Table of Contents

Skip to content

Audio System (aud)

Audaspace (pronounced “outer space”) is a high level audio library.

Basic Sound Playback

This script shows how to use the classes: `Device`, `Sound` and `Handle`.

```
import aud

device = aud.Device()
# load sound file (it can be a video file with audio)
sound = aud.Sound('music.ogg')

# play the audio, this return a handle to control play/pause
handle = device.play(sound)
# if the audio is not too big and will be used often you can buffer it
sound_buffered = aud.Sound.cache(sound)
handle_buffered = device.play(sound_buffered)

# stop the sounds (otherwise they play until their ends)
handle.stop()
handle_buffered.stop()
```

aud.**AP_LOCATION**

Constant value 3

aud.**AP_ORIENTATION**

Constant value 4

aud.**AP_PANNING**

Constant value 1

aud.**AP_PITCH**

Constant value 2

aud.**AP_VOLUME**

Constant value 0

aud.**CHANNELS_INVALID**

Constant value 0

aud.**CHANNELS_MONO**

Constant value 1

aud.**CHANNELS_STEREO**

Constant value 2

aud.**CHANNELS_STEREO_LFE**

Constant value 3

aud.**CHANNELS_SURROUND4**

Constant value 4

aud.**CHANNELS_SURROUND5**

Constant value 5

aud.**CHANNELS_SURROUND51**

Constant value 6

aud.**CHANNELS_SURROUND61**

Constant value 7

aud.**CHANNELS_SURROUND71**

Constant value 8

aud.**CODEC_AAC**

Constant value 1

aud.**CODEC_AC3**

Constant value 2

aud.**CODEC_FLAC**

Constant value 3

aud.**CODEC_INVALID**

Constant value 0

aud.**CODEC_MP2**

Constant value 4

aud.**CODEC_MP3**

Constant value 5

aud.**CODEC_OPUS**

Constant value 8

aud.**CODEC_PCM**

Constant value 6

aud.**CODEC_VORBIS**

Constant value 7

aud.**CONTAINER_AAC**

Constant value 8

aud.**CONTAINER_AC3**

Constant value 1

aud.**CONTAINER_FLAC**

Constant value 2

aud.**CONTAINER_INVALID**

Constant value 0

aud.**CONTAINER_MATROSKA**

Constant value 3

aud.**CONTAINER_MP2**

Constant value 4

aud.**CONTAINER_MP3**

Constant value 5

aud.**CONTAINER_OGG**

Constant value 6

aud.**CONTAINER_WAV**

Constant value 7

aud.**DISTANCE_MODEL_EXPONENT**

Constant value 5

aud.**DISTANCE_MODEL_EXPONENT_CLAMPED**

Constant value 6

aud.**DISTANCE_MODEL_INVALID**

Constant value 0

aud.**DISTANCE_MODEL_INVERSE**

Constant value 1

aud.**DISTANCE_MODEL_INVERSE_CLAMPED**

Constant value 2

aud.**DISTANCE_MODEL_LINEAR**

Constant value 3

aud.**DISTANCE_MODEL_LINEAR_CLAMPED**

Constant value 4

aud.**FORMAT_FLOAT32**

Constant value 36

aud.**FORMAT_FLOAT64**

Constant value 40

aud.**FORMAT_INVALID**

Constant value 0

aud.**FORMAT_S16**

Constant value 18

aud.**FORMAT_S24**

Constant value 19

aud.**FORMAT_S32**

Constant value 20

aud.**FORMAT_U8**

Constant value 1

aud.**RATE_11025**

Constant value 11025

aud.**RATE_16000**

Constant value 16000

aud.**RATE_192000**

Constant value 192000

aud.**RATE_22050**

Constant value 22050

aud.**RATE_32000**

Constant value 32000

aud.**RATE_44100**

Constant value 44100

aud.**RATE_48000**

Constant value 48000

aud.**RATE_8000**

Constant value 8000

aud.**RATE_88200**

Constant value 88200

aud.**RATE_96000**

Constant value 96000

aud.**RATE_INVALID**

Constant value 0

aud.**STATUS_INVALID**

Constant value 0

aud.**STATUS_PAUSED**

Constant value 2

aud.**STATUS_PLAYING**

Constant value 1

aud.**STATUS_STOPPED**

Constant value 3

class aud.**Device**

Device objects represent an audio output backend like OpenAL or SDL, but might also represent a file output or RAM buffer output.

lock()

Locks the device so that it's guaranteed, that no samples are read from the streams until `unlock()` is called. This is useful if you want to d
start/stop/pause/resume some sounds at the same time.

Note

The device has to be unlocked as often as locked to be able to continue playback.

Warning

Make sure the time between locking and unlocking is as short as possible to avoid clicks.

play(sound, keep=False)

Plays a sound.

PARAMETERS:

- **sound** (`Sound`) – The sound to play.
- **keep** (*bool*) – See `Handle.keep` .

RETURNS:

The playback handle with which playback can be controlled with.

RETURN TYPE:

`Handle`

stopAll()

Stops all playing and paused sounds.

unlock()

Unlocks the device after a lock call, see `lock()` for details.

channels

The channel count of the device.

distance_model

The distance model of the device.

See also

OpenAL Documentation

doppler_factor

The doppler factor of the device. This factor is a scaling factor for the velocity vectors in doppler calculation. So a value bigger than 1 will exaggerate the effect as it raises the velocity.

format

The native sample format of the device.

listener_location

The listeners's location in 3D space, a 3D tuple of floats.

listener_orientation

The listener's orientation in 3D space as quaternion, a 4 float tuple.

listener_velocity

The listener's velocity in 3D space, a 3D tuple of floats.

rate

The sampling rate of the device in Hz.

speed_of_sound

The speed of sound of the device. The speed of sound in air is typically 343.3 m/s.

volume

The overall volume of the device.

class aud.**DynamicMusic**

The DynamicMusic object allows to play music depending on a current scene, scene changes are managed by the class, with the possibility of custo transitions. The default transition is a crossfade effect, and the default scene is silent and has id 0

addScene(scene)

Adds a new scene.

PARAMETERS:

scene (`Sound`) – The scene sound.

RETURNS:

The new scene id.

RETURN TYPE:

int

addTransition(ini, end, transition)

Adds a new scene.

PARAMETERS:

- **ini** (*int*) – the initial scene foor the transition.
- **end** (*int*) – The final scene for the transition.
- **transition** (`Sound`) – The transition sound.

RETURNS:

false if the ini or end scenes don't exist, true othrwise.

RETURN TYPE:

bool

pause()

Pauses playback of the scene.

RETURNS:

Whether the action succeeded.

RETURN TYPE:

bool

resume()

Resumes playback of the scene.

RETURNS:

Whether the action succeeded.

RETURN TYPE:

bool

stop()

Stops playback of the scene.

RETURNS:

Whether the action succeeded.

RETURN TYPE:

bool

fadeTime

The length in seconds of the crossfade transition

position

The playback position of the scene in seconds.

scene

The current scene

status

Whether the scene is playing, paused or stopped (=invalid).

volume

The volume of the scene.

class aud.**HRTF**

An HRTF object represents a set of head related transfer functions as impulse responses. It's used for binaural sound

loadLeftHrtfSet(extension, directory)

Loads all HRTFs from a directory.

PARAMETERS:

- **extension** (*string*) – The file extension of the hrtfs.
- **directory** – The path to where the HRTF files are located.

RETURNS:

The loaded `HRTF` object.

RETURN TYPE:

`HRTF`

loadRightHrtfSet(extension, directory)

Loads all HRTFs from a directory.

PARAMETERS:

- **extension** (*string*) – The file extension of the hrtfs.
- **directory** – The path to where the HRTF files are located.

RETURNS:

The loaded `HRTF` object.

RETURN TYPE:

`HRTF`

addImpulseResponseFromSound(sound, azimuth, elevation)

Adds a new hrtf to the HRTF object

PARAMETERS:

- **sound** (`Sound`) – The sound that contains the hrtf.
- **azimuth** (*float*) – The azimuth angle of the hrtf.
- **elevation** (*float*) – The elevation angle of the hrtf.

RETURNS:

Whether the action succeeded.

RETURN TYPE:

bool

class aud.**Handle**

Handle objects are playback handles that can be used to control playback of a sound. If a sound is played back multiple times then there are as ma handles.

pause()

Pauses playback.

RETURNS:
Whether the action succeeded.

RETURN TYPE:
bool

resume()

Resumes playback.

RETURNS:
Whether the action succeeded.

RETURN TYPE:
bool

stop()

Stops playback.

RETURNS:
Whether the action succeeded.

RETURN TYPE:
bool

Note

This makes the handle invalid.

attenuation

This factor is used for distance based attenuation of the source.

See also

`Device.distance_model`

cone_angle_inner

The opening angle of the inner cone of the source. If the cone values of a source are set there are two (audible) cones with the apex at the `location` of the source and with infinite height, heading in the direction of the source's `orientation`. In the inner cone the volume i normal. Outside the outer cone the volume will be `cone_volume_outer` and in the area between the volume will be interpolated linear

cone_angle_outer

The opening angle of the outer cone of the source.

See also

`cone_angle_inner`

cone_volume_outer

The volume outside the outer cone of the source.

See also

`cone_angle_inner`

distance_maximum

The maximum distance of the source. If the listener is further away the source volume will be 0.

> See also
>
> `Device.distance_model`

distance_reference

The reference distance of the source. At this distance the volume will be exactly `volume`.

> See also
>
> `Device.distance_model`

keep

Whether the sound should be kept paused in the device when its end is reached. This can be used to seek the sound to some position and sta playback again.

> Warning
>
> If this is set to true and you forget stopping this equals a memory leak as the handle exists until the device is destroyed.

location

The source's location in 3D space, a 3D tuple of floats.

loop_count

The (remaining) loop count of the sound. A negative value indicates infinity.

orientation

The source's orientation in 3D space as quaternion, a 4 float tuple.

pitch

The pitch of the sound.

position

The playback position of the sound in seconds.

relative

Whether the source's location, velocity and orientation is relative or absolute to the listener.

status

Whether the sound is playing, paused or stopped (=invalid).

velocity

The source's velocity in 3D space, a 3D tuple of floats.

volume

The volume of the sound.

volume_maximum

The maximum volume of the source.

> See also
>
> `Device.distance_model`

volume_minimum

The minimum volume of the source.

> See also
>
> `Device.distance_model`

class aud.**ImpulseResponse**

An ImpulseResponse object represents a filter with which to convolve a sound.

class aud.**PlaybackManager**

A PlabackManager object allows to easily control groups os sounds organized in categories.

addCategory(volume)

Adds a category with a custom volume.

PARAMETERS:

volume (*float*) – The volume for ther new category.

RETURNS:

The key of the new category.

RETURN TYPE:

int

clean()

Cleans all the invalid and finished sound from the playback manager.

getVolume(catKey)

Retrieves the volume of a category.

PARAMETERS:

catKey (*int*) – the key of the category.

RETURNS:

The volume of the cateogry.

RETURN TYPE:

float

pause(catKey)

Pauses playback of the category.

PARAMETERS:

catKey (*int*) – the key of the category.

RETURNS:

Whether the action succeeded.

RETURN TYPE:

bool

play(sound, catKey)

Plays a sound through the playback manager and assigns it to a category.

PARAMETERS:

- **sound** (`Sound`) – The sound to play.
- **catKey** (*int*) – the key of the category in which the sound will be added, if it doesn't exist, a new one will be created.

RETURNS:

The playback handle with which playback can be controlled with.

The playback handle with which playback can be controlled with.

RETURN TYPE:

`Handle`

resume(catKey)

Resumes playback of the catgory.

PARAMETERS:

catKey (*int*) – the key of the category.

RETURNS:

Whether the action succeeded.

RETURN TYPE:

bool

setVolume(volume, catKey)

Changes the volume of a category.

PARAMETERS:

- **volume** (*float*) – the new volume value.
- **catKey** (*int*) – the key of the category.

RETURNS:

Whether the action succeeded.

RETURN TYPE:

int

stop(catKey)

Stops playback of the category.

PARAMETERS:

catKey (*int*) – the key of the category.

RETURNS:

Whether the action succeeded.

RETURN TYPE:

bool

class aud.**Sequence**

This sound represents sequenced entries to play a sound sequence.

add()

Adds a new entry to the sequence.

PARAMETERS:

- **sound** (`Sound`) – The sound this entry should play.
- **begin** (*double*) – The start time.
- **end** (*double*) – The end time or a negative value if determined by the sound.
- **skip** (*double*) – How much seconds should be skipped at the beginning.

RETURNS:

The entry added.

RETURN TYPE:

`SequenceEntry`

remove()

Removes an entry from the sequence.

PARAMETERS:

entry (`SequenceEntry`) – The entry to remove.

setAnimationData()

Writes animation data to a sequence.

PARAMETERS:

- **type** (*int*) – The type of animation data.
- **frame** (*int*) – The frame this data is for.
- **data** (*sequence of float*) – The data to write.
- **animated** (*bool*) – Whether the attribute is animated.

channels

The channel count of the sequence.

distance_model

The distance model of the sequence.

See also

OpenAL Documentation

doppler_factor

The doppler factor of the sequence. This factor is a scaling factor for the velocity vectors in doppler calculation. So a value bigger than 1 will exaggerate the effect as it raises the velocity.

fps

The listeners's location in 3D space, a 3D tuple of floats.

muted

Whether the whole sequence is muted.

rate

The sampling rate of the sequence in Hz.

speed_of_sound

The speed of sound of the sequence. The speed of sound in air is typically 343.3 m/s.

class aud.**SequenceEntry**

SequenceEntry objects represent an entry of a sequenced sound.

move()

Moves the entry.

PARAMETERS:

- **begin** (*double*) – The new start time.
- **end** (*double*) – The new end time or a negative value if unknown.
- **skip** (*double*) – How many seconds to skip at the beginning.

setAnimationData()

Writes animation data to a sequenced entry.

PARAMETERS:

- **type** (*int*) – The type of animation data.
- **frame** (*int*) – The frame this data is for.
- **data** (*sequence of float*) – The data to write.
- **animated** (*bool*) – Whether the attribute is animated.

attenuation

This factor is used for distance based attenuation of the source.

> See also
>
> `Device.distance_model`

cone_angle_inner

The opening angle of the inner cone of the source. If the cone values of a source are set there are two (audible) cones with the apex at the `location` of the source and with infinite height, heading in the direction of the source's `orientation`. In the inner cone the volume i normal. Outside the outer cone the volume will be `cone_volume_outer` and in the area between the volume will be interpolated linear

cone_angle_outer

The opening angle of the outer cone of the source.

> See also
>
> `cone_angle_inner`

cone_volume_outer

The volume outside the outer cone of the source.

> See also
>
> `cone_angle_inner`

distance_maximum

The maximum distance of the source. If the listener is further away the source volume will be 0.

> See also
>
> `Device.distance_model`

distance_reference

The reference distance of the source. At this distance the volume will be exactly `volume`.

> See also
>
> `Device.distance_model`

muted

Whether the entry is muted.

relative

Whether the source's location, velocity and orientation is relative or absolute to the listener.

sound

The sound the entry is representing and will be played in the sequence.

volume_maximum

The maximum volume of the source.

See also

`Device.distance_model`

volume_minimum

The minimum volume of the source.

See also

`Device.distance_model`

class aud.**Sound**

Sound objects are immutable and represent a sound that can be played simultaneously multiple times. They are called factories because they create reader objects internally that are used for playback.

classmethod buffer(data, rate)

Creates a sound from a data buffer.

PARAMETERS:

- **data** (`numpy.ndarray`) – The data as two dimensional numpy array.
- **rate** (*double*) – The sample rate.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

classmethod file(filename)

Creates a sound object of a sound file.

PARAMETERS:

filename (*string*) – Path of the file.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

Warning

If the file doesn't exist or can't be read you will not get an exception immediately, but when you try to start playback of that sound.

classmethod list()

Creates an empty sound list that can contain several sounds.

PARAMETERS:

random (*int*) – whether the playback will be random or not.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

classmethod sawtooth(frequency, rate=48000)

Creates a sawtooth sound which plays a sawtooth wave.

PARAMETERS:

- **frequency** (*float*) – The frequency of the sawtooth wave in Hz.
- **rate** (*int*) – The sampling rate in Hz. It's recommended to set this value to the playback device's samling rate to avoid resamping.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

classmethod silence(rate=48000)

Creates a silence sound which plays simple silence.

PARAMETERS:

rate (*int*) – The sampling rate in Hz. It's recommended to set this value to the playback device's samling rate to avoid resamping.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

classmethod sine(frequency, rate=48000)

Creates a sine sound which plays a sine wave.

PARAMETERS:

- **frequency** (*float*) – The frequency of the sine wave in Hz.
- **rate** (*int*) – The sampling rate in Hz. It's recommended to set this value to the playback device's samling rate to avoid resamping.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

classmethod square(frequency, rate=48000)

Creates a square sound which plays a square wave.

PARAMETERS:

- **frequency** (*float*) – The frequency of the square wave in Hz.
- **rate** (*int*) – The sampling rate in Hz. It's recommended to set this value to the playback device's samling rate to avoid resamping.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

classmethod triangle(frequency, rate=48000)

Creates a triangle sound which plays a triangle wave.

PARAMETERS:

- **frequency** (*float*) – The frequency of the triangle wave in Hz.
- **rate** (*int*) – The sampling rate in Hz. It's recommended to set this value to the playback device's samling rate to avoid resamping.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

ADSR(attack, decay, sustain, release)

ADSR(attack, decay, sustain, release)

Attack-Decay-Sustain-Release envelopes the volume of a sound. Note: there is currently no way to trigger the release with this API.

PARAMETERS:

- **attack** (*float*) – The attack time in seconds.
- **decay** (*float*) – The decay time in seconds.
- **sustain** (*float*) – The sustain level.
- **release** (*float*) – The release level.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

accumulate(additive=False)

Accumulates a sound by summing over positive input differences thus generating a monotonic sigal. If additivity is set to true negative input differences get added too, but positive ones with a factor of two.

Note that with additivity the signal is not monotonic anymore.

PARAMETERS:

additive – Whether the accumulation should be additive or not.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

addSound(sound)

Adds a new sound to a sound list.

PARAMETERS:

sound (`Sound`) – The sound that will be added to the list.

Note

You can only add a sound to a sound list.

binaural()

Creates a binaural sound using another sound as source. The original sound must be mono

PARAMETERS:

- **hrtfs** – An HRTF set.
- **source** (`Source`) – An object representing the source position of the sound.
- **threadPool** (`ThreadPool`) – A thread pool used to parallelize convolution.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

cache()

Caches a sound into RAM.

This saves CPU usage needed for decoding and file access if the underlying sound reads from a file on the harddisk, but it consumes a lot of memory.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

Note

Only known-length factories can be buffered.

Warning

Raw PCM data needs a lot of space, only buffer short factories.

convolver()

Creates a sound that will apply convolution to another sound.

PARAMETERS:

- **impulseResponse** (`ImpulseResponse`) – The filter with which convolve the sound.
- **threadPool** (`ThreadPool`) – A thread pool used to parallelize convolution.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

data()

Retrieves the data of the sound as numpy array.

RETURNS:

A two dimensional numpy float array.

RETURN TYPE:

`numpy.ndarray`

Note

Best efficiency with cached sounds.

delay(time)

Delays by playing adding silence in front of the other sound's data.

