existing files
- **filter_blender** (*boolean, (optional)*) – Filter .blend files
- **filter_backup** (*boolean, (optional)*) – Filter .blend files
- **filter_image** (*boolean, (optional)*) – Filter image files
- **filter_movie** (*boolean, (optional)*) – Filter movie files
- **filter_python** (*boolean, (optional)*) – Filter Python files
- **filter_font** (*boolean, (optional)*) – Filter font files
- **filter_sound** (*boolean, (optional)*) – Filter sound files
- **filter_text** (*boolean, (optional)*) – Filter text files
- **filter_archive** (*boolean, (optional)*) – Filter archive files
- **filter_btx** (*boolean, (optional)*) – Filter btx files

- **filter_collada** (*boolean, (optional)*) – Filter COLLADA files
- **filter_alembic** (*boolean, (optional)*) – Filter Alembic files
- **filter_usd** (*boolean, (optional)*) – Filter USD files
- **filter_obj** (*boolean, (optional)*) – Filter OBJ files
- **filter_volume** (*boolean, (optional)*) – Filter OpenVDB volume files
- **filter_folder** (*boolean, (optional)*) – Filter folders
- **filter_blenlib** (*boolean, (optional)*) – Filter Blender IDs
- **filemode** (*int in [1, 9], (optional)*) – File Browser Mode, The setting for the file browser mode to load a .blend file, a library or a special file
- **relative_path** (*boolean, (optional)*) – Relative Path, Select the file relative to the blend file
- **display_type** (*enum in ['DEFAULT', 'LIST_VERTICAL', 'LIST_HORIZONTAL', 'THUMBNAIL'], (optional)*) – Display Type
  - `DEFAULT` Default – Automatically determine display type for files.
  - `LIST_VERTICAL` Short List – Display files as short list.
  - `LIST_HORIZONTAL` Long List – Display files as a detailed list.
  - `THUMBNAIL` Thumbnails – Display files as thumbnails.
- **sort_method** (*enum in [], (optional)*) – File sorting mode
- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**multires_higher_levels_delete(*, modifier='')**

Deletes the higher resolution mesh, potential loss of detail

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**multires_rebuild_subdiv(*, modifier='')**

Rebuilds all possible subdivisions levels to generate a lower resolution base mesh

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**multires_reshape(*, modifier='')**

Copy vertex coordinates from other object

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**multires_subdivide(*, modifier='', mode='CATMULL_CLARK')**

Add a new level of subdivision

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **mode** (*enum in ['CATMULL_CLARK', 'SIMPLE', 'LINEAR'], (optional)*) – Subdivision Mode, How the mesh is going to be subdivided to create a new level
  - `CATMULL_CLARK` Catmull-Clark – Create a new level using Catmull-Clark subdivisions.
  - `SIMPLE` Simple – Create a new level using simple subdivisions.
  - `LINEAR` Linear – Create a new level using linear interpolation of the sculpted displacement.

bpy.ops.object.**multires_unsubdivide(*, modifier='')**

Rebuild a lower subdivision level of the current base mesh

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit bpy.ops.object.**ocean_bake(*, modifier='', free=False)**

Bake an image sequence of ocean data

PARAMETERS:

- **modifier** (*string, (optional, never None)*) – Modifier, Name of the modifier to edit
- **free** (*boolean, (optional)*) – Free, Free the bake, rather than generating it

bpy.ops.object.**origin_clear()**

Clear the object's origin

bpy.ops.object.**origin_set(*, type='GEOMETRY_ORIGIN', center='MEDIAN')**

Set the object's origin, by either moving the data, or set to center of data, or use 3D cursor

PARAMETERS:

- **type** (*enum in ['GEOMETRY_ORIGIN', 'ORIGIN_GEOMETRY', 'ORIGIN_CURSOR', 'ORIGIN_CENTER_OF_MASS', 'ORIGIN_CENTER_OF_VOLUME'], (optional)*) –

  Type

  - `GEOMETRY_ORIGIN` Geometry to Origin – Move object geometry to object origin.
  - `ORIGIN_GEOMETRY` Origin to Geometry – Calculate the center of geometry based on the current pivot point (median, otherwise bounding box).
  - `ORIGIN_CURSOR` Origin to 3D Cursor – Move object origin to position of the 3D cursor.
  - `ORIGIN_CENTER_OF_MASS` Origin to Center of Mass (Surface) – Calculate the center of mass from the surface area.
  - `ORIGIN_CENTER_OF_VOLUME` Origin to Center of Mass (Volume) – Calculate the center of mass from the volume (must be manifold geometry with consistent normals).