PARAMETERS:

time (*float*) – How many seconds of silence should be added before the sound.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

envelope(attack, release, threshold, arthreshold)

Delays by playing adding silence in front of the other sound's data.

PARAMETERS:

- **attack** (*float*) – The attack factor.
- **release** (*float*) – The release factor.
- **threshold** (*float*) – The general threshold value.
- **arthreshold** (*float*) – The attack/release threshold value.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

fadein(start, length)

Fades a sound in by raising the volume linearly in the given time interval.

PARAMETERS:

- **start** (*float*) – Time in seconds when the fading should start.
- **length** (*float*) – Time in seconds how long the fading should last.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

Note

Before the fade starts it plays silence.

fadeout(start, length)

Fades a sound in by lowering the volume linearly in the given time interval.

PARAMETERS:

- **start** (*float*) – Time in seconds when the fading should start.
- **length** (*float*) – Time in seconds how long the fading should last.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

Note

After the fade this sound plays silence, so that the length of the sound is not altered.

filter(b, a=(1,))

Filters a sound with the supplied IIR filter coefficients. Without the second parameter you'll get a FIR filter.

If the first value of the a sequence is 0, it will be set to 1 automatically. If the first value of the a sequence is neither 0 nor 1, all filter coefficients will be scaled by this value so that it is 1 in the end, you don't have to scale yourself.

PARAMETERS:

- **b** (*sequence of float*) – The nominator filter coefficients.
- **a** (*sequence of float*) – The denominator filter coefficients.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

highpass(frequency, Q=0.5)

Creates a second order highpass filter based on the transfer function \(H(s) = s^2 / (s^2 + s/Q + 1)\)

PARAMETERS:

- **frequency** (*float*) – The cut off trequency of the highpass.
- **Q** (*float*) – Q factor of the lowpass.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

join(sound)

Plays two factories in sequence.

PARAMETERS:

sound (`Sound`) – The sound to play second.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

> Note
>
> The two factories have to have the same specifications (channels and samplerate).

limit(start, end)

Limits a sound within a specific start and end time.

PARAMETERS:

- **start** (*float*) – Start time in seconds.
- **end** (*float*) – End time in seconds.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

loop(count)

Loops a sound.

PARAMETERS:

count (*integer*) – How often the sound should be looped. Negative values mean endlessly.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

> Note
>
> This is a filter function, you might consider using `Handle.loop_count` instead.

lowpass(frequency, Q=0.5)

Creates a second order lowpass filter based on the transfer function \(H(s) = 1 / (s^2 + s/Q + 1)\)

PARAMETERS:

- **frequency** (*float*) – The cut off trequency of the lowpass.
- **Q** (*float*) – Q factor of the lowpass.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

mix(sound)

Mixes two factories.

PARAMETERS:

sound (`Sound`) – The sound to mix over the other.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

> Note
>
> The two factories have to have the same specifications (channels and samplerate).

modulate(sound)

Modulates two factories.

PARAMETERS:

sound (`Sound`) – The sound to modulate over the other.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

> Note
>
> The two factories have to have the same specifications (channels and samplerate).

mutable()

Creates a sound that will be restarted when sought backwards. If the original sound is a sound list, the playing sound can change.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

pingpong()

Plays a sound forward and then backward. This is like joining a sound with its reverse.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

pitch(factor)

Changes the pitch of a sound with a specific factor.

PARAMETERS:

factor (*float*) – The factor to change the pitch with.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

Note

This is done by changing the sample rate of the underlying sound, which has to be an integer, so the factor value rounded and the factor may not be 100 % accurate.

Note

This is a filter function, you might consider using `Handle.pitch` instead.

rechannel(channels)

Rechannels the sound.

PARAMETERS:

channels (*int*) – The new channel configuration.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

resample(rate, quality)

Resamples the sound.

PARAMETERS:

- **rate** (*double*) – The new sample rate.
- **quality** (*int*) – Resampler performance vs quality choice (0=fastest, 3=slowest).

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

reverse()

Plays a sound reversed.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

Note

The sound has to have a finite length and has to be seekable. It’s recommended to use this only with factories with fast and accurate seeking, which is not true for encoded audio files, such ones should be buffered using `cache()` before being played reversed.

Warning

If seeking is not accurate in the underlying sound you’ll likely hear skips/jumps/cracks.

sum()

Sums the samples of a sound.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

threshold(threshold=0)

Makes a threshold wave out of an audio wave by setting all samples with a amplitude >= threshold to 1, all <= -threshold to -1 and all betwe
to 0.

PARAMETERS:

threshold (*float*) – Threshold value over which an amplitude counts non-zero.

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

volume(volume)

Changes the volume of a sound.

PARAMETERS:

volume (*float*) – The new volume..

RETURNS:

The created `Sound` object.

RETURN TYPE:

`Sound`

Note

Should be in the range [0, 1] to avoid clipping.

Note

This is a filter function, you might consider using `Handle.volume` instead.

write(filename, rate, channels, format, container, codec, bitrate, buffersize)

Writes the sound to a file.

PARAMETERS:

- **filename** (*string*) – The path to write to.
- **rate** (*int*) – The sample rate to write with.
- **channels** (*int*) – The number of channels to write with.
- **format** (*int*) – The sample format to write with.
- **container** (*int*) – The container format for the file.
- **codec** (*int*) – The codec to use in the file.
- **bitrate** (*int*) – The bitrate to write with.
- **buffersize** (*int*) – The size of the writing buffer.

length

The sample specification of the sound as a tuple with rate and channel count.

specs

The sample specification of the sound as a tuple with rate and channel count.

class aud.**Source**

The source object represents the source position of a binaural sound.

azimuth

The azimuth angle.

distance

The distance value. 0 is min, 1 is max.

elevation

The elevation angle.

class aud.**ThreadPool**

A ThreadPool is used to parallelize convolution efficiently.

class aud.**error**

Script Operators

bpy.ops.script.**execute_preset(*, filepath='', menu_idname='')**

Load a preset

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – filepath
- **menu_idname** (*string, (optional, never None)*) – Menu ID Name, ID name of the menu this was called from

FILE:

startup/bl_operators/presets.py:273

bpy.ops.script.**python_file_run(*, filepath='')**

Run Python file

PARAMETERS:

filepath (*string, (optional, never None)*) – Path

bpy.ops.script.**reload()**

Reload scripts

Sculpt Operators

bpy.ops.sculpt.**brush_stroke(*, stroke=None, mode='NORMAL', pen_flip=False, override_location=False, ignore_background_click=False)**

Sculpt a stroke into the geometry

PARAMETERS:

- **stroke** (`bpy_prop_collection` of `OperatorStrokeElement`, (optional)) – Stroke
- **mode** (*enum in ['NORMAL', 'INVERT', 'SMOOTH', 'ERASE'], (optional)*) –
  Stroke Mode, Action taken when a paint stroke is made
  - `NORMAL` Regular – Apply brush normally.
  - `INVERT` Invert – Invert action of brush for duration of stroke.
  - `SMOOTH` Smooth – Switch brush to smooth mode for duration of stroke.
  - `ERASE` Erase – Switch brush to erase mode for duration of stroke.
- **pen_flip** (*boolean, (optional)*) – Pen Flip, Whether a tablet’s eraser mode is being used
- **override_location** (*boolean, (optional)*) – Override Location, Override the given *location* array by recalculating object space positions fro
  the provided *mouse_event* positions
- **ignore_background_click** (*boolean, (optional)*) – Ignore Background Click, Clicks on the background do not start the stroke

bpy.ops.sculpt.**cloth_filter(*, start_mouse=(0, 0), area_normal_radius=0.25, strength=1.0, iteration_count=1, event_history=None, type='GRAVITY', force_axis={'X', 'Y', 'Z'}, orientation='LOCAL', cloth_mass=1.0, cloth_damping=0.0, use_face_sets=False, use_collisions=False)**

Applies a cloth simulation deformation to the entire mesh

PARAMETERS:

- **start_mouse** (*int array of 2 items in [0, 16384], (optional)*) – Starting Mouse
- **area_normal_radius** (*float in [0.001, 5], (optional)*) – Normal Radius, Radius used for calculating area normal on initial click,in percentag
  of brush radius
- **strength** (*float in [-10, 10], (optional)*) – Strength, Filter strength
- **iteration_count** (*int in [1, 10000], (optional)*) – Repeat, How many times to repeat the filter
- **type** (*enum in ['GRAVITY', 'INFLATE', 'EXPAND', 'PINCH', 'SCALE'], (optional)*) –
  Filter Type, Operation that is going to be applied to the mesh
  - `GRAVITY` Gravity – Applies gravity to the simulation.
  - `INFLATE` Inflate – Inflates the cloth.
  - `EXPAND` Expand – Expands the cloth’s dimensions.
  - `PINCH` Pinch – Pulls the cloth to the cursor’s start position.
  - `SCALE` Scale – Scales the mesh as a soft body using the origin of the object as scale.
- **force_axis** (*enum set in {'X', 'Y', 'Z'}, (optional)*) –
  Force Axis, Apply the force in the selected axis
  - `X` X – Apply force in the X axis.
  - `Y` Y – Apply force in the Y axis.
  - `Z` Z – Apply force in the Z axis.
- **orientation** (*enum in ['LOCAL', 'WORLD', 'VIEW'], (optional)*) –
  Orientation, Orientation of the axis to limit the filter force
  - `LOCAL` Local – Use the local axis to limit the force and set the gravity direction.
  - `WORLD` World – Use the global axis to limit the force and set the gravity direction.
  - `VIEW` View – Use the view axis to limit the force and set the gravity direction.

- **cloth_mass** (*float in [0, 2], (optional)*) – Cloth Mass, Mass of each simulation particle
- **cloth_damping** (*float in [0, 1], (optional)*) – Cloth Damping, How much the applied forces are propagated through the cloth
- **use_face_sets** (*boolean, (optional)*) – Use Face Sets, Apply the filter only to the Face Set under the cursor
- **use_collisions** (*boolean, (optional)*) – Use Collisions, Collide with other collider objects in the scene

bpy.ops.sculpt.**color_filter(*, start_mouse=(0, 0), area_normal_radius=0.25, strength=1.0, iteration_count=1, event_history=None, type='FILL', fill_color=(1.0, 1.0, 1.0))**

Applies a filter to modify the active color attribute

PARAMETERS:

- **start_mouse** (*int array of 2 items in [0, 16384], (optional)*) – Starting Mouse
- **area_normal_radius** (*float in [0.001, 5], (optional)*) – Normal Radius, Radius used for calculating area normal on initial click,in percentag of brush radius
- **strength** (*float in [-10, 10], (optional)*) – Strength, Filter strength
- **iteration_count** (*int in [1, 10000], (optional)*) – Repeat, How many times to repeat the filter
- **type** (*enum in ['FILL', 'HUE', 'SATURATION', 'VALUE', 'BRIGHTNESS', 'CONTRAST', 'SMOOTH', 'RED', 'GREEN', 'BLUE'], (optional)*) –

  Filter Type

  - `FILL` Fill – Fill with a specific color.
  - `HUE` Hue – Change hue.
  - `SATURATION` Saturation – Change saturation.
  - `VALUE` Value – Change value.
  - `BRIGHTNESS` Brightness – Change brightness.
  - `CONTRAST` Contrast – Change contrast.
  - `SMOOTH` Smooth – Smooth colors.
  - `RED` Red – Change red channel.
  - `GREEN` Green – Change green channel.
  - `BLUE` Blue – Change blue channel.

- **fill_color** (`mathutils.Color` of 3 items in [0, inf], (optional)) – Fill Color

bpy.ops.sculpt.**detail_flood_fill()**

Flood fill the mesh with the selected detail setting

bpy.ops.sculpt.**dynamic_topology_toggle()**

Dynamic topology alters the mesh topology while sculpting

bpy.ops.sculpt.**dyntopo_detail_size_edit()**

Modify the detail size of dyntopo interactively

bpy.ops.sculpt.**expand(*, target='MASK', falloff_type='GEODESIC', invert=False, use_mask_preserve=False, use_falloff_gradient=Fals use_modify_active=False, use_reposition_pivot=True, max_geodesic_move_preview=10000, use_auto_mask=False, normal_falloff_smooth=2)**

Generic sculpt expand operator

PARAMETERS:

- **target** (*enum in ['MASK', 'FACE_SETS', 'COLOR'], (optional)*) – Data Target, Data that is going to be modified in the expand operation
- **falloff_type** (*enum in ['GEODESIC', 'TOPOLOGY', 'TOPOLOGY_DIAGONALS', 'NORMALS', 'SPHERICAL', 'BOUNDARY_TOPOLOGY', 'BOUNDARY_FACE_SET', 'ACTIVE_FACE_SET'], (optional)*) – Falloff Type, Initial falloff of the expand operation
- **invert** (*boolean, (optional)*) – Invert, Invert the expand active elements
- **use_mask_preserve** (*boolean, (optional)*) – Preserve Previous, Preserve the previous state of the target data
- **use_falloff_gradient** (*boolean, (optional)*) – Falloff Gradient, Expand Using a linear falloff

- **use_modify_active** (*boolean, (optional)*) – Modify Active, Modify the active Face Set instead of creating a new one
- **use_reposition_pivot** (*boolean, (optional)*) – Reposition Pivot, Reposition the sculpt transform pivot to the boundary of the expand active area
- **max_geodesic_move_preview** (*int in [0, inf], (optional)*) – Max Vertex Count for Geodesic Move Preview, Maximum number of vertice in the mesh for using geodesic falloff when moving the origin of expand. If the total number of vertices is greater than this value, the falloff will l set to spherical when moving
- **use_auto_mask** (*boolean, (optional)*) – Auto Create, Fill in mask if nothing is already masked
- **normal_falloff_smooth** (*int in [0, 10], (optional)*) – Normal Smooth, Blurring steps for normal falloff

bpy.ops.sculpt.**face_set_box_gesture(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, use_front_faces_only=False)**

Add a face set in a rectangle defined by the cursor

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view

bpy.ops.sculpt.**face_set_change_visibility(*, mode='TOGGLE')**

Change the visibility of the Face Sets of the sculpt

PARAMETERS:

mode (*enum in ['TOGGLE', 'SHOW_ACTIVE', 'HIDE_ACTIVE'], (optional)*) –

Mode

- `TOGGLE` Toggle Visibility – Hide all Face Sets except for the active one.
- `SHOW_ACTIVE` Show Active Face Set – Show Active Face Set.
- `HIDE_ACTIVE` Hide Active Face Sets – Hide Active Face Sets.

bpy.ops.sculpt.**face_set_edit(*, active_face_set=1, mode='GROW', strength=1.0, modify_hidden=False)**

Edits the current active Face Set

PARAMETERS:

- **active_face_set** (*int in [0, inf], (optional)*) – Active Face Set
- **mode** (*enum in ['GROW', 'SHRINK', 'DELETE_GEOMETRY', 'FAIR_POSITIONS', 'FAIR_TANGENCY'], (optional)*) – Mode
  - `GROW` Grow Face Set – Grows the Face Sets boundary by one face based on mesh topology.
  - `SHRINK` Shrink Face Set – Shrinks the Face Sets boundary by one face based on mesh topology.
  - `DELETE_GEOMETRY` Delete Geometry – Deletes the faces that are assigned to the Face Set.
  - `FAIR_POSITIONS` Fair Positions – Creates a smooth as possible geometry patch from the Face Set minimizing changes in vertex positions.
  - `FAIR_TANGENCY` Fair Tangency – Creates a smooth as possible geometry patch from the Face Set minimizing changes in vertex tangents.
- **strength** (*float in [0, 1], (optional)*) – Strength
- **modify_hidden** (*boolean, (optional)*) – Modify Hidden, Apply the edit operation to hidden geometry

bpy.ops.sculpt.**face_set_lasso_gesture(*, path=None, use_smooth_stroke=False, smooth_stroke_factor=0.75, smooth_stroke_radius=35, use_front_faces_only=False)**

Add a face set in a shape defined by the cursor

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **use_smooth_stroke** (*boolean, (optional)*) – Stabilize Stroke, Selection lags behind mouse and follows a smoother path
- **smooth_stroke_factor** (*float in [0.5, 0.99], (optional)*) – Smooth Stroke Factor, Higher values gives a smoother stroke
- **smooth_stroke_radius** (*int in [10, 200], (optional)*) – Smooth Stroke Radius, Minimum distance from last point before selection continues
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view

bpy.ops.sculpt.**face_set_line_gesture(*, xstart=0, xend=0, ystart=0, yend=0, flip=False, cursor=5, use_front_faces_only=False, use_limit_to_segment=False)**

Add a face set to one side of a line defined by the cursor

PARAMETERS:

- **xstart** (*int in [-inf, inf], (optional)*) – X Start
- **xend** (*int in [-inf, inf], (optional)*) – X End
- **ystart** (*int in [-inf, inf], (optional)*) – Y Start
- **yend** (*int in [-inf, inf], (optional)*) – Y End
- **flip** (*boolean, (optional)*) – Flip
- **cursor** (*int in [0, inf], (optional)*) – Cursor, Mouse cursor style to use during the modal operator
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **use_limit_to_segment** (*boolean, (optional)*) – Limit to Segment, Apply the gesture action only to the area that is contained within the segment without extending its effect to the entire line

bpy.ops.sculpt.**face_set_polyline_gesture(*, path=None, use_front_faces_only=False)**

Add a face set in a shape defined by the cursor

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view

bpy.ops.sculpt.**face_sets_create(*, mode='MASKED')**

Create a new Face Set

PARAMETERS:

mode (*enum in ['MASKED', 'VISIBLE', 'ALL', 'SELECTION'], (optional)*) –

Mode

- `MASKED` Face Set from Masked – Create a new Face Set from the masked faces.
- `VISIBLE` Face Set from Visible – Create a new Face Set from the visible vertices.
- `ALL` Face Set Full Mesh – Create an unique Face Set with all faces in the sculpt.
- `SELECTION` Face Set from Edit Mode Selection – Create an Face Set corresponding to the Edit Mode face selection.

bpy.ops.sculpt.**face_sets_init(*, mode='LOOSE_PARTS', threshold=0.5)**

Initializes all Face Sets in the mesh

PARAMETERS:

- **mode** (*enum in ['LOOSE_PARTS', 'MATERIALS', 'NORMALS', 'UV_SEAMS', 'CREASES', 'BEVEL_WEIGHT', 'SHARP_EDGES', 'FACE_SET_BOUNDARIES'], (optional)*) –

  Mode

  - `LOOSE_PARTS` Face Sets from Loose Parts – Create a Face Set per loose part in the mesh.
  - `MATERIALS` Face Sets from Material Slots – Create a Face Set per Material Slot.
  - `NORMALS` Face Sets from Mesh Normals – Create Face Sets for Faces that have similar normal.
  - `UV_SEAMS` Face Sets from UV Seams – Create Face Sets using UV Seams as boundaries.
  - `CREASES` Face Sets from Edge Creases – Create Face Sets using Edge Creases as boundaries.
  - `BEVEL_WEIGHT` Face Sets from Bevel Weight – Create Face Sets using Bevel Weights as boundaries.

    - `BEVEL_WEIGHT` Face Sets from Bevel Weight – Create Face Sets using Bevel Weights as boundaries.
    - `SHARP_EDGES` Face Sets from Sharp Edges – Create Face Sets using Sharp Edges as boundaries.
    - `FACE_SET_BOUNDARIES` Face Sets from Face Set Boundaries – Create a Face Set per isolated Face Set.

- **threshold** (*float in [0, 1], (optional)*) – Threshold, Minimum value to consider a certain attribute a boundary when creating the Face Sets

bpy.ops.sculpt.**face_sets_randomize_colors()**

Generates a new set of random colors to render the Face Sets in the viewport

bpy.ops.sculpt.**mask_by_color(*, contiguous=False, invert=False, preserve_previous_mask=False, threshold=0.35)**

Creates a mask based on the active color attribute

PARAMETERS:

- **contiguous** (*boolean, (optional)*) – Contiguous, Mask only contiguous color areas
- **invert** (*boolean, (optional)*) – Invert, Invert the generated mask
- **preserve_previous_mask** (*boolean, (optional)*) – Preserve Previous Mask, Preserve the previous mask and add or subtract the new one generated by the colors
- **threshold** (*float in [0, 1], (optional)*) – Threshold, How much changes in color affect the mask generation

bpy.ops.sculpt.**mask_filter(*, filter_type='SMOOTH', iterations=1, auto_iteration_count=True)**

Applies a filter to modify the current mask

PARAMETERS:

- **filter_type** (*enum in ['SMOOTH', 'SHARPEN', 'GROW', 'SHRINK', 'CONTRAST_INCREASE', 'CONTRAST_DECREASE'], (optional*
  – Type, Filter that is going to be applied to the mask
- **iterations** (*int in [1, 100], (optional)*) – Iterations, Number of times that the filter is going to be applied
- **auto_iteration_count** (*boolean, (optional)*) – Auto Iteration Count, Use an automatic number of iterations based on the number of vertices the sculpt

bpy.ops.sculpt.**mask_from_boundary(*, mix_mode='MIX', mix_factor=1.0, settings_source='OPERATOR', boundary_mode='MESH', propagation_steps=1)**

Creates a mask based on the boundaries of the surface

PARAMETERS:

- **mix_mode** (*enum in ['MIX', 'MULTIPLY', 'DIVIDE', 'ADD', 'SUBTRACT'], (optional)*) – Mode, Mix mode
- **mix_factor** (*float in [0, 5], (optional)*) – Mix Factor
- **settings_source** (*enum in ['OPERATOR', 'BRUSH', 'SCENE'], (optional)*) –
  Settings, Use settings from here
    - `OPERATOR` Operator – Use settings from operator properties.
    - `BRUSH` Brush – Use settings from brush.
    - `SCENE` Scene – Use settings from scene.
- **boundary_mode** (*enum in ['MESH', 'FACE_SETS'], (optional)*) –
  Mode, Boundary type to mask
    - `MESH` Mesh – Calculate the boundary mask based on disconnected mesh topology islands.
    - `FACE_SETS` Face Sets – Calculate the boundary mask between face sets.
- **propagation_steps** (*int in [1, 20], (optional)*) – Propagation Steps

bpy.ops.sculpt.**mask_from_cavity(*, mix_mode='MIX', mix_factor=1.0, settings_source='OPERATOR', factor=0.5, blur_steps=2, use_curve=False, invert=False)**

Creates a mask based on the curvature of the surface

PARAMETERS:

- **mix_mode** (*enum in ['MIX', 'MULTIPLY', 'DIVIDE', 'ADD', 'SUBTRACT'], (optional)*) – Mode, Mix mode

- **mix_factor** (*float in [0, 5], (optional)*) – Mix Factor
- **settings_source** (*enum in ['OPERATOR', 'BRUSH', 'SCENE'], (optional)*) –
  Settings, Use settings from here
  - `OPERATOR` Operator – Use settings from operator properties.
  - `BRUSH` Brush – Use settings from brush.
  - `SCENE` Scene – Use settings from scene.
- **factor** (*float in [0, 5], (optional)*) – Factor, The contrast of the cavity mask
- **blur_steps** (*int in [0, 25], (optional)*) – Blur, The number of times the cavity mask is blurred
- **use_curve** (*boolean, (optional)*) – Custom Curve
- **invert** (*boolean, (optional)*) – Cavity (Inverted)

bpy.ops.sculpt.**mask_init(*, mode='RANDOM_PER_VERTEX')**

Creates a new mask for the entire mesh

PARAMETERS:

mode (*enum in ['RANDOM_PER_VERTEX', 'RANDOM_PER_FACE_SET', 'RANDOM_PER_LOOSE_PART'], (optional)*) – Mode

bpy.ops.sculpt.**mesh_filter(*, start_mouse=(0, 0), area_normal_radius=0.25, strength=1.0, iteration_count=1, event_history=None, type='INFLATE', deform_axis={'X', 'Y', 'Z'}, orientation='LOCAL', surface_smooth_shape_preservation=0.5, surface_smooth_current_vertex=0.5, sharpen_smooth_ratio=0.35, sharpen_intensify_detail_strength=0.0, sharpen_curvature_smooth_iterations=0)**

Applies a filter to modify the current mesh

PARAMETERS:

- **start_mouse** (*int array of 2 items in [0, 16384], (optional)*) – Starting Mouse
- **area_normal_radius** (*float in [0.001, 5], (optional)*) – Normal Radius, Radius used for calculating area normal on initial click,in percentag of brush radius
- **strength** (*float in [-10, 10], (optional)*) – Strength, Filter strength
- **iteration_count** (*int in [1, 10000], (optional)*) – Repeat, How many times to repeat the filter
- **type** (*enum in ['SMOOTH', 'SCALE', 'INFLATE', 'SPHERE', 'RANDOM', 'RELAX', 'RELAX_FACE_SETS', 'SURFACE_SMOOTH', 'SHARPEN', 'ENHANCE_DETAILS', 'ERASE_DISPLACEMENT'], (optional)*) –
  Filter Type, Operation that is going to be applied to the mesh
  - `SMOOTH` Smooth – Smooth mesh.
  - `SCALE` Scale – Scale mesh.
  - `INFLATE` Inflate – Inflate mesh.
  - `SPHERE` Sphere – Morph into sphere.
  - `RANDOM` Random – Randomize vertex positions.
  - `RELAX` Relax – Relax mesh.
  - `RELAX_FACE_SETS` Relax Face Sets – Smooth the edges of all the Face Sets.
  - `SURFACE_SMOOTH` Surface Smooth – Smooth the surface of the mesh, preserving the volume.
  - `SHARPEN` Sharpen – Sharpen the cavities of the mesh.
  - `ENHANCE_DETAILS` Enhance Details – Enhance the high frequency surface detail.
  - `ERASE_DISPLACEMENT` Erase Displacement – Deletes the displacement of the Multires Modifier.
- **deform_axis** (*enum set in {'X', 'Y', 'Z'}, (optional)*) –
  Deform Axis, Apply the deformation in the selected axis
  - `X` X – Deform in the X axis.
  - `Y` Y – Deform in the Y axis.
  - `Z` Z – Deform in the Z axis.
- **orientation** (*enum in ['LOCAL', 'WORLD', 'VIEW'], (optional)*) –
  Orientation, Orientation of the axis to limit the filter displacement

- `LOCAL` Local – Use the local axis to limit the displacement.
- `WORLD` World – Use the global axis to limit the displacement.
- `VIEW` View – Use the view axis to limit the displacement.

- **surface_smooth_shape_preservation** (*float in [0, 1], (optional)*) – Shape Preservation, How much of the original shape is preserved wh smoothing
- **surface_smooth_current_vertex** (*float in [0, 1], (optional)*) – Per Vertex Displacement, How much the position of each individual vertex influences the final result
- **sharpen_smooth_ratio** (*float in [0, 1], (optional)*) – Smooth Ratio, How much smoothing is applied to polished surfaces
- **sharpen_intensify_detail_strength** (*float in [0, 10], (optional)*) – Intensify Details, How much creases and valleys are intensified
- **sharpen_curvature_smooth_iterations** (*int in [0, 10], (optional)*) – Curvature Smooth Iterations, How much smooth the resulting shape ignoring high frequency details

bpy.ops.sculpt.**optimize()**

Recalculate the sculpt BVH to improve performance

bpy.ops.sculpt.**project_line_gesture(*, xstart=0, xend=0, ystart=0, yend=0, flip=False, cursor=5, use_front_faces_only=False, use_limit_to_segment=False)**

Project the geometry onto a plane defined by a line

PARAMETERS:

- **xstart** (*int in [-inf, inf], (optional)*) – X Start
- **xend** (*int in [-inf, inf], (optional)*) – X End
- **ystart** (*int in [-inf, inf], (optional)*) – Y Start
- **yend** (*int in [-inf, inf], (optional)*) – Y End
- **flip** (*boolean, (optional)*) – Flip
- **cursor** (*int in [0, inf], (optional)*) – Cursor, Mouse cursor style to use during the modal operator
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **use_limit_to_segment** (*boolean, (optional)*) – Limit to Segment, Apply the gesture action only to the area that is contained within the segment without extending its effect to the entire line

bpy.ops.sculpt.**sample_color()**

Sample the vertex color of the active vertex

bpy.ops.sculpt.**sample_detail_size(*, location=(0, 0), mode='DYNTOPO')**

Sample the mesh detail on clicked point

PARAMETERS:

- **location** (*int array of 2 items in [0, 32767], (optional)*) – Location, Screen coordinates of sampling
- **mode** (*enum in ['DYNTOPO', 'VOXEL'], (optional)*) –

  Detail Mode, Target sculpting workflow that is going to use the sampled size

  - `DYNTOPO` Dyntopo – Sample dyntopo detail.
  - `VOXEL` Voxel – Sample mesh voxel size.

bpy.ops.sculpt.**sculptmode_toggle()**

Toggle sculpt mode in 3D view

bpy.ops.sculpt.**set_persistent_base()**

Reset the copy of the mesh that is being sculpted on

bpy.ops.sculpt.**set_pivot_position(*, mode='UNMASKED', mouse_x=0.0, mouse_y=0.0)**

Sets the sculpt transform pivot position

PARAMETERS:

- **mode** (*enum in ['ORIGIN', 'UNMASKED', 'BORDER', 'ACTIVE', 'SURFACE'], (optional)*) –
  Mode
  - `ORIGIN` Origin – Sets the pivot to the origin of the sculpt.
  - `UNMASKED` Unmasked – Sets the pivot position to the average position of the unmasked vertices.
  - `BORDER` Mask Border – Sets the pivot position to the center of the border of the mask.
  - `ACTIVE` Active Vertex – Sets the pivot position to the active vertex position.
  - `SURFACE` Surface – Sets the pivot position to the surface under the cursor.
- **mouse_x** (*float in [0, inf], (optional)*) – Mouse Position X, Position of the mouse used for “Surface” mode
- **mouse_y** (*float in [0, inf], (optional)*) – Mouse Position Y, Position of the mouse used for “Surface” mode

bpy.ops.sculpt.**symmetrize(*, merge_tolerance=0.0005)**

Symmetrize the topology modifications

PARAMETERS:

merge_tolerance (*float in [0, inf], (optional)*) – Merge Distance, Distance within which symmetrical vertices are merged

bpy.ops.sculpt.**trim_box_gesture(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, use_front_faces_only=False, location=(0, 0), trim_mode='DIFFERENCE', use_cursor_depth=False, trim_orientation='VIEW', trim_extrude_mode='FIXED', trim_solver='FAST')**

Execute a boolean operation on the mesh and a rectangle defined by the cursor

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **location** (*int array of 2 items in [-inf, inf], (optional)*) – Location, Mouse location
- **trim_mode** (*enum in ['DIFFERENCE', 'UNION', 'JOIN'], (optional)*) –
  Trim Mode
  - `DIFFERENCE` Difference – Use a difference boolean operation.
  - `UNION` Union – Use a union boolean operation.
  - `JOIN` Join – Join the new mesh as separate geometry, without performing any boolean operation.
- **use_cursor_depth** (*boolean, (optional)*) – Use Cursor for Depth, Use cursor location and radius for the dimensions and position of the trimming shape
- **trim_orientation** (*enum in ['VIEW', 'SURFACE'], (optional)*) –
  Shape Orientation
  - `VIEW` View – Use the view to orientate the trimming shape.
  - `SURFACE` Surface – Use the surface normal to orientate the trimming shape.
- **trim_extrude_mode** (*enum in ['PROJECT', 'FIXED'], (optional)*) –
  Extrude Mode
  - `PROJECT` Project – Align trim geometry with the perspective of the current view for a tapered shape.
  - `FIXED` Fixed – Align trim geometry orthogonally for a shape with 90 degree angles.
- **trim_solver** (*enum in ['EXACT', 'FAST'], (optional)*) –
  Solver
  - `EXACT` Exact – Use the exact boolean solver.

- `FAST` Fast – Use the fast float boolean solver.

bpy.ops.sculpt.**trim_lasso_gesture(*, path=None, use_smooth_stroke=False, smooth_stroke_factor=0.75, smooth_stroke_radius=35, use_front_faces_only=False, location=(0, 0), trim_mode='DIFFERENCE', use_cursor_depth=False, trim_orientation='VIEW', trim_extrude_mode='FIXED', trim_solver='FAST')**

Execute a boolean operation on the mesh and a shape defined by the cursor

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **use_smooth_stroke** (*boolean, (optional)*) – Stabilize Stroke, Selection lags behind mouse and follows a smoother path
- **smooth_stroke_factor** (*float in [0.5, 0.99], (optional)*) – Smooth Stroke Factor, Higher values gives a smoother stroke
- **smooth_stroke_radius** (*int in [10, 200], (optional)*) – Smooth Stroke Radius, Minimum distance from last point before selection continues
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **location** (*int array of 2 items in [-inf, inf], (optional)*) – Location, Mouse location
- **trim_mode** (*enum in ['DIFFERENCE', 'UNION', 'JOIN'], (optional)*) –

  Trim Mode

  - `DIFFERENCE` Difference – Use a difference boolean operation.
  - `UNION` Union – Use a union boolean operation.
  - `JOIN` Join – Join the new mesh as separate geometry, without performing any boolean operation.
- **use_cursor_depth** (*boolean, (optional)*) – Use Cursor for Depth, Use cursor location and radius for the dimensions and position of the trimming shape
- **trim_orientation** (*enum in ['VIEW', 'SURFACE'], (optional)*) –

  Shape Orientation

  - `VIEW` View – Use the view to orientate the trimming shape.
  - `SURFACE` Surface – Use the surface normal to orientate the trimming shape.
- **trim_extrude_mode** (*enum in ['PROJECT', 'FIXED'], (optional)*) –

  Extrude Mode

  - `PROJECT` Project – Align trim geometry with the perspective of the current view for a tapered shape.
  - `FIXED` Fixed – Align trim geometry orthogonally for a shape with 90 degree angles.
- **trim_solver** (*enum in ['EXACT', 'FAST'], (optional)*) –

  Solver

  - `EXACT` Exact – Use the exact boolean solver.
  - `FAST` Fast – Use the fast float boolean solver.

bpy.ops.sculpt.**trim_line_gesture(*, xstart=0, xend=0, ystart=0, yend=0, flip=False, cursor=5, use_front_faces_only=False, use_limit_to_segment=False, location=(0, 0), trim_mode='DIFFERENCE', use_cursor_depth=False, trim_orientation='VIEW', trim_extrude_mode='FIXED', trim_solver='FAST')**

Remove a portion of the mesh on one side of a line

PARAMETERS:

- **xstart** (*int in [-inf, inf], (optional)*) – X Start
- **xend** (*int in [-inf, inf], (optional)*) – X End
- **ystart** (*int in [-inf, inf], (optional)*) – Y Start
- **yend** (*int in [-inf, inf], (optional)*) – Y End
- **flip** (*boolean, (optional)*) – Flip
- **cursor** (*int in [0, inf], (optional)*) – Cursor, Mouse cursor style to use during the modal operator
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **use_limit_to_segment** (*boolean, (optional)*) – Limit to Segment, Apply the gesture action only to the area that is contained within the segment without extending its effect to the entire line

- **location** (*int array of 2 items in [-inf, inf], (optional)*) – Location, Mouse location
- **trim_mode** (*enum in ['DIFFERENCE', 'UNION', 'JOIN'], (optional)*) –

  Trim Mode

  - `DIFFERENCE` Difference – Use a difference boolean operation.
  - `UNION` Union – Use a union boolean operation.
  - `JOIN` Join – Join the new mesh as separate geometry, without performing any boolean operation.
- **use_cursor_depth** (*boolean, (optional)*) – Use Cursor for Depth, Use cursor location and radius for the dimensions and position of the trimming shape
- **trim_orientation** (*enum in ['VIEW', 'SURFACE'], (optional)*) –

  Shape Orientation

  - `VIEW` View – Use the view to orientate the trimming shape.
  - `SURFACE` Surface – Use the surface normal to orientate the trimming shape.
- **trim_extrude_mode** (*enum in ['PROJECT', 'FIXED'], (optional)*) –

  Extrude Mode

  - `PROJECT` Project – Align trim geometry with the perspective of the current view for a tapered shape.
  - `FIXED` Fixed – Align trim geometry orthogonally for a shape with 90 degree angles.
- **trim_solver** (*enum in ['EXACT', 'FAST'], (optional)*) –

  Solver

  - `EXACT` Exact – Use the exact boolean solver.
  - `FAST` Fast – Use the fast float boolean solver.

bpy.ops.sculpt.**trim_polyline_gesture(*, path=None, use_front_faces_only=False, location=(0, 0), trim_mode='DIFFERENCE', use_cursor_depth=False, trim_orientation='VIEW', trim_extrude_mode='FIXED', trim_solver='FAST')**

Execute a boolean operation on the mesh and a polygonal shape defined by the cursor

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **location** (*int array of 2 items in [-inf, inf], (optional)*) – Location, Mouse location
- **trim_mode** (*enum in ['DIFFERENCE', 'UNION', 'JOIN'], (optional)*) –

  Trim Mode

  - `DIFFERENCE` Difference – Use a difference boolean operation.
  - `UNION` Union – Use a union boolean operation.
  - `JOIN` Join – Join the new mesh as separate geometry, without performing any boolean operation.
- **use_cursor_depth** (*boolean, (optional)*) – Use Cursor for Depth, Use cursor location and radius for the dimensions and position of the trimming shape
- **trim_orientation** (*enum in ['VIEW', 'SURFACE'], (optional)*) –

  Shape Orientation

  - `VIEW` View – Use the view to orientate the trimming shape.
  - `SURFACE` Surface – Use the surface normal to orientate the trimming shape.
- **trim_extrude_mode** (*enum in ['PROJECT', 'FIXED'], (optional)*) –

  Extrude Mode

  - `PROJECT` Project – Align trim geometry with the perspective of the current view for a tapered shape.
  - `FIXED` Fixed – Align trim geometry orthogonally for a shape with 90 degree angles.
- **trim_solver** (*enum in ['EXACT', 'FAST'], (optional)*) –

  Solver

- `EXACT` Exact – Use the exact boolean solver.
- `FAST` Fast – Use the fast float boolean solver.

bpy.ops.sculpt.**uv_sculpt_grab(*, use_invert=False)**

Grab UVs

PARAMETERS:

use_invert (*boolean, (optional)*) – Invert, Invert action for the duration of the stroke

bpy.ops.sculpt.**uv_sculpt_pinch(*, use_invert=False)**

Pinch UVs

PARAMETERS:

use_invert (*boolean, (optional)*) – Invert, Invert action for the duration of the stroke

bpy.ops.sculpt.**uv_sculpt_relax(*, use_invert=False, relax_method='COTAN')**

Relax UVs

PARAMETERS:

- **use_invert** (*boolean, (optional)*) – Invert, Invert action for the duration of the stroke
- **relax_method** (*enum in ['LAPLACIAN', 'HC', 'COTAN'], (optional)*) –

  Relax Method, Algorithm used for UV relaxation

  - `LAPLACIAN` Laplacian – Use Laplacian method for relaxation.
  - `HC` HC – Use HC method for relaxation.
  - `COTAN` Geometry – Use Geometry (cotangent) relaxation, making UVs follow the underlying 3D geometry.