- **center** (*enum in ['MEDIAN', 'BOUNDS'], (optional)*) – Center

bpy.ops.object.**parent_clear(*, type='CLEAR')**

Clear the object's parenting

PARAMETERS:

type (*enum in ['CLEAR', 'CLEAR_KEEP_TRANSFORM', 'CLEAR_INVERSE'], (optional)*) –

Type

- `CLEAR` Clear Parent – Completely clear the parenting relationship, including involved modifiers if any.
- `CLEAR_KEEP_TRANSFORM` Clear and Keep Transformation – As 'Clear Parent', but keep the current visual transformations of the object.
- `CLEAR_INVERSE` Clear Parent Inverse – Reset the transform corrections applied to the parenting relationship, does not remove parentin itself.

bpy.ops.object.**parent_inverse_apply()**

Apply the object's parent inverse to its data

bpy.ops.object.**parent_no_inverse_set(*, keep_transform=False)**

Set the object's parenting without setting the inverse parent correction

PARAMETERS:

keep_transform (*boolean, (optional)*) – Keep Transform, Preserve the world transform throughout parenting

bpy.ops.object.**parent_set(*, type='OBJECT', xmirror=False, keep_transform=False)**

Set the object's parenting

PARAMETERS:

- **type** (*enum in ['OBJECT', 'ARMATURE', 'ARMATURE_NAME', 'ARMATURE_AUTO', 'ARMATURE_ENVELOPE', 'BONE', 'BONE_RELATIVE', 'CURVE', 'FOLLOW', 'PATH_CONST', 'LATTICE', 'VERTEX', 'VERTEX_TRI'], (optional)*) – Type

- **xmirror** (*boolean, (optional)*) – X Mirror, Apply weights symmetrically along X axis, for Envelope/Automatic vertex groups creation
- **keep_transform** (*boolean, (optional)*) – Keep Transform, Apply transformation before parenting

bpy.ops.object.**particle_system_add()**

Add a particle system

bpy.ops.object.**particle_system_remove()**

Remove the selected particle system

bpy.ops.object.**paths_calculate(*, display_type='RANGE', range='SCENE')**

Generate motion paths for the selected objects

PARAMETERS:

- **display_type** (enum in Motionpath Display Type Items, (optional)) – Display type
- **range** (enum in Motionpath Range Items, (optional)) – Computation Range

bpy.ops.object.**paths_clear(*, only_selected=False)**

Undocumented, consider contributing.

PARAMETERS:

only_selected (*boolean, (optional)*) – Only Selected, Only clear motion paths of selected objects

bpy.ops.object.**paths_update()**

Recalculate motion paths for selected objects

bpy.ops.object.**paths_update_visible()**

Recalculate all visible motion paths for objects and poses

bpy.ops.object.**pointcloud_add(*, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Add a point cloud object to the scene

PARAMETERS:

- **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.object.**posemode_toggle()**

Enable or disable posing/selecting bones

bpy.ops.object.**quadriflow_remesh(*, use_mesh_symmetry=True, use_preserve_sharp=False, use_preserve_boundary=False, preserve_attributes=False, smooth_normals=False, mode='FACES', target_ratio=1.0, target_edge_length=0.1, target_faces=4000, mesh_area=-1.0, seed=0)**

Create a new quad based mesh using the surface data of the current mesh. All data layers will be lost

PARAMETERS:

- **use_mesh_symmetry** (*boolean, (optional)*) – Use Mesh Symmetry, Generates a symmetrical mesh using the mesh symmetry configuration
- **use_preserve_sharp** (*boolean, (optional)*) – Preserve Sharp, Try to preserve sharp features on the mesh
- **use_preserve_boundary** (*boolean, (optional)*) – Preserve Mesh Boundary, Try to preserve mesh boundary on the mesh
- **preserve_attributes** (*boolean, (optional)*) – Preserve Attributes, Reproject attributes onto the new mesh

- **smooth_normals** (*boolean, (optional)*) – Smooth Normals, Set the output mesh normals to smooth
- **mode** (*enum in ['RATIO', 'EDGE', 'FACES'], (optional)*) –

  Mode, How to specify the amount of detail for the new mesh

  - `RATIO` Ratio – Specify target number of faces relative to the current mesh.
  - `EDGE` Edge Length – Input target edge length in the new mesh.
  - `FACES` Faces – Input target number of faces in the new mesh.
- **target_ratio** (*float in [0, inf], (optional)*) – Ratio, Relative number of faces compared to the current mesh
- **target_edge_length** (*float in [1e-07, inf], (optional)*) – Edge Length, Target edge length in the new mesh
- **target_faces** (*int in [1, inf], (optional)*) – Number of Faces, Approximate number of faces (quads) in the new mesh
- **mesh_area** (*float in [-inf, inf], (optional)*) – Old Object Face Area, This property is only used to cache the object area for later calculatio
- **seed** (*int in [0, inf], (optional)*) – Seed, Random seed to use with the solver. Different seeds will cause the remesher to come up with differ quad layouts on the mesh

bpy.ops.object.**quick_explode(*, style='EXPLODE', amount=100, frame_duration=50, frame_start=1, frame_end=10, velocity=1.0, fade=True)**

Make selected objects explode

PARAMETERS:

- **style** (*enum in ['EXPLODE', 'BLEND'], (optional)*) – Explode Style
- **amount** (*int in [2, 10000], (optional)*) – Number of Pieces
- **frame_duration** (*int in [1, 300000], (optional)*) – Duration
- **frame_start** (*int in [1, 300000], (optional)*) – Start Frame
- **frame_end** (*int in [1, 300000], (optional)*) – End Frame
- **velocity** (*float in [0, 300000], (optional)*) – Outwards Velocity
- **fade** (*boolean, (optional)*) – Fade, Fade the pieces over time

FILE:

startup/bl_operators/object_quick_effects.py:260

bpy.ops.object.**quick_fur(*, density='MEDIUM', length=0.1, radius=0.001, view_percentage=1.0, apply_hair_guides=True, use_noise=True, use_frizz=True)**

Add a fur setup to the selected objects

PARAMETERS:

- **density** (*enum in ['LOW', 'MEDIUM', 'HIGH'], (optional)*) – Density
- **length** (*float in [0.001, 100], (optional)*) – Length
- **radius** (*float in [0, 10], (optional)*) – Hair Radius
- **view_percentage** (*float in [0, 1], (optional)*) – View Percentage
- **apply_hair_guides** (*boolean, (optional)*) – Apply Hair Guides
- **use_noise** (*boolean, (optional)*) – Noise
- **use_frizz** (*boolean, (optional)*) – Frizz

FILE:

startup/bl_operators/object_quick_effects.py:91

bpy.ops.object.**quick_liquid(*, show_flows=False)**

Make selected objects liquid

PARAMETERS:

show_flows (*boolean, (optional)*) – Render Liquid Objects, Keep the liquid objects visible during rendering

FILE:

startup/bl_operators/object_quick_effects.py:546

bpy.ops.object.**quick_smoke(*, style='SMOKE', show_flows=False)**

Use selected objects as smoke emitters

PARAMETERS:

- **style** (*enum in ['SMOKE', 'FIRE', 'BOTH'], (optional)*) – Smoke Style
- **show_flows** (*boolean, (optional)*) – Render Smoke Objects, Keep the smoke objects visible during rendering

FILE:

startup/bl_operators/object_quick_effects.py:437

bpy.ops.object.**randomize_transform(*, random_seed=0, use_delta=False, use_loc=True, loc=(0.0, 0.0, 0.0), use_rot=True, rot=(0.0, 0.0, 0.0), use_scale=True, scale_even=False, scale=(1.0, 1.0, 1.0))**

Randomize objects location, rotation, and scale

PARAMETERS:

- **random_seed** (*int in [0, 10000], (optional)*) – Random Seed, Seed value for the random generator
- **use_delta** (*boolean, (optional)*) – Transform Delta, Randomize delta transform values instead of regular transform
- **use_loc** (*boolean, (optional)*) – Randomize Location, Randomize the location values
- **loc** (`mathutils.Vector` of 3 items in [-100, 100], (optional)) – Location, Maximum distance the objects can spread over each axis
- **use_rot** (*boolean, (optional)*) – Randomize Rotation, Randomize the rotation values
- **rot** (`mathutils.Euler` rotation of 3 items in [-3.14159, 3.14159], (optional)) – Rotation, Maximum rotation over each axis
- **use_scale** (*boolean, (optional)*) – Randomize Scale, Randomize the scale values
- **scale_even** (*boolean, (optional)*) – Scale Even, Use the same scale value for all axis
- **scale** (*float array of 3 items in [-100, 100], (optional)*) – Scale, Maximum scale randomization over each axis

FILE:

startup/bl_operators/object_randomize_transform.py:161

bpy.ops.object.**reset_override_library()**

Reset the selected local overrides to their linked references values

bpy.ops.object.**rotation_clear(*, clear_delta=False)**

Clear the object's rotation

PARAMETERS:

clear_delta (*boolean, (optional)*) – Clear Delta, Clear delta rotation in addition to clearing the normal rotation transform

bpy.ops.object.**scale_clear(*, clear_delta=False)**

Clear the object's scale

PARAMETERS:

clear_delta (*boolean, (optional)*) – Clear Delta, Clear delta scale in addition to clearing the normal scale transform

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

Change selection of all visible objects in scene

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.object.**select_by_type(*, extend=False, type='MESH')** bpy.ops.object.**select_by_type(*, extend=False, type='MESH')**

Select all visible objects that are of a type

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **type** (enum in Object Type Items, (optional)) – Type

bpy.ops.object.**select_camera(*, extend=False)**

Select the active camera

PARAMETERS:

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

FILE:

startup/bl_operators/object.py:122

bpy.ops.object.**select_grouped(*, extend=False, type='CHILDREN_RECURSIVE')**

Select all visible objects grouped by various properties

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **type** (*enum in ['CHILDREN_RECURSIVE', 'CHILDREN', 'PARENT', 'SIBLINGS', 'TYPE', 'COLLECTION', 'HOOK', 'PASS', 'COLO 'KEYINGSET', 'LIGHT_TYPE'], (optional)*) –