Sculpt Curves Operators

bpy.ops.sculpt_curves.**brush_stroke(*, stroke=None, mode='NORMAL', pen_flip=False)**

Sculpt curves using a brush

PARAMETERS:

- **stroke** (`bpy_prop_collection` of `OperatorStrokeElement`, (optional)) – Stroke
- **mode** (*enum in ['NORMAL', 'INVERT', 'SMOOTH', 'ERASE'], (optional)*) –

  Stroke Mode, Action taken when a paint stroke is made

  - `NORMAL` Regular – Apply brush normally.
  - `INVERT` Invert – Invert action of brush for duration of stroke.
  - `SMOOTH` Smooth – Switch brush to smooth mode for duration of stroke.
  - `ERASE` Erase – Switch brush to erase mode for duration of stroke.
- **pen_flip** (*boolean, (optional)*) – Pen Flip, Whether a tablet's eraser mode is being used

bpy.ops.sculpt_curves.**min_distance_edit()**

Change the minimum distance used by the density brush

bpy.ops.sculpt_curves.**select_grow(*, distance=0.1)**

Select curves which are close to curves that are selected already

PARAMETERS:

distance (*float in [-inf, inf], (optional)*) – Distance, By how much to grow the selection

bpy.ops.sculpt_curves.**select_random(*, seed=0, partial=False, probability=0.5, min=0.0, constant_per_curve=True)**

Randomizes existing selection or create new random selection

PARAMETERS:

- **seed** (*int in [-inf, inf], (optional)*) – Seed, Source of randomness
- **partial** (*boolean, (optional)*) – Partial, Allow points or curves to be selected partially
- **probability** (*float in [0, 1], (optional)*) – Probability, Chance of every point or curve being included in the selection
- **min** (*float in [0, 1], (optional)*) – Min, Minimum value for the random selection
- **constant_per_curve** (*boolean, (optional)*) – Constant per Curve, The generated random number is the same for every control point of a curve

Skip to content

Sequencer Operators

bpy.ops.sequencer.**change_effect_input()**

Undocumented, consider contributing.

bpy.ops.sequencer.**change_effect_type(*, type='CROSS')**

Undocumented, consider contributing.

PARAMETERS:

type (*enum in ['CROSS', 'ADD', 'SUBTRACT', 'ALPHA_OVER', 'ALPHA_UNDER', 'GAMMA_CROSS', 'MULTIPLY', 'OVER_DROP', 'WIPE', 'GLOW', 'TRANSFORM', 'COLOR', 'SPEED', 'MULTICAM', 'ADJUSTMENT', 'GAUSSIAN_BLUR', 'TEXT', 'COLORMIX'], (optional)*) –

Type, Sequencer effect type

- `CROSS` Crossfade – Crossfade effect strip type.
- `ADD` Add – Add effect strip type.
- `SUBTRACT` Subtract – Subtract effect strip type.
- `ALPHA_OVER` Alpha Over – Alpha Over effect strip type.
- `ALPHA_UNDER` Alpha Under – Alpha Under effect strip type.
- `GAMMA_CROSS` Gamma Cross – Gamma Cross effect strip type.
- `MULTIPLY` Multiply – Multiply effect strip type.
- `OVER_DROP` Alpha Over Drop – Alpha Over Drop effect strip type.
- `WIPE` Wipe – Wipe effect strip type.
- `GLOW` Glow – Glow effect strip type.
- `TRANSFORM` Transform – Transform effect strip type.
- `COLOR` Color – Color effect strip type.
- `SPEED` Speed – Color effect strip type.
- `MULTICAM` Multicam Selector.
- `ADJUSTMENT` Adjustment Layer.
- `GAUSSIAN_BLUR` Gaussian Blur.
- `TEXT` Text.
- `COLORMIX` Color Mix.

bpy.ops.sequencer.**change_path(*, filepath='', directory='', files=None, hide_props_region=True, check_existing=False, 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=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, display_type='DEFAULT', sort_method='', use_placeholders=False)**

Undocumented, consider contributing.

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_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_placeholders** (*boolean, (optional)*) – Use Placeholders, Use placeholders for missing frames of the strip

bpy.ops.sequencer.**change_scene(*, scene='')**

Change Scene assigned to Strip

PARAMETERS:

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

bpy.ops.sequencer.**connect(*, toggle=True)**

Link selected strips together for simplified group selection

PARAMETERS:

toggle (*boolean, (optional)*) – Toggle, Toggle strip connections

bpy.ops.sequencer.**copy()**

Copy the selected strips to the internal clipboard

bpy.ops.sequencer.**crossfade_sounds()**

Do cross-fading volume animation of two selected sound strips

FILE:

startup/bl_operators/sequencer.py:40

bpy.ops.sequencer.**cursor_set(*, location=(0.0, 0.0))**

Set 2D cursor location

PARAMETERS:

location (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Cursor location in normalized preview coordinates

bpy.ops.sequencer.**deinterlace_selected_movies()**

Deinterlace all selected movie sources

FILE:

startup/bl_operators/sequencer.py:128

bpy.ops.sequencer.**delete(*, delete_data=False)**

Delete selected strips from the sequencer

PARAMETERS:

delete_data (*boolean, (optional)*) – Delete Data, After removing the Strip, delete the associated data also

bpy.ops.sequencer.**disconnect()**

Unlink selected strips so that they can be selected individually

bpy.ops.sequencer.**duplicate()**

Duplicate the selected strips

bpy.ops.sequencer.**duplicate_move(*, SEQUENCER_OT_duplicate=None, TRANSFORM_OT_seq_slide=None)**

Duplicate selected strips and move them

PARAMETERS:

- **SEQUENCER_OT_duplicate** (`SEQUENCER_OT_duplicate`, (optional)) – Duplicate Strips, Duplicate the selected strips
- **TRANSFORM_OT_seq_slide** (`TRANSFORM_OT_seq_slide`, (optional)) – Sequence Slide, Slide a sequence strip in time

bpy.ops.sequencer.**effect_strip_add(*, type='CROSS', frame_start=0, frame_end=0, channel=1, replace_sel=True, overlap=False, overlap_shuffle_override=False, color=(0.0, 0.0, 0.0))**

Add an effect to the sequencer, most are applied on top of existing strips

PARAMETERS:

- **type** (*enum in ['CROSS', 'ADD', 'SUBTRACT', 'ALPHA_OVER', 'ALPHA_UNDER', 'GAMMA_CROSS', 'MULTIPLY', 'OVER_DROP', 'WIPE', 'GLOW', 'TRANSFORM', 'COLOR', 'SPEED', 'MULTICAM', 'ADJUSTMENT', 'GAUSSIAN_BLUR', 'TEXT', 'COLORMIX'], (optional)*) –

  Type, Sequencer effect type

  - `CROSS` Crossfade – Crossfade effect strip type.
  - `ADD` Add – Add effect strip type.
  - `SUBTRACT` Subtract – Subtract effect strip type.
  - `ALPHA_OVER` Alpha Over – Alpha Over effect strip type.
  - `ALPHA_UNDER` Alpha Under – Alpha Under effect strip type.
  - `GAMMA_CROSS` Gamma Cross – Gamma Cross effect strip type.
  - `MULTIPLY` Multiply – Multiply effect strip type.
  - `OVER_DROP` Alpha Over Drop – Alpha Over Drop effect strip type.
  - `WIPE` Wipe – Wipe effect strip type.
  - `GLOW` Glow – Glow effect strip type.
  - `TRANSFORM` Transform – Transform effect strip type.
  - `COLOR` Color – Color effect strip type.
  - `SPEED` Speed – Color effect strip type.
  - `MULTICAM` Multicam Selector.
  - `ADJUSTMENT` Adjustment Layer.
  - `GAUSSIAN_BLUR` Gaussian Blur.
  - `TEXT` Text.
  - `COLORMIX` Color Mix.

- **frame_start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the sequence strip
- **frame_end** (*int in [-inf, inf], (optional)*) – End Frame, End frame for the color strip
- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into

- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into
- **replace_sel** (*boolean, (optional)*) – Replace Selection, Deselect previously selected strips
- **overlap** (*boolean, (optional)*) – Allow Overlap, Don't correct overlap on new sequence strips
- **overlap_shuffle_override** (*boolean, (optional)*) – Override Overlap Shuffle Behavior, Use the overlap_mode tool settings to determine ho to shuffle overlapping strips
- **color** (`mathutils.Color` of 3 items in [0, 1], (optional)) – Color, Initialize the strip with this color

bpy.ops.sequencer.**enable_proxies(*, proxy_25=False, proxy_50=False, proxy_75=False, proxy_100=False, overwrite=False)**

Enable selected proxies on all selected Movie and Image strips

PARAMETERS:

- **proxy_25** (*boolean, (optional)*) – 25%
- **proxy_50** (*boolean, (optional)*) – 50%
- **proxy_75** (*boolean, (optional)*) – 75%
- **proxy_100** (*boolean, (optional)*) – 100%
- **overwrite** (*boolean, (optional)*) – Overwrite

bpy.ops.sequencer.**export_subtitles(*, 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='')**

Export .srt file containing text strips

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
- **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

bpy.ops.sequencer.**fades_add(*, duration_seconds=1.0, type='IN_OUT')**

Adds or updates a fade animation for either visual or audio strips

PARAMETERS:

- **duration_seconds** (*float in [0.01, inf], (optional)*) – Fade Duration, Duration of the fade in seconds
- **type** (*enum in ['IN_OUT', 'IN', 'OUT', 'CURSOR_FROM', 'CURSOR_TO'], (optional)*) – Fade Type, Fade in, out, both in and out, to, or from the current frame. Default is both in and out
   - `IN_OUT` Fade In and Out – Fade selected strips in and out.
   - `IN` Fade In – Fade in selected strips.
   - `OUT` Fade Out – Fade out selected strips.
   - `CURSOR_FROM` From Current Frame – Fade from the time cursor to the end of overlapping sequences.
   - `CURSOR_TO` To Current Frame – Fade from the start of sequences under the time cursor to the current frame.

FILE:

startup/bl_operators/sequencer.py:206

bpy.ops.sequencer.**fades_clear()**

Removes fade animation from selected sequences

FILE:

startup/bl_operators/sequencer.py:147

bpy.ops.sequencer.**gap_insert(*, frames=10)**

Insert gap at current frame to first strips at the right, independent of selection or locked state of strips

PARAMETERS:

frames (*int in [0, inf], (optional)*) – Frames, Frames to insert after current strip

bpy.ops.sequencer.**gap_remove(*, all=False)**

Remove gap at current frame to first strip at the right, independent of selection or locked state of strips

PARAMETERS:

all (*boolean, (optional)*) – All Gaps, Do all gaps to right of current frame

bpy.ops.sequencer.**image_strip_add(*, directory='', files=None, check_existing=False, filter_blender=False, filter_backup=False, filter_image=True, 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=False, filter_folder=True, filter_blenlib=False, filemode=9, relative_path=True, show_multiview=False, use_multiview=False, display_type='DEFAULT', sort_method='', frame_start=0, frame_end=0, channel=1, replace_sel=True, overlap=False, overlap_shuffle_override=False, fit_method='FIT', set_view_transform=True, use_placeholders=False)**

Add an image or image sequence to the sequencer

PARAMETERS:

- **directory** (*string, (optional, never None)*) – Directory, Directory of the file
- **files** (`bpy_prop_collection` of `OperatorFileListElement`, (optional)) – Files
- **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..
- **frame_start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the sequence strip
- **frame_end** (*int in [-inf, inf], (optional)*) – End Frame, End frame for the color strip
- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into
- **replace_sel** (*boolean, (optional)*) – Replace Selection, Deselect previously selected strips
- **overlap** (*boolean, (optional)*) – Allow Overlap, Don't correct overlap on new sequence strips
- **overlap_shuffle_override** (*boolean, (optional)*) – Override Overlap Shuffle Behavior, Use the overlap_mode tool settings to determine ho to shuffle overlapping strips
- **fit_method** (*enum in ['FIT', 'FILL', 'STRETCH', 'ORIGINAL'], (optional)*) –

  Fit Method, Scale fit method

  - `FIT` Scale to Fit – Scale image to fit within the canvas.
  - `FILL` Scale to Fill – Scale image to completely fill the canvas.
  - `STRETCH` Stretch to Fill – Stretch image to fill the canvas.
  - `ORIGINAL` Use Original Size – Keep image at its original size.
- **set_view_transform** (*boolean, (optional)*) – Set View Transform, Set appropriate view transform based on media color space
- **use_placeholders** (*boolean, (optional)*) – Use Placeholders, Use placeholders for missing frames of the strip

bpy.ops.sequencer.**images_separate(*, length=1)**

On image sequence strips, it returns a strip for each image

PARAMETERS:

length (*int in [1, inf], (optional)*) – Length, Length of each frame

bpy.ops.sequencer.**lock()**

Lock strips so they can’t be transformed

bpy.ops.sequencer.**mask_strip_add(*, frame_start=0, channel=1, replace_sel=True, overlap=False, overlap_shuffle_override=False, mask='')**

Add a mask strip to the sequencer

PARAMETERS:

- **frame_start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the sequence strip
- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into
- **replace_sel** (*boolean, (optional)*) – Replace Selection, Deselect previously selected strips
- **overlap** (*boolean, (optional)*) – Allow Overlap, Don’t correct overlap on new sequence strips
- **overlap_shuffle_override** (*boolean, (optional)*) – Override Overlap Shuffle Behavior, Use the overlap_mode tool settings to determine ho to shuffle overlapping strips
- **mask** (*enum in [], (optional)*) – Mask

bpy.ops.sequencer.**meta_make()**

Group selected strips into a meta-strip

bpy.ops.sequencer.**meta_separate()**

Put the contents of a meta-strip back in the sequencer

bpy.ops.sequencer.**meta_toggle()**

Toggle a meta-strip (to edit enclosed strips)

bpy.ops.sequencer.**movie_strip_add(*, filepath='', directory='', files=None, check_existing=False, filter_blender=False, filter_backup=False, filter_image=False, 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='', frame_start=0, channel=1, replace_sel=True, overlap=False, overlap_shuffle_override=False, fit_method='FIT', set_view_transform=True, adjust_playback_rate=True, sound=True, use_framerate=True)**

Add a movie strip to the sequencer

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
- **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..
- **frame_start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the sequence strip
- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into
- **replace_sel** (*boolean, (optional)*) – Replace Selection, Deselect previously selected strips
- **overlap** (*boolean, (optional)*) – Allow Overlap, Don't correct overlap on new sequence strips
- **overlap_shuffle_override** (*boolean, (optional)*) – Override Overlap Shuffle Behavior, Use the overlap_mode tool settings to determine ho to shuffle overlapping strips
- **fit_method** (*enum in ['FIT', 'FILL', 'STRETCH', 'ORIGINAL'], (optional)*) –
  Fit Method, Scale fit method
  - `FIT` Scale to Fit – Scale image to fit within the canvas.
  - `FILL` Scale to Fill – Scale image to completely fill the canvas.
  - `STRETCH` Stretch to Fill – Stretch image to fill the canvas.
  - `ORIGINAL` Use Original Size – Keep image at its original size.
- **set_view_transform** (*boolean, (optional)*) – Set View Transform, Set appropriate view transform based on media color space
- **adjust_playback_rate** (*boolean, (optional)*) – Adjust Playback Rate, Play at normal speed regardless of scene FPS
- **sound** (*boolean, (optional)*) – Sound, Load sound with the movie
- **use_framerate** (*boolean, (optional)*) – Set Scene Frame Rate, Set frame rate of the current scene to the frame rate of the movie

bpy.ops.sequencer.**movieclip_strip_add(*, frame_start=0, channel=1, replace_sel=True, overlap=False, overlap_shuffle_override=False, clip='')**

Add a movieclip strip to the sequencer

PARAMETERS:

- **frame_start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the sequence strip
- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into
- **replace_sel** (*boolean, (optional)*) – Replace Selection, Deselect previously selected strips
- **overlap** (*boolean, (optional)*) – Allow Overlap, Don't correct overlap on new sequence strips
- **overlap_shuffle_override** (*boolean, (optional)*) – Override Overlap Shuffle Behavior, Use the overlap_mode tool settings to determine ho to shuffle overlapping strips
- **clip** (*enum in [], (optional)*) – Clip

bpy.ops.sequencer.**mute(*, unselected=False)**

Mute (un)selected strips

PARAMETERS:

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

bpy.ops.sequencer.**offset_clear()**

Clear strip offsets from the start and end frames

bpy.ops.sequencer.**paste(*, keep_offset=False)**

Paste strips from the internal clipboard

PARAMETERS:

keep_offset (*boolean, (optional)*) – Keep Offset, Keep strip offset relative to the current frame when pasting

bpy.ops.sequencer.**preview_duplicate_move(*, SEQUENCER_OT_duplicate=None, TRANSFORM_OT_translate=None)**

Duplicate selected strips and move them

PARAMETERS:

- **SEQUENCER_OT_duplicate** (`SEQUENCER_OT_duplicate`, (optional)) – Duplicate Strips, Duplicate the selected strips
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.sequencer.**reassign_inputs()**

Reassign the inputs for the effect strip

bpy.ops.sequencer.**rebuild_proxy()**

Rebuild all selected proxies and timecode indices

bpy.ops.sequencer.**refresh_all()**

Refresh the sequencer editor

bpy.ops.sequencer.**reload(*, adjust_length=False)**

Reload strips in the sequencer

PARAMETERS:

adjust_length (*boolean, (optional)*) – Adjust Length, Adjust length of strips to their data length

bpy.ops.sequencer.**rename_channel()**

Undocumented, consider contributing.

bpy.ops.sequencer.**rendersize()**

Set render size and aspect from active sequence

bpy.ops.sequencer.**retiming_add_freeze_frame_slide(*, SEQUENCER_OT_retiming_freeze_frame_add=None, TRANSFORM_OT_seq_slide=None)**

TRANSFORM_OT_seq_slide=None)

Add freeze frame and move it

PARAMETERS:

- **SEQUENCER_OT_retiming_freeze_frame_add** (`SEQUENCER_OT_retiming_freeze_frame_add`, (optional)) – Add Freeze Frame, Add freeze frame
- **TRANSFORM_OT_seq_slide** (`TRANSFORM_OT_seq_slide`, (optional)) – Sequence Slide, Slide a sequence strip in time

bpy.ops.sequencer.**retiming_add_transition_slide(*, SEQUENCER_OT_retiming_transition_add=None, TRANSFORM_OT_seq_slide=None)**

Add smooth transition between 2 retimed segments and change its duration

PARAMETERS:

- **SEQUENCER_OT_retiming_transition_add** (`SEQUENCER_OT_retiming_transition_add`, (optional)) – Add Speed Transition, Add smooth transition between 2 retimed segments
- **TRANSFORM_OT_seq_slide** (`TRANSFORM_OT_seq_slide`, (optional)) – Sequence Slide, Slide a sequence strip in time

bpy.ops.sequencer.**retiming_freeze_frame_add(*, duration=0)**

Add freeze frame

PARAMETERS:

duration (*int in [0, inf], (optional)*) – Duration, Duration of freeze frame segment

bpy.ops.sequencer.**retiming_key_add(*, timeline_frame=0)**

Add retiming Key

PARAMETERS:

timeline_frame (*int in [0, inf], (optional)*) – Timeline Frame, Frame where key will be added

bpy.ops.sequencer.**retiming_key_delete()**

Delete selected strips from the sequencer

bpy.ops.sequencer.**retiming_reset()**

Reset strip retiming

bpy.ops.sequencer.**retiming_segment_speed_set(*, speed=100.0, keep_retiming=True)**

Set speed of retimed segment

PARAMETERS:

- **speed** (*float in [0.001, inf], (optional)*) – Speed, New speed of retimed segment
- **keep_retiming** (*boolean, (optional)*) – Preserve Current Retiming, Keep speed of other segments unchanged, change strip length instead

bpy.ops.sequencer.**retiming_show()**

Show retiming keys in selected strips

bpy.ops.sequencer.**retiming_transition_add(*, duration=0)**

Add smooth transition between 2 retimed segments

PARAMETERS:

duration (*int in [0, inf], (optional)*) – Duration, Duration of freeze frame segment

bpy.ops.sequencer.**sample(*, size=1)**

Use mouse to sample color in current frame

PARAMETERS:

size (*int in [1, 128], (optional)*) – Sample Size bpy.ops.sequencer.**scene_frame_range_update()**

Update frame range of scene strip

bpy.ops.sequencer.**scene_strip_add(*, frame_start=0, channel=1, replace_sel=True, overlap=False, overlap_shuffle_override=False, scene='')**

Add a strip to the sequencer using a Blender scene as a source

PARAMETERS:

- **frame_start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the sequence strip
- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into
- **replace_sel** (*boolean, (optional)*) – Replace Selection, Deselect previously selected strips
- **overlap** (*boolean, (optional)*) – Allow Overlap, Don't correct overlap on new sequence strips
- **overlap_shuffle_override** (*boolean, (optional)*) – Override Overlap Shuffle Behavior, Use the overlap_mode tool settings to determine ho to shuffle overlapping strips
- **scene** (*enum in [], (optional)*) – Scene

bpy.ops.sequencer.**scene_strip_add_new(*, frame_start=0, channel=1, replace_sel=True, overlap=False, overlap_shuffle_override=False, type='NEW')**

Create a new Strip and assign a new Scene as source

PARAMETERS:

- **frame_start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the sequence strip
- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into
- **replace_sel** (*boolean, (optional)*) – Replace Selection, Deselect previously selected strips
- **overlap** (*boolean, (optional)*) – Allow Overlap, Don't correct overlap on new sequence strips
- **overlap_shuffle_override** (*boolean, (optional)*) – Override Overlap Shuffle Behavior, Use the overlap_mode tool settings to determine ho to shuffle overlapping strips
- **type** (*enum in ['NEW', 'EMPTY', 'LINK_COPY', 'FULL_COPY'], (optional)*) –

  Type

  - `NEW` New – Add new Strip with a new empty Scene with default settings.
  - `EMPTY` Copy Settings – Add a new Strip, with an empty scene, and copy settings from the current scene.
  - `LINK_COPY` Linked Copy – Add a Strip and link in the collections from the current scene (shallow copy).
  - `FULL_COPY` Full Copy – Add a Strip and make a full copy of the current scene.

bpy.ops.sequencer.**select(*, wait_to_deselect_others=False, mouse_x=0, mouse_y=0, extend=False, deselect=False, toggle=False, deselect_all=False, select_passthrough=False, center=False, linked_handle=False, linked_time=False, side_of_frame=False, ignore_connections=False)**

Select a strip (last selected becomes the "active strip")

PARAMETERS:

- **wait_to_deselect_others** (*boolean, (optional)*) – Wait to Deselect Others
- **mouse_x** (*int in [-inf, inf], (optional)*) – Mouse X
- **mouse_y** (*int in [-inf, inf], (optional)*) – Mouse Y
- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **deselect** (*boolean, (optional)*) – Deselect, Remove from selection
- **toggle** (*boolean, (optional)*) – Toggle Selection, Toggle the selection
- **deselect_all** (*boolean, (optional)*) – Deselect On Nothing, Deselect all when nothing under the cursor
- **select_passthrough** (*boolean, (optional)*) – Only Select Unselected, Ignore the select action when the element is already selected
- **center** (*boolean, (optional)*) – Center, Use the object center when selecting, in edit mode used to extend object selection
- **linked_handle** (*boolean, (optional)*) – Linked Handle, Select handles next to the active strip
- **linked_time** (*boolean, (optional)*) – Linked Time, Select other strips or handles at the same time, or all retiming keys after the current in retiming mode

- **side_of_frame** (*boolean, (optional)*) – Side of Frame, Select all strips on same side of the current frame as the mouse cursor
- **ignore_connections** (*boolean, (optional)*) – Ignore Connections, Select strips individually whether or not they are connected

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

Select or deselect all strips

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.sequencer.**select_box(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, mode='SET', tweak=False, include_handles=False, ignore_connections=False)**

Select strips using box selection

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **mode** (*enum in ['SET', 'ADD', 'SUB'], (optional)*) –
  Mode
  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.
- **tweak** (*boolean, (optional)*) – Tweak, Make box select pass through to sequence slide when the cursor is hovering on a strip
- **include_handles** (*boolean, (optional)*) – Select Handles, Select the strips and their handles
- **ignore_connections** (*boolean, (optional)*) – Ignore Connections, Select strips individually whether or not they are connected

bpy.ops.sequencer.**select_grouped(*, type='TYPE', extend=False, use_active_channel=False)**

Select all strips grouped by various properties

PARAMETERS:

- **type** (*enum in ['TYPE', 'TYPE_BASIC', 'TYPE_EFFECT', 'DATA', 'EFFECT', 'EFFECT_LINK', 'OVERLAP'], (optional)*) –
  Type
  - `TYPE` Type – Shared strip type.
  - `TYPE_BASIC` Global Type – All strips of same basic type (graphical or sound).
  - `TYPE_EFFECT` Effect Type – Shared strip effect type (if active strip is not an effect one, select all non-effect strips).
  - `DATA` Data – Shared data (scene, image, sound, etc.).
  - `EFFECT` Effect – Shared effects.
  - `EFFECT_LINK` Effect/Linked – Other strips affected by the active one (sharing some time, and below or effect-assigned).
  - `OVERLAP` Overlap – Overlapping time.
- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **use_active_channel** (*boolean, (optional)*) – Same Channel, Only consider strips on the same channel as the active one

bpy.ops.sequencer.**select_handle(*, wait_to_deselect_others=False, mouse_x=0, mouse_y=0, ignore_connections=False)**

Select strip handle

PARAMETERS:

- **wait_to_deselect_others** (*boolean, (optional)*) – Wait to Deselect Others
- **mouse_x** (*int in [-inf, inf], (optional)*) – Mouse X
- **mouse_y** (*int in [-inf, inf], (optional)*) – Mouse Y
- **ignore_connections** (*boolean, (optional)*) – Ignore Connections, Select strips individually whether or not they are connected

bpy.ops.sequencer.**select_handles(*, side='BOTH')**

Select gizmo handles on the sides of the selected strip

PARAMETERS:

side (*enum in ['LEFT', 'RIGHT', 'BOTH', 'LEFT_NEIGHBOR', 'RIGHT_NEIGHBOR', 'BOTH_NEIGHBORS'], (optional)*) – Side, The side of the handle that is selected

bpy.ops.sequencer.**select_less()**

Shrink the current selection of adjacent selected strips

bpy.ops.sequencer.**select_linked()**

Select all strips adjacent to the current selection

bpy.ops.sequencer.**select_linked_pick(*, extend=False)**

Select a chain of linked strips nearest to the mouse pointer

PARAMETERS:

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

bpy.ops.sequencer.**select_more()**

Select more strips adjacent to the current selection

bpy.ops.sequencer.**select_side(*, side='BOTH')**

Select strips on the nominated side of the selected strips

PARAMETERS:

side (*enum in ['MOUSE', 'LEFT', 'RIGHT', 'BOTH', 'NO_CHANGE'], (optional)*) – Side, The side to which the selection is applied

bpy.ops.sequencer.**select_side_of_frame(*, extend=False, side='LEFT')**

Select strips relative to the current frame

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend the selection
- **side** (*enum in ['LEFT', 'RIGHT', 'CURRENT'], (optional)*) –

  Side

  - `LEFT` Left – Select to the left of the current frame.
  - `RIGHT` Right – Select to the right of the current frame.
  - `CURRENT` Current Frame – Select intersecting with the current frame.

bpy.ops.sequencer.**set_range_to_strips(*, preview=False)**

Set the frame range to the selected strips start and end

PARAMETERS:

preview (*boolean, (optional)*) – Preview, Set the preview range instead

bpy.ops.sequencer.**slip(*, offset=0.0)**

Slip the contents of selected strips

PARAMETERS:

offset (*float in [-inf, inf], (optional)*) – Offset, Offset to the data of the strip

bpy.ops.sequencer.**snap(*, frame=0)**

Frame where selected strips will be snapped

PARAMETERS:

frame (*int in [-inf, inf], (optional)*) – Frame, Frame where selected strips will be snapped

bpy.ops.sequencer.**sound_strip_add(*, filepath='', directory='', files=None, check_existing=False, filter_blender=False, filter_backup=False, filter_image=False, filter_movie=False, filter_python=False, filter_font=False, filter_sound=True, 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, display_type='DEFAULT', sort_method='', frame_start=0, channel=1, replace_sel=True, overlap=False, overlap_shuffle_override=False, cache=False, mono=False)**

Add a sound strip to the sequencer

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
- **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 ['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..
- **frame_start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the sequence strip
- **channel** (*int in [1, 128], (optional)*) – Channel, Channel to place this strip into
- **replace_sel** (*boolean, (optional)*) – Replace Selection, Deselect previously selected strips
- **overlap** (*boolean, (optional)*) – Allow Overlap, Don't correct overlap on new sequence strips
- **overlap_shuffle_override** (*boolean, (optional)*) – Override Overlap Shuffle Behavior, Use the overlap_mode tool settings to determine ho to shuffle overlapping strips
- **cache** (*boolean, (optional)*) – Cache, Cache the sound in memory
- **mono** (*boolean, (optional)*) – Mono, Merge all the sound's channels into one

bpy.ops.sequencer.**split(*, frame=0, channel=0, type='SOFT', use_cursor_position=False, side='MOUSE', ignore_selection=False)**

Split the selected strips in two

PARAMETERS:

- **frame** (*int in [-inf, inf], (optional)*) – Frame, Frame where selected strips will be split
- **channel** (*int in [-inf, inf], (optional)*) – Channel, Channel in which strip will be cut
- **type** (*enum in ['SOFT', 'HARD'], (optional)*) – Type, The type of split operation to perform on strips
- **use_cursor_position** (*boolean, (optional)*) – Use Cursor Position, Split at position of the cursor instead of current frame
- **side** (*enum in ['MOUSE', 'LEFT', 'RIGHT', 'BOTH', 'NO_CHANGE'], (optional)*) – Side, The side that remains selected after splitting
- **ignore_selection** (*boolean, (optional)*) – Ignore Selection, Make cut even if strip is not selected preserving selection state after cut

bpy.ops.sequencer.**split_multicam(*, camera=1)**

Split multicam strip and select camera

PARAMETERS:

camera (*int in [1, 32], (optional)*) – Camera

FILE:

startup/bl_operators/sequencer.py:95

bpy.ops.sequencer.**strip_color_tag_set(*, color='NONE')**

Set a color tag for the selected strips

PARAMETERS:

color (enum in Strip Color Items, (optional)) – Color Tag

bpy.ops.sequencer.**strip_jump(*, next=True, center=True)**

Move frame to previous edit point

PARAMETERS:

- **next** (*boolean, (optional)*) – Next Strip
- **center** (*boolean, (optional)*) – Use Strip Center

bpy.ops.sequencer.**strip_modifier_add(*, type='')**

Add a modifier to the strip

PARAMETERS:

type (*enum in [], (optional)*) – Type bpy.ops.sequencer.**strip_modifier_copy(*, type='REPLACE')**

Copy modifiers of the active strip to all selected strips

PARAMETERS:

type (*enum in ['REPLACE', 'APPEND'], (optional)*) –

Type

- `REPLACE` Replace – Replace modifiers in destination.
- `APPEND` Append – Append active modifiers to selected strips.

bpy.ops.sequencer.**strip_modifier_equalizer_redefine(*, graphs='SIMPLE', name='Name')**

Redefine equalizer graphs

PARAMETERS:

- **graphs** (*enum in ['SIMPLE', 'DOUBLE', 'TRIPLE'], (optional)*) –

  Graphs, Number of graphs

  - `SIMPLE` Unique – One unique graphical definition.
  - `DOUBLE` Double – Graphical definition in 2 sections.
  - `TRIPLE` Triplet – Graphical definition in 3 sections.
- **name** (*string, (optional, never None)*) – Name, Name of modifier to redefine

bpy.ops.sequencer.**strip_modifier_move(*, name='Name', direction='UP')**

Move modifier up and down in the stack

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of modifier to remove
- **direction** (*enum in ['UP', 'DOWN'], (optional)*) –

  Type

  - `UP` Up – Move modifier up in the stack.
  - `DOWN` Down – Move modifier down in the stack.

bpy.ops.sequencer.**strip_modifier_remove(*, name='Name')**

Remove a modifier from the strip

PARAMETERS:

name (*string, (optional, never None)*) – Name, Name of modifier to remove

bpy.ops.sequencer.**strip_transform_clear(*, property='ALL')**

Reset image transformation to default value

PARAMETERS:

property (*enum in ['POSITION', 'SCALE', 'ROTATION', 'ALL'], (optional)*) –

Property, Strip transform property to be reset

- `POSITION` Position – Reset strip transform location.
- `SCALE` Scale – Reset strip transform scale.
- `ROTATION` Rotation – Reset strip transform rotation.
- `ALL` All – Reset strip transform location, scale and rotation.

bpy.ops.sequencer.**strip_transform_fit(*, fit_method='FIT')**

Undocumented, consider contributing.

PARAMETERS:

fit_method (*enum in ['FIT', 'FILL', 'STRETCH'], (optional)*) –

fit_method (*enum in ['FIT', 'FILL', 'STRETCH'], (optional)*) –

Fit Method, Scale fit fit_method

- `FIT` Scale to Fit – Scale image so fits in preview.
- `FILL` Scale to Fill – Scale image so it fills preview completely.
- `STRETCH` Stretch to Fill – Stretch image so it fills preview.

bpy.ops.sequencer.**swap(*, side='RIGHT')**

Swap active strip with strip to the right or left

PARAMETERS:

side (*enum in ['LEFT', 'RIGHT'], (optional)*) – Side, Side of the strip to swap

bpy.ops.sequencer.**swap_data()**

Swap 2 sequencer strips

bpy.ops.sequencer.**swap_inputs()**

Swap the two inputs of the effect strip

bpy.ops.sequencer.**text_cursor_move(*, type='LINE_BEGIN', select_text=False)**

Move cursor in text

PARAMETERS:

- **type** (*enum in ['LINE_BEGIN', 'LINE_END', 'TEXT_BEGIN', 'TEXT_END', 'PREVIOUS_CHARACTER', 'NEXT_CHARACTER', 'PREVIOUS_WORD', 'NEXT_WORD', 'PREVIOUS_LINE', 'NEXT_LINE'], (optional)*) – Type, Where to move cursor to, to make a selection
- **select_text** (*boolean, (optional)*) – Select Text, Select text while moving cursor

bpy.ops.sequencer.**text_cursor_set(*, select_text=False)**

Set cursor position in text

PARAMETERS:

select_text (*boolean, (optional)*) – Select Text, Select text while moving cursor

bpy.ops.sequencer.**text_delete(*, type='NEXT_OR_SELECTION')**

Delete text at cursor position

PARAMETERS:

type (*enum in ['NEXT_OR_SELECTION', 'PREVIOUS_OR_SELECTION'], (optional)*) – Type, Which part of the text to delete

bpy.ops.sequencer.**text_deselect_all()**

Deselect all characters

bpy.ops.sequencer.**text_edit_copy()**

Copy text to clipboard

bpy.ops.sequencer.**text_edit_cut()**

Cut text to clipboard

bpy.ops.sequencer.**text_edit_mode_toggle()**

Toggle text editing

bpy.ops.sequencer.**text_edit_paste()**

Paste text to clipboard

bpy.ops.sequencer.**text_insert(*, string='')**

Insert text at cursor position

Insert text at cursor position

PARAMETERS:

string (*string, (optional, never None)*) – String, String to be inserted at cursor position

bpy.ops.sequencer.**text_line_break()**

Insert line break at cursor position

bpy.ops.sequencer.**text_select_all()**

Select all characters

bpy.ops.sequencer.**unlock()**

Unlock strips so they can be transformed

bpy.ops.sequencer.**unmute(*, unselected=False)**

Unmute (un)selected strips

PARAMETERS:

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

bpy.ops.sequencer.**view_all()**

View all the strips in the sequencer

bpy.ops.sequencer.**view_all_preview()**

Zoom preview to fit in the area

bpy.ops.sequencer.**view_frame()**

Move the view to the current frame

bpy.ops.sequencer.**view_ghost_border(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True)**

Set the boundaries of the border used for offset view

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.sequencer.**view_selected()**

Zoom the sequencer on the selected strips

bpy.ops.sequencer.**view_zoom_ratio(*, ratio=1.0)**

Change zoom ratio of sequencer preview

PARAMETERS:

ratio (*float in [-inf, inf], (optional)*) – Ratio, Zoom ratio, 1.0 is 1:1, higher is zoomed in, lower is zoomed out

Skip to content

Sound Operators

bpy.ops.sound.**bake_animation()**

Update the audio animation cache

bpy.ops.sound.**mixdown(*, filepath='', check_existing=True, filter_blender=False, filter_backup=False, filter_image=False, filter_movie=False, filter_python=False, filter_font=False, filter_sound=True, 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, display_type='DEFAULT', sort_method='', accuracy=102 container='FLAC', codec='FLAC', channels='STEREO', format='S16', mixrate=48000, bitrate=192, split_channels=False)**

Mix the scene's audio to a sound file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **accuracy** (*int in [1, inf], (optional)*) – Accuracy, Sample accuracy, important for animation data (the lower the value, the more accurate)
- **container** (*enum in ['AAC', 'AC3', 'FLAC', 'MATROSKA', 'MP2', 'MP3', 'OGG', 'WAV'], (optional)*) – Container, File format
  - `AAC` AAC – Advanced Audio Coding.
  - `AC3` AC3 – Dolby Digital ATRAC 3.
  - `FLAC` FLAC – Free Lossless Audio Codec.
  - `MATROSKA` MKV – Matroska.