  Type

  - `CHILDREN_RECURSIVE` Children.
  - `CHILDREN` Immediate Children.
  - `PARENT` Parent.
  - `SIBLINGS` Siblings – Shared parent.
  - `TYPE` Type – Shared object type.
  - `COLLECTION` Collection – Shared collection.
  - `HOOK` Hook.
  - `PASS` Pass – Render pass index.
  - `COLOR` Color – Object color.
  - `KEYINGSET` Keying Set – Objects included in active Keying Set.
  - `LIGHT_TYPE` Light Type – Matching light types.

bpy.ops.object.**select_hierarchy(*, direction='PARENT', extend=False)**

Select object relative to the active object's position in the hierarchy

PARAMETERS:

- **direction** (*enum in ['PARENT', 'CHILD'], (optional)*) – Direction, Direction to select in the hierarchy
- **extend** (*boolean, (optional)*) – Extend, Extend the existing selection

FILE:

startup/bl_operators/object.py:172

bpy.ops.object.**select_less()**

Deselect objects at the boundaries of parent/child relationships

bpy.ops.object.**select_linked(*, extend=False, type='OBDATA')**

Select all visible objects that are linked

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **type** (*enum in ['OBDATA', 'MATERIAL', 'DUPGROUP', 'PARTICLE', 'LIBRARY', 'LIBRARY_OBDATA'], (optional)*) – Type

bpy.ops.object.**select_mirror(*, extend=False)**

Select the mirror objects of the selected object e.g. “L.sword” and “R.sword”

PARAMETERS:

extend (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first

bpy.ops.object.**select_more()**

Select connected parent/child objects

bpy.ops.object.**select_pattern(*, pattern='*', case_sensitive=False, extend=True)**

Select objects matching a naming pattern

PARAMETERS:

- **pattern** (*string, (optional, never None)*) – Pattern, Name filter using ‘*’, ‘?’ and ‘[abc]’ unix style wildcards
- **case_sensitive** (*boolean, (optional)*) – Case Sensitive, Do a case sensitive compare
- **extend** (*boolean, (optional)*) – Extend, Extend the existing selection

FILE:

startup/bl_operators/object.py:45

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

Select or deselect random visible objects

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.object.**select_same_collection(*, collection='')**

Select object in the same collection

PARAMETERS:

collection (*string, (optional, never None)*) – Collection, Name of the collection to select

bpy.ops.object.**shade_auto_smooth(*, use_auto_smooth=True, angle=0.523599)**

Add modifier to automatically set the sharpness of mesh edges based on the angle between the neighboring faces

PARAMETERS:

- **use_auto_smooth** (*boolean, (optional)*) – Auto Smooth, Add modifier to set edge sharpness automatically
- **angle** (*float in [0, 3.14159], (optional)*) – Angle, Maximum angle between face normals that will be considered as smooth

bpy.ops.object.**shade_flat(*, keep_sharp_edges=True)**

Render and display faces uniform, using face normals

PARAMETERS:

keep_sharp_edges (*boolean, (optional)*) – Keep Sharp Edges, Don’t remove sharp edges, which are redundant with faces shaded smooth

bpy.ops.object.**shade_smooth(*, keep_sharp_edges=True)**

Render and display faces smooth, using interpolated vertex normals

PARAMETERS:

keep_sharp_edges (*boolean, (optional)*) – Keep Sharp Edges, Don’t remove sharp edges. Tagged edges will remain sharp bpy.ops.object.**shade_smooth_by_angle(*, angle=0.523599, keep_sharp_edges=True)**

Set the sharpness of mesh edges based on the angle between the neighboring faces

PARAMETERS:

- **angle** (*float in [0, 3.14159], (optional)*) – Angle, Maximum angle between face normals that will be considered as smooth
- **keep_sharp_edges** (*boolean, (optional)*) – Keep Sharp Edges, Only add sharp edges instead of clearing existing tags first

bpy.ops.object.**shaderfx_add(*, type='FX_BLUR')**

Add a visual effect to the active object

PARAMETERS:

type (enum in Object Shaderfx Type Items, (optional)) – Type

bpy.ops.object.**shaderfx_copy(*, shaderfx='')**

Duplicate effect at the same position in the stack

PARAMETERS:

shaderfx (*string, (optional, never None)*) – Shader, Name of the shaderfx to edit

bpy.ops.object.**shaderfx_move_down(*, shaderfx='')**

Move effect down in the stack

PARAMETERS:

shaderfx (*string, (optional, never None)*) – Shader, Name of the shaderfx to edit

bpy.ops.object.**shaderfx_move_to_index(*, shaderfx='', index=0)**

Change the effect's position in the list so it evaluates after the set number of others

PARAMETERS:

- **shaderfx** (*string, (optional, never None)*) – Shader, Name of the shaderfx to edit
- **index** (*int in [0, inf], (optional)*) – Index, The index to move the effect to

bpy.ops.object.**shaderfx_move_up(*, shaderfx='')**

Move effect up in the stack

PARAMETERS:

shaderfx (*string, (optional, never None)*) – Shader, Name of the shaderfx to edit

bpy.ops.object.**shaderfx_remove(*, shaderfx='', report=False)**

Remove a effect from the active Grease Pencil object

PARAMETERS:

- **shaderfx** (*string, (optional, never None)*) – Shader, Name of the shaderfx to edit
- **report** (*boolean, (optional)*) – Report, Create a notification after the operation

bpy.ops.object.**shape_key_add(*, from_mix=True)**

Add shape key to the object

PARAMETERS:

from_mix (*boolean, (optional)*) – From Mix, Create the new shape key from the existing mix of keys

bpy.ops.object.**shape_key_clear()**

Reset the weights of all shape keys to 0 or to the closest value respecting the limits

bpy.ops.object.**shape_key_lock(*, action='LOCK')**

Change the lock state of all shape keys of active object

PARAMETERS:

action (*enum in ['LOCK', 'UNLOCK'], (optional)*) –

Action, Lock action to execute on vertex groups

- `LOCK` Lock – Lock all shape keys.
- `UNLOCK` Unlock – Unlock all shape keys.

bpy.ops.object.**shape_key_mirror(*, use_topology=False)**

Mirror the current shape key along the local X axis

PARAMETERS:

use_topology (*boolean, (optional)*) – Topology Mirror, Use topology based mirroring (for when both sides of mesh have matching, unique topology)

bpy.ops.object.**shape_key_move(*, type='TOP')**

Move the active shape key up/down in the list

PARAMETERS:

type (*enum in ['TOP', 'UP', 'DOWN', 'BOTTOM'], (optional)*) –

Type

- `TOP` Top – Top of the list.
- `UP` Up.
- `DOWN` Down.
- `BOTTOM` Bottom – Bottom of the list.

bpy.ops.object.**shape_key_remove(*, all=False, apply_mix=False)**

Remove shape key from the object

PARAMETERS:

- **all** (*boolean, (optional)*) – All, Remove all shape keys
- **apply_mix** (*boolean, (optional)*) – Apply Mix, Apply current mix of shape keys to the geometry before removing them

bpy.ops.object.**shape_key_retime()**

Resets the timing for absolute shape keys

bpy.ops.object.**shape_key_transfer(*, mode='OFFSET', use_clamp=False)**

Copy the active shape key of another selected object to this one

PARAMETERS:

- **mode** (*enum in ['OFFSET', 'RELATIVE_FACE', 'RELATIVE_EDGE'], (optional)*) –

  Transformation Mode, Relative shape positions to the new shape method

  - `OFFSET` Offset – Apply the relative positional offset.
  - `RELATIVE_FACE` Relative Face – Calculate relative position (using faces).
  - `RELATIVE_EDGE` Relative Edge – Calculate relative position (using edges).

- **use_clamp** (*boolean, (optional)*) – Clamp Offset, Clamp the transformation to the distance each vertex moves in the original shape

FILE:

startup/bl_operators/object.py:474

bpy.ops.object.**simulation_nodes_cache_bake(*, selected=False)**

Bake simulations in geometry nodes modifiers

PARAMETERS:

selected (*boolean, (optional)*) – Selected, Bake cache on all selected objects

selected (*boolean, (optional)*) – Selected, Bake cache on all selected objects

bpy.ops.object.**simulation_nodes_cache_calculate_to_frame(*, selected=False)**

Calculate simulations in geometry nodes modifiers from the start to current frame

PARAMETERS:

selected (*boolean, (optional)*) – Selected, Calculate all selected objects instead of just the active object

bpy.ops.object.**simulation_nodes_cache_delete(*, selected=False)**

Delete cached/baked simulations in geometry nodes modifiers

PARAMETERS:

selected (*boolean, (optional)*) – Selected, Delete cache on all selected objects

bpy.ops.object.**skin_armature_create(*, modifier='')**

Create an armature that parallels the skin layout

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**skin_loose_mark_clear(*, action='MARK')**

Mark/clear selected vertices as loose

PARAMETERS:

action (*enum in ['MARK', 'CLEAR'], (optional)*) –

Action

- `MARK` Mark – Mark selected vertices as loose.
- `CLEAR` Clear – Set selected vertices as not loose.

bpy.ops.object.**skin_radii_equalize()**

Make skin radii of selected vertices equal on each axis

bpy.ops.object.**skin_root_mark()**

Mark selected vertices as roots

bpy.ops.object.**speaker_add(*, 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))**

Add a speaker object to the scene

PARAMETERS:

- **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.object.**subdivision_set(*, level=1, relative=False)**

Sets a Subdivision Surface level (1 to 5)

PARAMETERS:

- **level** (*int in [-100, 100], (optional)*) – Level
- **relative** (*boolean, (optional)*) – Relative, Apply the subdivision surface level as an offset relative to the current level

FILE:

startup/bl_operators/object.py:239

bpy.ops.object.**surfacedeform_bind(*, modifier='')**

Bind mesh to target in surface deform modifier

PARAMETERS:

modifier (*string, (optional, never None)*) – Modifier, Name of the modifier to edit

bpy.ops.object.**text_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))**

Add a text object to the scene

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.object.**track_clear(*, type='CLEAR')**