    - `MP2` MP2 – MPEG-1 Audio Layer II.
    - `MP3` MP3 – MPEG-2 Audio Layer III.
    - `OGG` OGG – Xiph.Org Ogg Container.
    - `WAV` WAV – Waveform Audio File Format.
- **codec** (*enum in ['AAC', 'AC3', 'FLAC', 'MP2', 'MP3', 'PCM', 'VORBIS'], (optional)*) –
  Codec, Audio Codec
    - `AAC` AAC – Advanced Audio Coding.
    - `AC3` AC3 – Dolby Digital ATRAC 3.
    - `FLAC` FLAC – Free Lossless Audio Codec.
    - `MP2` MP2 – MPEG-1 Audio Layer II.
    - `MP3` MP3 – MPEG-2 Audio Layer III.
    - `PCM` PCM – Pulse Code Modulation (RAW).
    - `VORBIS` Vorbis – Xiph.Org Vorbis Codec.
- **channels** (*enum in ['MONO', 'STEREO', 'STEREO_LFE', 'SURROUND4', 'SURROUND5', 'SURROUND51', 'SURROUND61', 'SURROUND71'], (optional)*) –
  Channels, Audio channel count
    - `MONO` Mono – Single audio channel.
    - `STEREO` Stereo – Stereo audio channels.
    - `STEREO_LFE` Stereo LFE – Stereo with LFE channel.
    - `SURROUND4` 4 Channels – 4 channel surround sound.
    - `SURROUND5` 5 Channels – 5 channel surround sound.
    - `SURROUND51` 5.1 Surround – 5.1 surround sound.
    - `SURROUND61` 6.1 Surround – 6.1 surround sound.
    - `SURROUND71` 7.1 Surround – 7.1 surround sound.
- **format** (*enum in ['U8', 'S16', 'S24', 'S32', 'F32', 'F64'], (optional)*) –
  Format, Sample format
    - `U8` U8 – 8-bit unsigned.
    - `S16` S16 – 16-bit signed.
    - `S24` S24 – 24-bit signed.
    - `S32` S32 – 32-bit signed.
    - `F32` F32 – 32-bit floating-point.
    - `F64` F64 – 64-bit floating-point.
- **mixrate** (*int in [8000, 192000], (optional)*) – Sample Rate, Sample rate in samples/s
- **bitrate** (*int in [32, 512], (optional)*) – Bitrate, Bitrate in kbit/s
- **split_channels** (*boolean, (optional)*) – Split channels, Each channel will be rendered into a mono file

bpy.ops.sound.**open(*, filepath='', hide_props_region=True, check_existing=False, filter_blender=False, filter_backup=False, filter_image=False, filter_movie=True, filter_python=False, filter_font=False, filter_sound=True, 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='', cache=False, mono=False)**

Load a sound 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 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 [], (optional)*) – File sorting mode
- **cache** (*boolean, (optional)*) – Cache, Cache the sound in memory
- **mono** (*boolean, (optional)*) – Mono, Merge all the sound’s channels into one

bpy.ops.sound.**open_mono(*, filepath='', hide_props_region=True, check_existing=False, filter_blender=False, filter_backup=False, filter_image=False, filter_movie=True, filter_python=False, filter_font=False, filter_sound=True, 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='', cache=False, mono=True)**

Load a sound file as mono

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 [], (optional)*) – File sorting mode
- **cache** (*boolean, (optional)*) – Cache, Cache the sound in memory
- **mono** (*boolean, (optional)*) – Mono, Mixdown the sound to mono

bpy.ops.sound.**pack()**

Pack the sound into the current blend file

bpy.ops.sound.**unpack(*, method='USE_LOCAL', id='')**

Unpack the sound to the samples filename

PARAMETERS:

- **method** (enum in Unpack Method Items, (optional)) – Method, How to unpack
- **id** (*string, (optional, never None)*) – Sound Name, Sound data-block name to unpack

bpy.ops.sound.**update_animation_flags()**

Update animation flags

Spreadsheet Operators

bpy.ops.spreadsheet.**add_row_filter_rule()**

Add a filter to remove rows from the displayed data

bpy.ops.spreadsheet.**change_spreadsheet_data_source(*, component_type=0, attribute_domain_type=0)**

Change visible data source in the spreadsheet

PARAMETERS:

- **component_type** (*int in [0, 32767], (optional)*) – Component Type
- **attribute_domain_type** (*int in [0, 32767], (optional)*) – Attribute Domain Type

bpy.ops.spreadsheet.**remove_row_filter_rule(*, index=0)**

Remove a row filter from the rules

PARAMETERS:

index (*int in [0, inf], (optional)*) – Index

bpy.ops.spreadsheet.**toggle_pin()**

Turn on or off pinning

FILE:

startup/bl_operators/spreadsheet.py:21

Skip to content

Surface Operators

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

Construct a Nurbs surface Circle

PARAMETERS:

- **radius** (*float in [0, inf], (optional)*) – Radius
- **enter_editmode** (*boolean, (optional)*) – Enter Edit Mode, Enter edit mode when adding this object
- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –

  Align, The alignment of the new object

  - `WORLD` World – Align the new object to the world.
  - `VIEW` View – Align the new object to the view.
  - `CURSOR` 3D Cursor – Use the 3D cursor orientation for the new object.
- **location** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Location, Location for the newly added object
- **rotation** (`mathutils.Euler` rotation of 3 items in [-inf, inf], (optional)) – Rotation, Rotation for the newly added object
- **scale** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Scale, Scale for the newly added object

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

Construct a Nurbs surface Curve

PARAMETERS:

- **radius** (*float in [0, inf], (optional)*) – Radius
- **enter_editmode** (*boolean, (optional)*) – Enter Edit Mode, Enter edit mode when adding this object
- **align** (*enum in ['WORLD', 'VIEW', 'CURSOR'], (optional)*) –

  Align, The alignment of the new object

  - `WORLD` World – Align the new object to the world.
  - `VIEW` View – Align the new object to the view.
  - `CURSOR` 3D Cursor – Use the 3D cursor orientation for the new object.
- **location** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Location, Location for the newly added object
- **rotation** (`mathutils.Euler` rotation of 3 items in [-inf, inf], (optional)) – Rotation, Rotation for the newly added object
- **scale** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Scale, Scale for the newly added object

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

Construct a Nurbs surface Cylinder

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.surface.**primitive_nurbs_surface_sphere_add(*, radius=1.0, enter_editmode=False, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Construct a Nurbs surface Sphere

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.surface.**primitive_nurbs_surface_surface_add(*, radius=1.0, enter_editmode=False, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Construct a Nurbs surface Patch

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.surface.**primitive_nurbs_surface_torus_add(*, radius=1.0, enter_editmode=False, align='WORLD', location=(0.0, 0.0, 0.0), rotation=(0.0, 0.0, 0.0), scale=(0.0, 0.0, 0.0))**

Construct a Nurbs surface Torus

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

Skip to content

Text Operators

bpy.ops.text.**autocomplete()**

Show a list of used text in the open document

bpy.ops.text.**comment_toggle(*, type='TOGGLE')**

Undocumented, consider contributing.

PARAMETERS:

type (*enum in ['TOGGLE', 'COMMENT', 'UNCOMMENT'], (optional)*) – Type, Add or remove comments

bpy.ops.text.**convert_whitespace(*, type='SPACES')**

Convert whitespaces by type

PARAMETERS:

type (*enum in ['SPACES', 'TABS'], (optional)*) – Type, Type of whitespace to convert to

bpy.ops.text.**copy()**

Copy selected text to clipboard

bpy.ops.text.**cursor_set(*, x=0, y=0)**

Set cursor position

PARAMETERS:

- **x** (*int in [-inf, inf], (optional)*) – X
- **y** (*int in [-inf, inf], (optional)*) – Y

bpy.ops.text.**cut()**

Cut selected text to clipboard

bpy.ops.text.**delete(*, type='NEXT_CHARACTER')**

Delete text by cursor position

PARAMETERS:

type (*enum in ['NEXT_CHARACTER', 'PREVIOUS_CHARACTER', 'NEXT_WORD', 'PREVIOUS_WORD'], (optional)*) – Type, Whic
part of the text to delete

bpy.ops.text.**duplicate_line()**

Duplicate the current line

bpy.ops.text.**find()**

Find specified text

bpy.ops.text.**find_set_selected()**

Find specified text and set as selected

bpy.ops.text.**indent()**

Indent selected text

bpy.ops.text.**indent_or_autocomplete()**

Indent selected text or autocomplete

bpy.ops.text.**insert(*, text='')**

Insert text at cursor position

PARAMETERS:

text (*string, (optional, never None)*) – Text, Text to insert at the cursor position

bpy.ops.text.**jump(*, line=1)**

Jump cursor to line

PARAMETERS:

line (*int in [1, inf], (optional)*) – Line, Line number to jump to

bpy.ops.text.**jump_to_file_at_point(*, filepath='', line=0, column=0)**

Jump to a file for the text editor

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – Filepath
- **line** (*int in [0, inf], (optional)*) – Line, Line to jump to
- **column** (*int in [0, inf], (optional)*) – Column, Column to jump to

bpy.ops.text.**line_break()**

Insert line break at cursor position

bpy.ops.text.**line_number()**

The current line number

bpy.ops.text.**make_internal()**

Make active text file internal

bpy.ops.text.**move(*, type='LINE_BEGIN')**

Move cursor to position type

PARAMETERS:

type (*enum in ['LINE_BEGIN', 'LINE_END', 'FILE_TOP', 'FILE_BOTTOM', 'PREVIOUS_CHARACTER', 'NEXT_CHARACTER', 'PREVIOUS_WORD', 'NEXT_WORD', 'PREVIOUS_LINE', 'NEXT_LINE', 'PREVIOUS_PAGE', 'NEXT_PAGE'], (optional)*) – Type, Where to move cursor to

bpy.ops.text.**move_lines(*, direction='DOWN')**

Move the currently selected line(s) up/down

PARAMETERS:

direction (*enum in ['UP', 'DOWN'], (optional)*) – Direction

bpy.ops.text.**move_select(*, type='LINE_BEGIN')**

Move the cursor while selecting

PARAMETERS:

type (*enum in ['LINE_BEGIN', 'LINE_END', 'FILE_TOP', 'FILE_BOTTOM', 'PREVIOUS_CHARACTER', 'NEXT_CHARACTER', 'PREVIOUS_WORD', 'NEXT_WORD', 'PREVIOUS_LINE', 'NEXT_LINE', 'PREVIOUS_PAGE', 'NEXT_PAGE'], (optional)*) – Type, Where to move cursor to, to make a selection

bpy.ops.text.**new()**

Create a new text data-block

bpy.ops.text.**open(*, filepath='', hide_props_region=True, check_existing=False, filter_blender=False, filter_backup=False, filter_image=False, filter_movie=False, filter_python=True, filter_font=False, filter_sound=False, filter_text=True, 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, display_type='DEFAULT', sort_method='', internal=False)**

Open a new text data-block

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
- **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..
- **internal** (*boolean, (optional)*) – Make Internal, Make text file internal after loading

bpy.ops.text.**overwrite_toggle()**

Toggle overwrite while typing

bpy.ops.text.**paste(*, selection=False)**

Paste text from clipboard

PARAMETERS:

- **selection** (*boolean, (optional)*) – Selection, Paste text selected elsewhere rather than copied (X11/Wayland only)

selection (*boolean, (optional)*) – Selection, Paste text selected elsewhere rather than copied (X11/Wayland only)

bpy.ops.text.**refresh_pyconstraints()**

Refresh all pyconstraints

bpy.ops.text.**reload()**

Reload active text data-block from its file

bpy.ops.text.**replace(*, all=False)**

Replace text with the specified text

PARAMETERS:

all (*boolean, (optional)*) – Replace All, Replace all occurrences

bpy.ops.text.**replace_set_selected()**

Replace text with specified text and set as selected

bpy.ops.text.**resolve_conflict(*, resolution='IGNORE')**

When external text is out of sync, resolve the conflict

PARAMETERS:

resolution (*enum in ['IGNORE', 'RELOAD', 'SAVE', 'MAKE_INTERNAL'], (optional)*) – Resolution, How to solve conflict due to differences in internal and external text

bpy.ops.text.**run_script()**

Run active script

bpy.ops.text.**save()**

Save active text data-block

bpy.ops.text.**save_as(*, filepath='', hide_props_region=True, check_existing=True, filter_blender=False, filter_backup=False, filter_image=False, filter_movie=False, filter_python=True, filter_font=False, filter_sound=False, filter_text=True, 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, display_type='DEFAULT', sort_method='')**

Save active text file with options

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
- **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

bpy.ops.text.**scroll(*, lines=1)**

Undocumented, consider contributing.

PARAMETERS:

lines (*int in [-inf, inf], (optional)*) – Lines, Number of lines to scroll

bpy.ops.text.**scroll_bar(*, lines=1)**

Undocumented, consider contributing.

PARAMETERS:

lines (*int in [-inf, inf], (optional)*) – Lines, Number of lines to scroll

bpy.ops.text.**select_all()**

Select all text

bpy.ops.text.**select_line()**

Select text by line

bpy.ops.text.**select_word()**

Select word under cursor

bpy.ops.text.**selection_set()**

Set text selection

bpy.ops.text.**start_find()**

Start searching text

bpy.ops.text.**to_3d_object(*, split_lines=False)**

Create 3D text object from active text data-block

PARAMETERS:

split_lines (*boolean, (optional)*) – Split Lines, Create one object per line in the text

bpy.ops.text.**unindent()**

Unindent selected text

bpy.ops.text.**unlink()**

Unlink active text data-block

Text Editor Operators

bpy.ops.text_editor.**preset_add(*, name='', remove_name=False, remove_active=False)**

Add or remove a Text Editor Preset

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the preset, used to make the path name
- **remove_name** (*boolean, (optional)*) – remove_name
- **remove_active** (*boolean, (optional)*) – remove_active

FILE:

startup/bl_operators/presets.py:119

Texture Operators

bpy.ops.texture.**new()**

Add a new texture

bpy.ops.texture.**slot_copy()**

Copy the material texture settings and nodes

bpy.ops.texture.**slot_move(*, type='UP')**

Move texture slots up and down

PARAMETERS:

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

bpy.ops.texture.**slot_paste()**

Copy the texture settings and nodes

Skip to content

Transform Operators

bpy.ops.transform.**bbone_resize(*, value=(1.0, 1.0, 1.0), orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', constraint_axis=(False, False, False), mirror=False, release_confirm=False, use_accurate=False)**

Scale selected bendy bones display size

PARAMETERS:

- **value** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Display Size
- **orient_type** (*enum in [], (optional)*) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (*enum in [], (optional)*) – Matrix Orientation
- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**bend(*, value=0.0, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, snap=False, gpencil_strokes=False, center_override=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False)**

Bend selected items between the 3D cursor and the mouse

PARAMETERS:

- **value** (*float array of 1 items in [-inf, inf], (optional)*) – Angle
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **center_override** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center Override, Force using this center value (when set)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**create_orientation(*, name='', use_view=False, use=False, overwrite=False)**

Create transformation orientation from selection

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the new custom orientation
- **use_view** (*boolean, (optional)*) – Use View, Use the current view instead of the active object to create the new orientation
- **use** (*boolean, (optional)*) – Use After Creation, Select orientation after its creation
- **overwrite** (*boolean, (optional)*) – Overwrite Previous, Overwrite previously created orientation with same name

bpy.ops.transform.**delete_orientation()**

Delete transformation orientation

bpy.ops.transform.**edge_bevelweight(*, value=0.0, snap=False, release_confirm=False, use_accurate=False)**

Change the bevel weight of edges

PARAMETERS:

- **value** (*float in [-1, 1], (optional)*) – Factor

- **value** (*float in [-1, 1], (optional)*) – Factor
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**edge_crease(*, value=0.0, snap=False, release_confirm=False, use_accurate=False)**

Change the crease of edges

PARAMETERS:

- **value** (*float in [-1, 1], (optional)*) – Factor
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**edge_slide(*, value=0.0, single_side=False, use_even=False, flipped=False, use_clamp=True, mirror=False, snap=False snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False, snap_point=(0.0, 0.0, 0.0), correct_uv=True, release_confirm=False, use_accurate=False)**

Slide an edge loop along a mesh

PARAMETERS:

- **value** (*float in [-10, 10], (optional)*) – Factor
- **single_side** (*boolean, (optional)*) – Single Side
- **use_even** (*boolean, (optional)*) – Even, Make the edge loop match the shape of the adjacent edge loop
- **flipped** (*boolean, (optional)*) – Flipped, When Even mode is active, flips between the two adjacent edge loops
- **use_clamp** (*boolean, (optional)*) – Clamp, Clamp within the edge extents
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **snap_elements** (enum set in Snap Element Items, (optional)) – Snap to Elements
- **use_snap_project** (*boolean, (optional)*) – Project Individual Elements
- **snap_target** (enum in Snap Source Items, (optional)) – Snap Base, Point on source that will snap to target
- **use_snap_self** (*boolean, (optional)*) – Target: Include Active
- **use_snap_edit** (*boolean, (optional)*) – Target: Include Edit
- **use_snap_nonedit** (*boolean, (optional)*) – Target: Include Non-Edited
- **use_snap_selectable** (*boolean, (optional)*) – Target: Exclude Non-Selectable
- **snap_point** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Point
- **correct_uv** (*boolean, (optional)*) – Correct UVs, Correct UV coordinates when transforming
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**from_gizmo()**

Transform selected items by mode type

bpy.ops.transform.**mirror(*, orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', constraint_axis=(False, False, False), gpencil_strokes=False, center_override=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False)**

Mirror selected items around one or more axes

PARAMETERS:

- **orient_type** (*enum in [], (optional)*) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (*enum in [], (optional)*) – Matrix Orientation
- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis

- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **center_override** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center Override, Force using this center value (when set)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**push_pull(*, value=0.0, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, snap=False, center_override=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False)**

Push/Pull selected items

PARAMETERS:

- **value** (*float in [-inf, inf], (optional)*) – Distance
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **center_override** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center Override, Force using this center value (when set)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**resize(*, value=(1.0, 1.0, 1.0), mouse_dir_constraint=(0.0, 0.0, 0.0), orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0. (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', constraint_axis=(False, False, False), mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, snap=False, snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False, snap_point=(0.0, 0.0, 0.0), gpencil_strokes=False, texture_space=False, remove_on_cancel=False, use_duplicated_keyframes=False, center_override=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False)**

Scale (resize) selected items

PARAMETERS:

- **value** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Scale
- **mouse_dir_constraint** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Mouse Directional Constraint
- **orient_type** (*enum in [], (optional)*) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (*enum in [], (optional)*) – Matrix Orientation
- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **snap_elements** (enum set in Snap Element Items, (optional)) – Snap to Elements
- **use_snap_project** (*boolean, (optional)*) – Project Individual Elements
- **snap_target** (enum in Snap Source Items, (optional)) – Snap Base, Point on source that will snap to target
- **use_snap_self** (*boolean, (optional)*) – Target: Include Active
- **use_snap_edit** (*boolean, (optional)*) – Target: Include Edit

- **use_snap_nonedit** (*boolean, (optional)*) – Target: Include Non-Edited
- **use_snap_selectable** (*boolean, (optional)*) – Target: Exclude Non-Selectable
- **snap_point** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Point
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **texture_space** (*boolean, (optional)*) – Edit Texture Space, Edit object data texture space
- **remove_on_cancel** (*boolean, (optional)*) – Remove on Cancel, Remove elements on cancel
- **use_duplicated_keyframes** (*boolean, (optional)*) – Duplicated Keyframes, Transform duplicated keyframes
- **center_override** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center Override, Force using this center value (when set)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**rotate(*, value=0.0, orient_axis='Z', orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', constraint_axis=(False, False, False), mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=Fals snap=False, snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False, snap_point=(0.0, 0.0, 0.0), gpencil_strokes=False, center_override=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False)**

Rotate selected items

PARAMETERS:

- **value** (*float in [-inf, inf], (optional)*) – Angle
- **orient_axis** (enum in Axis Xyz Items, (optional)) – Axis
- **orient_type** (*enum in [], (optional)*) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (*enum in [], (optional)*) – Matrix Orientation
- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **snap_elements** (enum set in Snap Element Items, (optional)) – Snap to Elements
- **use_snap_project** (*boolean, (optional)*) – Project Individual Elements
- **snap_target** (enum in Snap Source Items, (optional)) – Snap Base, Point on source that will snap to target
- **use_snap_self** (*boolean, (optional)*) – Target: Include Active
- **use_snap_edit** (*boolean, (optional)*) – Target: Include Edit
- **use_snap_nonedit** (*boolean, (optional)*) – Target: Include Non-Edited
- **use_snap_selectable** (*boolean, (optional)*) – Target: Exclude Non-Selectable
- **snap_point** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Point
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **center_override** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center Override, Force using this center value (when set)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**rotate_normal(*, value=0.0, orient_axis='Z', orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', constraint_axis=(False, False, False), mirror=False, release_confirm=False, use_accurate=False)**

Rotate split normal of selected items

PARAMETERS:

- **value** (*float in [-inf, inf], (optional)*) – Angle
- **orient_axis** (enum in Axis Xyz Items, (optional)) – Axis
- **orient_type** (*enum in [], (optional)*) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (*enum in [], (optional)*) – Matrix Orientation
- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**select_orientation(*, orientation='GLOBAL')**

Select transformation orientation

PARAMETERS:

orientation (*enum in [], (optional)*) – Orientation, Transformation orientation

bpy.ops.transform.**seq_slide(*, value=(0.0, 0.0), use_restore_handle_selection=False, snap=False, view2d_edge_pan=False, release_confirm=False, use_accurate=False)**

Slide a sequence strip in time

PARAMETERS:

- **value** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Offset
- **use_restore_handle_selection** (*boolean, (optional)*) – Restore Handle Selection, Restore handle selection after tweaking
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **view2d_edge_pan** (*boolean, (optional)*) – Edge Pan, Enable edge panning in 2D view
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**shear(*, value=0.0, orient_axis='Z', orient_axis_ortho='X', orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=Fals snap=False, gpencil_strokes=False, release_confirm=False, use_accurate=False)**

Shear selected items along the given axis

PARAMETERS:

- **value** (*float in [-inf, inf], (optional)*) – Offset
- **orient_axis** (enum in Axis Xyz Items, (optional)) – Axis
- **orient_axis_ortho** (enum in Axis Xyz Items, (optional)) – Axis Ortho
- **orient_type** (*enum in [], (optional)*) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (*enum in [], (optional)*) – Matrix Orientation
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**shrink_fatten(*, value=0.0, use_even_offset=False, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=Fals snap=False, release_confirm=False, use_accurate=False)**

Shrink/fatten selected vertices along normals

PARAMETERS:

- **value** (*float in [-inf, inf], (optional)*) – Offset
- **use_even_offset** (*boolean, (optional)*) – Offset Even, Scale the offset to give more even thickness
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**skin_resize(*, value=(1.0, 1.0, 1.0), orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', constraint_axis=(False, False, False), mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=Fals snap=False, snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False, snap_point=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False)**

Scale selected vertices' skin radii

PARAMETERS:

- **value** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Scale
- **orient_type** (*enum in [], (optional)*) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (*enum in [], (optional)*) – Matrix Orientation
- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **snap_elements** (enum set in Snap Element Items, (optional)) – Snap to Elements
- **use_snap_project** (*boolean, (optional)*) – Project Individual Elements
- **snap_target** (enum in Snap Source Items, (optional)) – Snap Base, Point on source that will snap to target
- **use_snap_self** (*boolean, (optional)*) – Target: Include Active
- **use_snap_edit** (*boolean, (optional)*) – Target: Include Edit
- **use_snap_nonedit** (*boolean, (optional)*) – Target: Include Non-Edited
- **use_snap_selectable** (*boolean, (optional)*) – Target: Exclude Non-Selectable
- **snap_point** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Point
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**tilt(*, value=0.0, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH',**

proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, snap=False, release_confirm=False, use_accurate=False)

Tilt selected control vertices of 3D curve

PARAMETERS:

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

bpy.ops.transform.**tosphere(*, value=0.0, mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, snap=False, gpencil_strokes=False, center_override=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False)**

Move selected items outward in a spherical shape around geometric center

PARAMETERS:

- **value** (*float in [0, 1], (optional)*) – Factor
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **center_override** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center Override, Force using this center value (when set)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**trackball(*, value=(0.0, 0.0), mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, snap=False, gpencil_strokes=False, center_override=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False)**

Trackball style rotation of selected items

PARAMETERS:

- **value** (*float array of 2 items in [-inf, inf], (optional)*) – Angle
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **center_override** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center Override, Force using this center value (when set)

- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**transform(*, mode='TRANSLATION', value=(0.0, 0.0, 0.0, 0.0), orient_axis='Z', orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', constraint_axis=(False, False, False), mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=False, snap=False, snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False, snap_point=(0.0, 0.0, 0.0), snap_align=False, snap_normal=(0.0, 0.0, 0.0), gpencil_strokes=False, texture_space=False, remove_on_cancel=False, use_duplicated_keyframes=False, center_override=(0.0, 0.0, 0.0), release_confirm=False, use_accurate=False, use_automerge_and_split=False)**

Transform selected items by mode type

PARAMETERS:

- **mode** (enum in Transform Mode Type Items, (optional)) – Mode
- **value** (`mathutils.Vector` of 4 items in [-inf, inf], (optional)) – Values
- **orient_axis** (enum in Axis Xyz Items, (optional)) – Axis
- **orient_type** (enum in Transform Orientation Items, (optional)) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (enum in Transform Orientation Items, (optional)) – Matrix Orientation
- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **snap_elements** (enum set in Snap Element Items, (optional)) – Snap to Elements
- **use_snap_project** (*boolean, (optional)*) – Project Individual Elements
- **snap_target** (enum in Snap Source Items, (optional)) – Snap Base, Point on source that will snap to target
- **use_snap_self** (*boolean, (optional)*) – Target: Include Active
- **use_snap_edit** (*boolean, (optional)*) – Target: Include Edit
- **use_snap_nonedit** (*boolean, (optional)*) – Target: Include Non-Edited
- **use_snap_selectable** (*boolean, (optional)*) – Target: Exclude Non-Selectable
- **snap_point** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Point
- **snap_align** (*boolean, (optional)*) – Align with Point Normal
- **snap_normal** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Normal
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **texture_space** (*boolean, (optional)*) – Edit Texture Space, Edit object data texture space
- **remove_on_cancel** (*boolean, (optional)*) – Remove on Cancel, Remove elements on cancel
- **use_duplicated_keyframes** (*boolean, (optional)*) – Duplicated Keyframes, Transform duplicated keyframes
- **center_override** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center Override, Force using this center value (when set)
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation
- **use_automerge_and_split** (*boolean, (optional)*) – Auto Merge & Split, Forces the use of Auto Merge and Split

bpy.ops.transform.**translate(*, value=(0.0, 0.0, 0.0), orient_type='GLOBAL', orient_matrix=((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), orient_matrix_type='GLOBAL', constraint_axis=(False, False, False), mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1.0, use_proportional_connected=False, use_proportional_projected=Fals snap=False, snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False, snap_point=(0.0, 0.0, 0.0), snap_align=False,**

snap_normal=(0.0, 0.0, 0.0), gpencil_strokes=False, cursor_transform=False, texture_space=False, remove_on_cancel=False, use_duplicated_keyframes=False, view2d_edge_pan=False, release_confirm=False, use_accurate=False, use_automerge_and_split=False)

Move selected items

PARAMETERS:

- **value** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Move
- **orient_type** (enum in Transform Orientation Items, (optional)) – Orientation, Transformation orientation
- **orient_matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (optional)) – Matrix
- **orient_matrix_type** (enum in Transform Orientation Items, (optional)) – Matrix Orientation
- **constraint_axis** (*boolean array of 3 items, (optional)*) – Constraint Axis
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **use_proportional_edit** (*boolean, (optional)*) – Proportional Editing
- **proportional_edit_falloff** (enum in Proportional Falloff Items, (optional)) – Proportional Falloff, Falloff type for proportional editing mode
- **proportional_size** (*float in [1e-06, inf], (optional)*) – Proportional Size
- **use_proportional_connected** (*boolean, (optional)*) – Connected
- **use_proportional_projected** (*boolean, (optional)*) – Projected (2D)
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **snap_elements** (enum set in Snap Element Items, (optional)) – Snap to Elements
- **use_snap_project** (*boolean, (optional)*) – Project Individual Elements
- **snap_target** (enum in Snap Source Items, (optional)) – Snap Base, Point on source that will snap to target
- **use_snap_self** (*boolean, (optional)*) – Target: Include Active
- **use_snap_edit** (*boolean, (optional)*) – Target: Include Edit
- **use_snap_nonedit** (*boolean, (optional)*) – Target: Include Non-Edited
- **use_snap_selectable** (*boolean, (optional)*) – Target: Exclude Non-Selectable
- **snap_point** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Point
- **snap_align** (*boolean, (optional)*) – Align with Point Normal
- **snap_normal** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Normal
- **gpencil_strokes** (*boolean, (optional)*) – Edit Grease Pencil, Edit selected Grease Pencil strokes
- **cursor_transform** (*boolean, (optional)*) – Transform Cursor
- **texture_space** (*boolean, (optional)*) – Edit Texture Space, Edit object data texture space
- **remove_on_cancel** (*boolean, (optional)*) – Remove on Cancel, Remove elements on cancel
- **use_duplicated_keyframes** (*boolean, (optional)*) – Duplicated Keyframes, Transform duplicated keyframes
- **view2d_edge_pan** (*boolean, (optional)*) – Edge Pan, Enable edge panning in 2D view
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation
- **use_automerge_and_split** (*boolean, (optional)*) – Auto Merge & Split, Forces the use of Auto Merge and Split

bpy.ops.transform.**vert_crease(*, value=0.0, snap=False, release_confirm=False, use_accurate=False)**

Change the crease of vertices

PARAMETERS:

- **value** (*float in [-1, 1], (optional)*) – Factor
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**vert_slide(*, value=0.0, use_even=False, flipped=False, use_clamp=True, mirror=False, snap=False, snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False, snap_point=(0.0, 0.0, 0.0), correct_uv=True, release_confirm=False, use_accurate=False)**

Slide a vertex along a mesh

PARAMETERS:

- **value** (*float in [-10, 10], (optional)*) – Factor
- **use_even** (*boolean, (optional)*) – Even, Make the edge loop match the shape of the adjacent edge loop
- **flipped** (*boolean, (optional)*) – Flipped, When Even mode is active, flips between the two adjacent edge loops
- **use_clamp** (*boolean, (optional)*) – Clamp, Clamp within the edge extents
- **mirror** (*boolean, (optional)*) – Mirror Editing
- **snap** (*boolean, (optional)*) – Use Snapping Options
- **snap_elements** (enum set in Snap Element Items, (optional)) – Snap to Elements
- **use_snap_project** (*boolean, (optional)*) – Project Individual Elements
- **snap_target** (enum in Snap Source Items, (optional)) – Snap Base, Point on source that will snap to target
- **use_snap_self** (*boolean, (optional)*) – Target: Include Active
- **use_snap_edit** (*boolean, (optional)*) – Target: Include Edit
- **use_snap_nonedit** (*boolean, (optional)*) – Target: Include Non-Edited
- **use_snap_selectable** (*boolean, (optional)*) – Target: Exclude Non-Selectable
- **snap_point** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Point
- **correct_uv** (*boolean, (optional)*) – Correct UVs, Correct UV coordinates when transforming
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation

bpy.ops.transform.**vertex_random(*, offset=0.0, uniform=0.0, normal=0.0, seed=0, wait_for_input=True)**

Randomize vertices

PARAMETERS:

- **offset** (*float in [-inf, inf], (optional)*) – Amount, Distance to offset
- **uniform** (*float in [0, 1], (optional)*) – Uniform, Increase for uniform offset distance
- **normal** (*float in [0, 1], (optional)*) – Normal, Align offset direction to normals
- **seed** (*int in [0, 10000], (optional)*) – Random Seed, Seed for the random number generator
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.transform.**vertex_warp(*, warp_angle=6.28319, offset_angle=0.0, min=-1.0, max=1.0, viewmat=((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)), center=(0.0, 0.0, 0.0))**

Warp vertices around the cursor

PARAMETERS:

- **warp_angle** (*float in [-inf, inf], (optional)*) – Warp Angle, Amount to warp about the cursor
- **offset_angle** (*float in [-inf, inf], (optional)*) – Offset Angle, Angle to use as the basis for warping
- **min** (*float in [-inf, inf], (optional)*) – Min
- **max** (*float in [-inf, inf], (optional)*) – Max
- **viewmat** (`mathutils.Matrix` of 4 * 4 items in [-inf, inf], (optional)) – Matrix
- **center** (`mathutils.Vector` of 3 items in [-inf, inf], (optional)) – Center

Skip to content

Ui Operators

bpy.ops.ui.**assign_default_button()**

Set this property's current value as the new default

bpy.ops.ui.**button_execute(*, skip_depressed=False)**

Presses active button

PARAMETERS:

skip_depressed (*boolean, (optional)*) – Skip Depressed

bpy.ops.ui.**button_string_clear()**

Unsets the text of the active button

bpy.ops.ui.**copy_as_driver_button()**

Create a new driver with this property as input, and copy it to the internal clipboard. Use Paste Driver to add it to the target property, or Paste Driver Variables to extend an existing driver

bpy.ops.ui.**copy_data_path_button(*, full_path=False)**

Copy the RNA data path for this property to the clipboard

PARAMETERS:

full_path (*boolean, (optional)*) – full_path, Copy full data path

bpy.ops.ui.**copy_driver_to_selected_button(*, all=False)**

Copy the property's driver from the active item to the same property of all selected items, if the same property exists

PARAMETERS:

all (*boolean, (optional)*) – All, Copy to selected the drivers of all elements of the array

bpy.ops.ui.**copy_python_command_button()**

Copy the Python command matching this button

bpy.ops.ui.**copy_to_selected_button(*, all=True)**

Copy the property's value from the active item to the same property of all selected items if the same property exists

PARAMETERS:

all (*boolean, (optional)*) – All, Copy to selected all elements of the array

bpy.ops.ui.**drop_color(*, color=(0.0, 0.0, 0.0, 0.0), gamma=False, has_alpha=False)**

Drop colors to buttons

PARAMETERS:

- **color** (*float array of 4 items in [0, inf], (optional)*) – Color, Source color
- **gamma** (*boolean, (optional)*) – Gamma Corrected, The source color is gamma corrected
- **has_alpha** (*boolean, (optional)*) – Has Alpha, The source color contains an Alpha component

bpy.ops.ui.**drop_material(*, session_uid=0)**

Drag material to Material slots in Properties

PARAMETERS:

session_uid (*int in [-inf, inf], (optional)*) – Session UID, Session UID of the data-block to use by the operator

bpy.ops.ui.**drop_name(*, string='')**

Drop name to button

PARAMETERS:

string (*string, (optional, never None)*) – String, The string value to drop into the button

bpy.ops.ui.**editsource()**

Edit UI source code of the active button

bpy.ops.ui.**eyedropper_bone()**

Sample a bone from the 3D View or the Outliner to store in a property

bpy.ops.ui.**eyedropper_color(*, prop_data_path='')**

Sample a color from the Blender window to store in a property

PARAMETERS:

prop_data_path (*string, (optional, never None)*) – Data Path, Path of property to be set with the depth

bpy.ops.ui.**eyedropper_colorramp()**

Sample a color band

bpy.ops.ui.**eyedropper_colorramp_point()**

Point-sample a color band

bpy.ops.ui.**eyedropper_depth(*, prop_data_path='')**

Sample depth from the 3D view

PARAMETERS:

prop_data_path (*string, (optional, never None)*) – Data Path, Path of property to be set with the depth

bpy.ops.ui.**eyedropper_driver(*, mapping_type='SINGLE_MANY')**

Pick a property to use as a driver target

PARAMETERS:

mapping_type (*enum in ['SINGLE_MANY', 'DIRECT', 'MATCH', 'NONE_ALL', 'NONE_SINGLE'], (optional)*) –

Mapping Type, Method used to match target and driven properties

- `SINGLE_MANY` All from Target – Drive all components of this property using the target picked.
- `DIRECT` Single from Target – Drive this component of this property using the target picked.
- `MATCH` Match Indices – Create drivers for each pair of corresponding elements.
- `NONE_ALL` Manually Create Later – Create drivers for all properties without assigning any targets yet.
- `NONE_SINGLE` Manually Create Later (Single) – Create driver for this property only and without assigning any targets yet.

bpy.ops.ui.**eyedropper_grease_pencil_color(*, mode='MATERIAL', material_mode='STROKE')**

Sample a color from the Blender Window and create Grease Pencil material

PARAMETERS:

- **mode** (*enum in ['MATERIAL', 'PALETTE', 'BRUSH'], (optional)*) – Mode
- **material_mode** (*enum in ['STROKE', 'FILL', 'BOTH'], (optional)*) – Material Mode

bpy.ops.ui.**eyedropper_id()**

Sample a data-block from the 3D View to store in a property

bpy.ops.ui.**jump_to_target_button()**

Switch to the target object or bone

bpy.ops.ui.**list_start_filter()**

Start entering filter text for the list in focus bpy.ops.ui.**override_idtemplate_clear()**

Delete the selected local override and relink its usages to the linked data-block if possible, else reset it and mark it as non editable

bpy.ops.ui.**override_idtemplate_make()**

Create a local override of the selected linked data-block, and its hierarchy of dependencies

bpy.ops.ui.**override_idtemplate_reset()**

Reset the selected local override to its linked reference values

bpy.ops.ui.**override_remove_button(*, all=True)**

Remove an override operation

PARAMETERS:

all (*boolean, (optional)*) – All, Reset to default values all elements of the array

bpy.ops.ui.**override_type_set_button(*, all=True, type='REPLACE')**

Create an override operation, or set the type of an existing one

PARAMETERS:

- **all** (*boolean, (optional)*) – All, Reset to default values all elements of the array
- **type** (*enum in ['NOOP', 'REPLACE', 'DIFFERENCE', 'FACTOR'], (optional)*) – Type, Type of override operation
  - `NOOP` NoOp – 'No-Operation', place holder preventing automatic override to ever affect the property.
  - `REPLACE` Replace – Completely replace value from linked data by local one.
  - `DIFFERENCE` Difference – Store difference to linked data value.
  - `FACTOR` Factor – Store factor to linked data value (useful e.g. for scale).

bpy.ops.ui.**reloadtranslation()**

Force a full reload of UI translation

bpy.ops.ui.**reset_default_button(*, all=True)**

Reset this property's value to its default value

PARAMETERS:

all (*boolean, (optional)*) – All, Reset to default values all elements of the array

bpy.ops.ui.**unset_property_button()**

Clear the property and use default or generated value in operators

bpy.ops.ui.**view_drop()**

Drag and drop onto a data-set or item within the data-set

bpy.ops.ui.**view_item_rename()**

Rename the active item in the data-set view

bpy.ops.ui.**view_scroll()**

Undocumented, consider contributing.

bpy.ops.ui.**view_start_filter()**

Start entering filter text for the data-set in focus

Uilist Operators

bpy.ops.uilist.**entry_add(*, list_path='', active_index_path='')**

Add an entry to the list after the current active item

PARAMETERS:

- **list_path** (*string, (optional, never None)*) – list_path
- **active_index_path** (*string, (optional, never None)*) – active_index_path

FILE:

startup/bl_ui/generic_ui_list.py:210

bpy.ops.uilist.**entry_move(*, list_path='', active_index_path='', direction='UP')**

Move an entry in the list up or down

PARAMETERS:

- **list_path** (*string, (optional, never None)*) – list_path
- **active_index_path** (*string, (optional, never None)*) – active_index_path
- **direction** (*enum in ['UP', 'DOWN'], (optional)*) –

  Direction

  - `UP` UP – UP.
  - `DOWN` DOWN – DOWN.