Clear tracking constraint or flag from object

PARAMETERS:

type (*enum in ['CLEAR', 'CLEAR_KEEP_TRANSFORM'], (optional)*) – Type

bpy.ops.object.**track_set(*, type='DAMPTRACK')**

Make the object track another object, using various methods/constraints

PARAMETERS:

type (*enum in ['DAMPTRACK', 'TRACKTO', 'LOCKTRACK'], (optional)*) – Type

bpy.ops.object.**transfer_mode(*, use_flash_on_transfer=True)**

Switches the active object and assigns the same mode to a new one under the mouse cursor, leaving the active mode in the current one

PARAMETERS:

use_flash_on_transfer (*boolean, (optional)*) – Flash On Transfer, Flash the target object when transferring the mode

bpy.ops.object.**transform_apply(*, location=True, rotation=True, scale=True, properties=True, isolate_users=False)**

Apply the object's transformation to its data

PARAMETERS:

- **location** (*boolean, (optional)*) – Location
- **rotation** (*boolean, (optional)*) – Rotation
- **scale** (*boolean, (optional)*) – Scale
- **properties** (*boolean, (optional)*) – Apply Properties, Modify properties such as curve vertex radius, font size and bone envelope
- **isolate_users** (*boolean, (optional)*) – Isolate Multi User Data, Create new object-data users if needed

bpy.ops.object.**transform_axis_target()**

Interactively point cameras and lights to a location (Ctrl translates)

bpy.ops.object.**transform_to_mouse(*, name='', session_uid=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)), drop_x=0, drop_y=0)**

Snap selected item(s) to the mouse location

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Object name to place (uses the active object when this and 'session_uid' are unset)
- **session_uid** (*int in [-inf, inf], (optional)*) – Session UUID, Session UUID of the object to place (uses the active object when this and 'nam are unset)
- **matrix** (`mathutils.Matrix` of 4 * 4 items in [-inf, inf], (optional)) – Matrix
- **drop_x** (*int in [-inf, inf], (optional)*) – Drop X, X-coordinate (screen space) to place the new object under
- **drop_y** (*int in [-inf, inf], (optional)*) – Drop Y, Y-coordinate (screen space) to place the new object under

bpy.ops.object.**transforms_to_deltas(*, mode='ALL', reset_values=True)**

Convert normal object transforms to delta transforms, any existing delta transforms will be included as well

PARAMETERS:

- **mode** (*enum in ['ALL', 'LOC', 'ROT', 'SCALE'], (optional)*) –

  Mode, Which transforms to transfer

  - `ALL` All Transforms – Transfer location, rotation, and scale transforms.
  - `LOC` Location – Transfer location transforms only.
  - `ROT` Rotation – Transfer rotation transforms only.
  - `SCALE` Scale – Transfer scale transforms only.
- **reset_values** (*boolean, (optional)*) – Reset Values, Clear transform values after transferring to deltas

FILE:

startup/bl_operators/object.py:736

bpy.ops.object.**unlink_data()**

Undocumented, consider contributing.

bpy.ops.object.**vertex_group_add()**

Add a new vertex group to the active object

bpy.ops.object.**vertex_group_assign()**

Assign the selected vertices to the active vertex group

bpy.ops.object.**vertex_group_assign_new()**

Assign the selected vertices to a new vertex group

bpy.ops.object.**vertex_group_clean(*, group_select_mode='', limit=0.0, keep_single=False)**

Remove vertex group assignments which are not required

PARAMETERS:

- **group_select_mode** (*enum in [], (optional)*) – Subset, Define which subset of groups shall be used
- **limit** (*float in [0, 1], (optional)*) – Limit, Remove vertices which weight is below or equal to this limit
- **keep_single** (*boolean, (optional)*) – Keep Single, Keep verts assigned to at least one group when cleaning

bpy.ops.object.**vertex_group_copy()**

Make a copy of the active vertex group

bpy.ops.object.**vertex_group_copy_to_selected()**

Replace vertex groups of selected objects by vertex groups of active object

bpy.ops.object.**vertex_group_deselect()**

Deselect all selected vertices assigned to the active vertex group

bpy.ops.object.**vertex_group_invert(*, group_select_mode='', auto_assign=True, auto_remove=True)**

Invert active vertex group's weights

PARAMETERS:

- **group_select_mode** (*enum in [], (optional)*) – Subset, Define which subset of groups shall be used
- **auto_assign** (*boolean, (optional)*) – Add Weights, Add vertices from groups that have zero weight before inverting
- **auto_remove** (*boolean, (optional)*) – Remove Weights, Remove vertices from groups that have zero weight after inverting

bpy.ops.object.**vertex_group_levels(*, group_select_mode='', offset=0.0, gain=1.0)**

Add some offset and multiply with some gain the weights of the active vertex group

PARAMETERS:

- **group_select_mode** (*enum in [], (optional)*) – Subset, Define which subset of groups shall be used
- **offset** (*float in [-1, 1], (optional)*) – Offset, Value to add to weights
- **gain** (*float in [0, inf], (optional)*) – Gain, Value to multiply weights by

bpy.ops.object.**vertex_group_limit_total(*, group_select_mode='', limit=4)**

Limit deform weights associated with a vertex to a specified number by removing lowest weights

PARAMETERS:

- **group_select_mode** (*enum in [], (optional)*) – Subset, Define which subset of groups shall be used
- **limit** (*int in [1, 32], (optional)*) – Limit, Maximum number of deform weights

bpy.ops.object.**vertex_group_lock(*, action='TOGGLE', mask='ALL')**

Change the lock state of all or some vertex groups of active object

PARAMETERS:

- **action** (*enum in ['TOGGLE', 'LOCK', 'UNLOCK', 'INVERT'], (optional)*) –
  Action, Lock action to execute on vertex groups
  - `TOGGLE` Toggle – Unlock all vertex groups if there is at least one locked group, lock all in other case.
  - `LOCK` Lock – Lock all vertex groups.
  - `UNLOCK` Unlock – Unlock all vertex groups.
  - `INVERT` Invert – Invert the lock state of all vertex groups.
- **mask** (*enum in ['ALL', 'SELECTED', 'UNSELECTED', 'INVERT_UNSELECTED'], (optional)*) –
  Mask, Apply the action based on vertex group selection
  - `ALL` All – Apply action to all vertex groups.
  - `SELECTED` Selected – Apply to selected vertex groups.
  - `UNSELECTED` Unselected – Apply to unselected vertex groups.
  - `INVERT_UNSELECTED` Invert Unselected – Apply the opposite of Lock/Unlock to unselected vertex groups.

bpy.ops.object.**vertex_group_mirror(*, mirror_weights=True, flip_group_names=True, all_groups=False, use_topology=False)**

Mirror vertex group, flip weights and/or names, editing only selected vertices, flipping when both sides are selected otherwise copy from unselectec

PARAMETERS:

- **mirror_weights** (*boolean, (optional)*) – Mirror Weights, Mirror weights
- **flip_group_names** (*boolean, (optional)*) – Flip Group Names, Flip vertex group names
- **all_groups** (*boolean, (optional)*) – All Groups, Mirror all vertex groups weights

[illegible]

- **use_topology** (*boolean, (optional)*) – Topology Mirror, Use topology based mirroring (for when both sides of mesh have matching, unique topology)

bpy.ops.object.**vertex_group_move(*, direction='UP')**

Move the active vertex group up/down in the list

PARAMETERS:

direction (*enum in ['UP', 'DOWN'], (optional)*) – Direction, Direction to move the active vertex group towards

bpy.ops.object.**vertex_group_normalize()**

Normalize weights of the active vertex group, so that the highest ones are now 1.0

bpy.ops.object.**vertex_group_normalize_all(*, group_select_mode='', lock_active=True)**

Normalize all weights of all vertex groups, so that for each vertex, the sum of all weights is 1.0

PARAMETERS:

- **group_select_mode** (*enum in [], (optional)*) – Subset, Define which subset of groups shall be used
- **lock_active** (*boolean, (optional)*) – Lock Active, Keep the values of the active group while normalizing others

bpy.ops.object.**vertex_group_quantize(*, group_select_mode='', steps=4)**

Set weights to a fixed number of steps

PARAMETERS:

- **group_select_mode** (*enum in [], (optional)*) – Subset, Define which subset of groups shall be used
- **steps** (*int in [1, 1000], (optional)*) – Steps, Number of steps between 0 and 1

bpy.ops.object.**vertex_group_remove(*, all=False, all_unlocked=False)**

Delete the active or all vertex groups from the active object

PARAMETERS:

- **all** (*boolean, (optional)*) – All, Remove all vertex groups
- **all_unlocked** (*boolean, (optional)*) – All Unlocked, Remove all unlocked vertex groups

bpy.ops.object.**vertex_group_remove_from(*, use_all_groups=False, use_all_verts=False)**

Remove the selected vertices from active or all vertex group(s)

PARAMETERS:

- **use_all_groups** (*boolean, (optional)*) – All Groups, Remove from all groups
- **use_all_verts** (*boolean, (optional)*) – All Vertices, Clear the active group

bpy.ops.object.**vertex_group_select()**

Select all the vertices assigned to the active vertex group

bpy.ops.object.**vertex_group_set_active(*, group='')**

Set the active vertex group

PARAMETERS:

group (*enum in [], (optional)*) – Group, Vertex group to set as active

bpy.ops.object.**vertex_group_smooth(*, group_select_mode='', factor=0.5, repeat=1, expand=0.0)**

Smooth weights for selected vertices

PARAMETERS:

- **group_select_mode** (*enum in [], (optional)*) – Subset, Define which subset of groups shall be used
- **factor** (*float in [0, 1], (optional)*) – Factor
- **repeat** (*int in [1, 10000], (optional)*) – Iterations

- **repeat** (*int in [1, 10000], (optional)*) – Iterations
- **expand** (*float in [-1, 1], (optional)*) – Expand/Contract, Expand/contract weights

bpy.ops.object.**vertex_group_sort(*, sort_type='NAME')**

Sort vertex groups

PARAMETERS:

sort_type (*enum in ['NAME', 'BONE_HIERARCHY'], (optional)*) – Sort Type, Sort type

bpy.ops.object.**vertex_parent_set()**

Parent selected objects to the selected vertices

bpy.ops.object.**vertex_weight_copy()**

Copy weights from active to selected

bpy.ops.object.**vertex_weight_delete(*, weight_group=-1)**

Delete this weight from the vertex (disabled if vertex group is locked)

PARAMETERS:

weight_group (*int in [-1, inf], (optional)*) – Weight Index, Index of source weight in active vertex group

bpy.ops.object.**vertex_weight_normalize_active_vertex()**

Normalize active vertex's weights

bpy.ops.object.**vertex_weight_paste(*, weight_group=-1)**

Copy this group's weight to other selected vertices (disabled if vertex group is locked)

PARAMETERS:

weight_group (*int in [-1, inf], (optional)*) – Weight Index, Index of source weight in active vertex group

bpy.ops.object.**vertex_weight_set_active(*, weight_group=-1)**

Set as active vertex group

PARAMETERS:

weight_group (*int in [-1, inf], (optional)*) – Weight Index, Index of source weight in active vertex group

bpy.ops.object.**visual_transform_apply()**

Apply the object's visual transformation to its data

bpy.ops.object.**volume_add(*, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Add a volume object to the scene

PARAMETERS:

- **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.object.**volume_import(*, filepath='', directory='', files=None, hide_props_region=True, check_existing=False, filter_blender=Fals filter_backup=False, filter_image=False, filter_movie=False, filter_python=False, filter_font=False, filter_sound=False, filter_text=False, filter_archive=False, filter_btx=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=True, filter_folder=True, filter_blenlib=False, filemode=9, relative_path=True,**

filter_obj=False, filter_volume=True, filter_folder=True, filter_blenlib=False, filemode=9, relative_path=True, display_type='DEFAULT', sort_method='', use_sequence_detection=True, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))

Import OpenVDB volume file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **directory** (*string, (optional, never None)*) – Directory, Directory of the file
- **files** (`bpy_prop_collection` of `OperatorFileListElement`, (optional)) – Files
- **hide_props_region** (*boolean, (optional)*) – Hide Operator Properties, Collapse the region displaying the operator settings
- **check_existing** (*boolean, (optional)*) – Check Existing, Check and warn on overwriting existing files
- **filter_blender** (*boolean, (optional)*) – Filter .blend files
- **filter_backup** (*boolean, (optional)*) – Filter .blend files
- **filter_image** (*boolean, (optional)*) – Filter image files
- **filter_movie** (*boolean, (optional)*) – Filter movie files
- **filter_python** (*boolean, (optional)*) – Filter Python files
- **filter_font** (*boolean, (optional)*) – Filter font files
- **filter_sound** (*boolean, (optional)*) – Filter sound files
- **filter_text** (*boolean, (optional)*) – Filter text files
- **filter_archive** (*boolean, (optional)*) – Filter archive files
- **filter_btx** (*boolean, (optional)*) – Filter btx files
- **filter_collada** (*boolean, (optional)*) – Filter COLLADA files
- **filter_alembic** (*boolean, (optional)*) – Filter Alembic files
- **filter_usd** (*boolean, (optional)*) – Filter USD files
- **filter_obj** (*boolean, (optional)*) – Filter OBJ files
- **filter_volume** (*boolean, (optional)*) – Filter OpenVDB volume files
- **filter_folder** (*boolean, (optional)*) – Filter folders
- **filter_blenlib** (*boolean, (optional)*) – Filter Blender IDs
- **filemode** (*int in [1, 9], (optional)*) – File Browser Mode, The setting for the file browser mode to load a .blend file, a library or a special fil
- **relative_path** (*boolean, (optional)*) – Relative Path, Select the file relative to the blend file
- **display_type** (*enum in ['DEFAULT', 'LIST_VERTICAL', 'LIST_HORIZONTAL', 'THUMBNAIL'], (optional)*) –
  Display Type
  - `DEFAULT` Default – Automatically determine display type for files.
  - `LIST_VERTICAL` Short List – Display files as short list.
  - `LIST_HORIZONTAL` Long List – Display files as a detailed list.
  - `THUMBNAIL` Thumbnails – Display files as thumbnails.
- **sort_method** (*enum in [], (optional)*) – File sorting mode
- **use_sequence_detection** (*boolean, (optional)*) – Detect Sequences, Automatically detect animated sequences in selected volume files (bas on file names)
- **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.object.**voxel_remesh()**

Calculates a new manifold mesh based on the volume of the current mesh. All data layers will be lost

bpy.ops.object.**voxel_size_edit()**

Modify the mesh voxel size interactively used in the voxel remesher