FILE:

startup/bl_ui/generic_ui_list.py:238

bpy.ops.uilist.**entry_remove(*, list_path='', active_index_path='')**

Remove the selected entry from the list

PARAMETERS:

- **list_path** (*string, (optional, never None)*) – list_path
- **active_index_path** (*string, (optional, never None)*) – active_index_path

FILE:

startup/bl_ui/generic_ui_list.py:193

Skip to content

Uv Operators

bpy.ops.uv.**align(*, axis='ALIGN_AUTO')**

Aligns selected UV vertices on a line

PARAMETERS:

axis (*enum in ['ALIGN_S', 'ALIGN_T', 'ALIGN_U', 'ALIGN_AUTO', 'ALIGN_X', 'ALIGN_Y'], (optional)*) –

Axis, Axis to align UV locations on

- `ALIGN_S` Straighten – Align UV vertices along the line defined by the endpoints.
- `ALIGN_T` Straighten X – Align UV vertices, moving them horizontally to the line defined by the endpoints.
- `ALIGN_U` Straighten Y – Align UV vertices, moving them vertically to the line defined by the endpoints.
- `ALIGN_AUTO` Align Auto – Automatically choose the direction on which there is most alignment already.
- `ALIGN_X` Align Vertically – Align UV vertices on a vertical line.
- `ALIGN_Y` Align Horizontally – Align UV vertices on a horizontal line.

bpy.ops.uv.**align_rotation(*, method='AUTO', axis='X', correct_aspect=False)**

Align the UV island's rotation

PARAMETERS:

- **method** (*enum in ['AUTO', 'EDGE', 'GEOMETRY'], (optional)*) –

  Method, Method to calculate rotation angle

  - `AUTO` Auto – Align from all edges.
  - `EDGE` Edge – Only selected edges.
  - `GEOMETRY` Geometry – Align to Geometry axis.

- **axis** (*enum in ['X', 'Y', 'Z'], (optional)*) –

  Axis, Axis to align to

  - `X` X – X axis.
  - `Y` Y – Y axis.
  - `Z` Z – Z axis.

- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Take image aspect ratio into account

FILE:

startup/bl_operators/uvcalc_transform.py:299

bpy.ops.uv.**average_islands_scale(*, scale_uv=False, shear=False)**

Average the size of separate UV islands, based on their area in 3D space

PARAMETERS:

- **scale_uv** (*boolean, (optional)*) – Non-Uniform, Scale U and V independently
- **shear** (*boolean, (optional)*) – Shear, Reduce shear within islands

bpy.ops.uv.**copy()**

Copy selected UV vertices

bpy.ops.uv.**cube_project(*, cube_size=1.0, correct_aspect=True, clip_to_bounds=False, scale_to_bounds=False)**

Project the UV vertices of the mesh over the six faces of a cube

PARAMETERS:

- **cube_size** (*float in [0, inf], (optional)*) – Cube Size, Size of the cube to project on
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into account

- **clip_to_bounds** (*boolean, (optional)*) – Clip to Bounds, Clip UV coordinates to bounds after unwrapping
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**cursor_set(*, location=(0.0, 0.0))**

Set 2D cursor location

PARAMETERS:

location (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Cursor location in normalized (0.0 to 1.0) coordinates

bpy.ops.uv.**cylinder_project(*, direction='VIEW_ON_EQUATOR', align='POLAR_ZX', pole='PINCH', seam=False, radius=1.0, correct_aspect=True, clip_to_bounds=False, scale_to_bounds=False)**

Project the UV vertices of the mesh over the curved wall of a cylinder

PARAMETERS:

- **direction** (*enum in ['VIEW_ON_EQUATOR', 'VIEW_ON_POLES', 'ALIGN_TO_OBJECT'], (optional)*) –
  Direction, Direction of the sphere or cylinder
  - `VIEW_ON_EQUATOR` View on Equator – 3D view is on the equator.
  - `VIEW_ON_POLES` View on Poles – 3D view is on the poles.
  - `ALIGN_TO_OBJECT` Align to Object – Align according to object transform.
- **align** (*enum in ['POLAR_ZX', 'POLAR_ZY'], (optional)*) –
  Align, How to determine rotation around the pole
  - `POLAR_ZX` Polar ZX – Polar 0 is X.
  - `POLAR_ZY` Polar ZY – Polar 0 is Y.
- **pole** (*enum in ['PINCH', 'FAN'], (optional)*) –
  Pole, How to handle faces at the poles
  - `PINCH` Pinch – UVs are pinched at the poles.
  - `FAN` Fan – UVs are fanned at the poles.
- **seam** (*boolean, (optional)*) – Preserve Seams, Separate projections by islands isolated by seams
- **radius** (*float in [0, inf], (optional)*) – Radius, Radius of the sphere or cylinder
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun
- **clip_to_bounds** (*boolean, (optional)*) – Clip to Bounds, Clip UV coordinates to bounds after unwrapping
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**export_layout(*, filepath='', export_all=False, export_tiles='NONE', modified=False, mode='PNG', size=(1024, 1024), opacity=0.25, check_existing=True)**

Export UV layout to file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – filepath
- **export_all** (*boolean, (optional)*) – All UVs, Export all UVs in this mesh (not just visible ones)
- **export_tiles** (*enum in ['NONE', 'UDIM', 'UV'], (optional)*) –
  Export Tiles, Choose whether to export only the [0, 1] range, or all UV tiles
  - `NONE` None – Export only UVs in the [0, 1] range.
  - `UDIM` UDIM – Export tiles in the UDIM numbering scheme: 1001 + u_tile + 10*v_tile.
  - `UV` UVTILE – Export tiles in the UVTILE numbering scheme: u(u_tile + 1)_v(v_tile + 1).
- **modified** (*boolean, (optional)*) – Modified, Exports UVs from the modified mesh
- **mode** (*enum in ['SVG', 'EPS', 'PNG'], (optional)*) –
  Format, File format to export the UV layout to
  - `SVG` Scalable Vector Graphic (.svg) – Export the UV layout to a vector SVG file.

    - `EPS` Encapsulated PostScript (.eps) – Export the UV layout to a vector EPS file.
    - `PNG` PNG Image (.png) – Export the UV layout to a bitmap image.

- **size** (*int array of 2 items in [8, 32768], (optional)*) – Size, Dimensions of the exported file
- **opacity** (*float in [0, 1], (optional)*) – Fill Opacity, Set amount of opacity for exported UV layout
- **check_existing** (*boolean, (optional)*) – check_existing

FILE:

addons_core/io_mesh_uv_layout/__init__.py:137

bpy.ops.uv.**follow_active_quads(*, mode='LENGTH_AVERAGE')**

Follow UVs from active quads along continuous face loops

PARAMETERS:

mode (*enum in ['EVEN', 'LENGTH', 'LENGTH_AVERAGE'], (optional)*) –

Edge Length Mode, Method to space UV edge loops

- `EVEN` Even – Space all UVs evenly.
- `LENGTH` Length – Average space UVs edge length of each loop.
- `LENGTH_AVERAGE` Length Average – Average space UVs edge length of each loop.

FILE:

startup/bl_operators/uvcalc_follow_active.py:297

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

Hide (un)selected UV vertices

PARAMETERS:

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

bpy.ops.uv.**lightmap_pack(*, PREF_CONTEXT='SEL_FACES', PREF_PACK_IN_ONE=True, PREF_NEW_UVLAYER=False, PREF_BOX_DIV=12, PREF_MARGIN_DIV=0.1)**

Pack each face’s UVs into the UV bounds

PARAMETERS:

- **PREF_CONTEXT** (*enum in ['SEL_FACES', 'ALL_FACES'], (optional)*) –

  Selection

    - `SEL_FACES` Selected Faces – Space all UVs evenly.
    - `ALL_FACES` All Faces – Average space UVs edge length of each loop.

- **PREF_PACK_IN_ONE** (*boolean, (optional)*) – Share Texture Space, Objects share texture space, map all objects into a single UV map
- **PREF_NEW_UVLAYER** (*boolean, (optional)*) – New UV Map, Create a new UV map for every mesh packed
- **PREF_BOX_DIV** (*int in [1, 48], (optional)*) – Pack Quality, Quality of the packing. Higher values will be slower but waste less space
- **PREF_MARGIN_DIV** (*float in [0.001, 1], (optional)*) – Margin, Size of the margin as a division of the UV

FILE:

startup/bl_operators/uvcalc_lightmap.py:662

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

Mark selected UV edges as seams

PARAMETERS:

clear (*boolean, (optional)*) – Clear Seams, Clear instead of marking seams

bpy.ops.uv.**minimize_stretch(*, fill_holes=True, blend=0.0, iterations=0)**

Reduce UV stretching by relaxing angles

PARAMETERS:

PARAMETERS:

- **fill_holes** (*boolean, (optional)*) – Fill Holes, Virtually fill holes in mesh before unwrapping, to better avoid overlaps and preserve symmetry
- **blend** (*float in [0, 1], (optional)*) – Blend, Blend factor between stretch minimized and original
- **iterations** (*int in [0, inf], (optional)*) – Iterations, Number of iterations to run, 0 is unlimited when run interactively

bpy.ops.uv.**pack_islands(*, udim_source='CLOSEST_UDIM', rotate=True, rotate_method='ANY', scale=True, merge_overlap=False, margin_method='SCALED', margin=0.001, pin=False, pin_method='LOCKED', shape_method='CONCAVE')**

Transform all islands so that they fill up the UV/UDIM space as much as possible

PARAMETERS:

- **udim_source** (*enum in ['CLOSEST_UDIM', 'ACTIVE_UDIM', 'ORIGINAL_AABB'], (optional)*) –
  Pack to
  - `CLOSEST_UDIM` Closest UDIM – Pack islands to closest UDIM.
  - `ACTIVE_UDIM` Active UDIM – Pack islands to active UDIM image tile or UDIM grid tile where 2D cursor is located.
  - `ORIGINAL_AABB` Original bounding box – Pack to starting bounding box of islands.
- **rotate** (*boolean, (optional)*) – Rotate, Rotate islands to improve layout
- **rotate_method** (*enum in ['ANY', 'CARDINAL', 'AXIS_ALIGNED', 'AXIS_ALIGNED_X', 'AXIS_ALIGNED_Y'], (optional)*) –
  Rotation Method
  - `ANY` Any – Any angle is allowed for rotation.
  - `CARDINAL` Cardinal – Only 90 degree rotations are allowed.
  - `AXIS_ALIGNED` Axis-aligned – Rotated to a minimal rectangle, either vertical or horizontal.
  - `AXIS_ALIGNED_X` Axis-aligned (Horizontal) – Rotate islands to be aligned horizontally.
  - `AXIS_ALIGNED_Y` Axis-aligned (Vertical) – Rotate islands to be aligned vertically.
- **scale** (*boolean, (optional)*) – Scale, Scale islands to fill unit square
- **merge_overlap** (*boolean, (optional)*) – Merge Overlapping, Overlapping islands stick together
- **margin_method** (*enum in ['SCALED', 'ADD', 'FRACTION'], (optional)*) –
  Margin Method
  - `SCALED` Scaled – Use scale of existing UVs to multiply margin.
  - `ADD` Add – Just add the margin, ignoring any UV scale.
  - `FRACTION` Fraction – Specify a precise fraction of final UV output.
- **margin** (*float in [0, 1], (optional)*) – Margin, Space between islands
- **pin** (*boolean, (optional)*) – Lock Pinned Islands, Constrain islands containing any pinned UV's
- **pin_method** (*enum in ['SCALE', 'ROTATION', 'ROTATION_SCALE', 'LOCKED'], (optional)*) –
  Pin Method
  - `SCALE` Scale – Pinned islands won't rescale.
  - `ROTATION` Rotation – Pinned islands won't rotate.
  - `ROTATION_SCALE` Rotation and Scale – Pinned islands will translate only.
  - `LOCKED` All – Pinned islands are locked in place.
- **shape_method** (*enum in ['CONCAVE', 'CONVEX', 'AABB'], (optional)*) –
  Shape Method
  - `CONCAVE` Exact Shape (Concave) – Uses exact geometry.
  - `CONVEX` Boundary Shape (Convex) – Uses convex hull.
  - `AABB` Bounding Box – Uses bounding boxes.

bpy.ops.uv.**paste()**

Paste selected UV vertices

bpy.ops.uv.**pin(*, clear=False, invert=False)**

Set/clear selected UV vertices as anchored between multiple unwrap operations

PARAMETERS:

- **clear** (*boolean, (optional)*) – Clear, Clear pinning for the selection instead of setting it
- **invert** (*boolean, (optional)*) – Invert, Invert pinning for the selection instead of setting it

bpy.ops.uv.**project_from_view(*, orthographic=False, camera_bounds=True, correct_aspect=True, clip_to_bounds=False, scale_to_bounds=False)**

Project the UV vertices of the mesh as seen in current 3D view

PARAMETERS:

- **orthographic** (*boolean, (optional)*) – Orthographic, Use orthographic projection
- **camera_bounds** (*boolean, (optional)*) – Camera Bounds, Map UVs to the camera region taking resolution and aspect into account
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into account
- **clip_to_bounds** (*boolean, (optional)*) – Clip to Bounds, Clip UV coordinates to bounds after unwrapping
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

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

Randomize the UV island’s location, rotation, and scale

PARAMETERS:

- **random_seed** (*int in [0, 10000], (optional)*) – Random Seed, Seed value for the random generator
- **use_loc** (*boolean, (optional)*) – Randomize Location, Randomize the location values
- **loc** (`mathutils.Vector` of 2 items in [-100, 100], (optional)) – Location, Maximum distance the objects can spread over each axis
- **use_rot** (*boolean, (optional)*) – Randomize Rotation, Randomize the rotation value
- **rot** (*float in [-6.28319, 6.28319], (optional)*) – Rotation, Maximum rotation
- **use_scale** (*boolean, (optional)*) – Randomize Scale, Randomize the scale values
- **scale_even** (*boolean, (optional)*) – Scale Even, Use the same scale value for both axes
- **scale** (*float array of 2 items in [-100, 100], (optional)*) – Scale, Maximum scale randomization over each axis

FILE:

startup/bl_operators/uvcalc_transform.py:473

bpy.ops.uv.**remove_doubles(*, threshold=0.02, use_unselected=False, use_shared_vertex=False)**

Selected UV vertices that are within a radius of each other are welded together

PARAMETERS:

- **threshold** (*float in [0, 10], (optional)*) – Merge Distance, Maximum distance between welded vertices
- **use_unselected** (*boolean, (optional)*) – Unselected, Merge selected to other unselected vertices
- **use_shared_vertex** (*boolean, (optional)*) – Shared Vertex, Weld UVs based on shared vertices

bpy.ops.uv.**reset()**

Reset UV projection

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

Reveal all hidden UV vertices

PARAMETERS:

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

bpy.ops.uv.**rip(*, mirror=False, release_confirm=False, use_accurate=False, location=(0.0, 0.0))**

Rip selected vertices or a selected region

PARAMETERS:

- **mirror** (*boolean, (optional)*) – Mirror Editing
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**rip_move(*, UV_OT_rip=None, TRANSFORM_OT_translate=None)**

Unstitch UVs and move the result

PARAMETERS:

- **UV_OT_rip** (`UV_OT_rip` , (optional)) – UV Rip, Rip selected vertices or a selected region
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate` , (optional)) – Move, Move selected items

bpy.ops.uv.**seams_from_islands(*, mark_seams=True, mark_sharp=False)**

Set mesh seams according to island setup in the UV editor

PARAMETERS:

- **mark_seams** (*boolean, (optional)*) – Mark Seams, Mark boundary edges as seams
- **mark_sharp** (*boolean, (optional)*) – Mark Sharp, Mark boundary edges as sharp

bpy.ops.uv.**select(*, extend=False, deselect=False, toggle=False, deselect_all=False, select_passthrough=False, location=(0.0, 0.0))**

Select UV vertices

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **deselect** (*boolean, (optional)*) – Deselect, Remove from selection
- **toggle** (*boolean, (optional)*) – Toggle Selection, Toggle the selection
- **deselect_all** (*boolean, (optional)*) – Deselect On Nothing, Deselect all when nothing under the cursor
- **select_passthrough** (*boolean, (optional)*) – Only Select Unselected, Ignore the select action when the element is already selected
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

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

Change selection of all UV vertices

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.uv.**select_box(*, pinned=False, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, mode='SET')**

Select UV vertices using box selection

PARAMETERS:

- **pinned** (*boolean, (optional)*) – Pinned, Border select pinned UVs only
- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

- **mode** (*enum in ['SET', 'ADD', 'SUB'], (optional)*) –

  Mode

  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.

bpy.ops.uv.**select_circle(*, x=0, y=0, radius=25, wait_for_input=True, mode='SET')**

Select UV vertices using circle selection

PARAMETERS:

- **x** (*int in [-inf, inf], (optional)*) – X
- **y** (*int in [-inf, inf], (optional)*) – Y
- **radius** (*int in [1, inf], (optional)*) – Radius
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **mode** (*enum in ['SET', 'ADD', 'SUB'], (optional)*) –

  Mode

  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.

bpy.ops.uv.**select_edge_ring(*, extend=False, location=(0.0, 0.0))**

Select an edge ring of connected UV vertices

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection rather than clearing the existing selection
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**select_lasso(*, path=None, use_smooth_stroke=False, smooth_stroke_factor=0.75, smooth_stroke_radius=35, mode='SET')**

Select UVs using lasso selection

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **use_smooth_stroke** (*boolean, (optional)*) – Stabilize Stroke, Selection lags behind mouse and follows a smoother path
- **smooth_stroke_factor** (*float in [0.5, 0.99], (optional)*) – Smooth Stroke Factor, Higher values gives a smoother stroke
- **smooth_stroke_radius** (*int in [10, 200], (optional)*) – Smooth Stroke Radius, Minimum distance from last point before selection continue
- **mode** (*enum in ['SET', 'ADD', 'SUB'], (optional)*) –

  Mode

  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.

bpy.ops.uv.**select_less()**

Deselect UV vertices at the boundary of each selection region

bpy.ops.uv.**select_linked()**

Select all UV vertices linked to the active UV map

bpy.ops.uv.**select_linked_pick(*, extend=False, deselect=False, location=(0.0, 0.0))**

Select all UV vertices linked under the mouse

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection rather than clearing the existing selection
- **deselect** (*boolean, (optional)*) – Deselect, Deselect linked UV vertices rather than selecting them
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**select_loop(*, extend=False, location=(0.0, 0.0))**

Select a loop of connected UV vertices

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection rather than clearing the existing selection
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**select_mode(*, type='VERTEX')**

Change UV selection mode

PARAMETERS:

type (enum in Mesh Select Mode Uv Items, (optional)) – Type

bpy.ops.uv.**select_more()**

Select more UV vertices connected to initial selection

bpy.ops.uv.**select_overlap(*, extend=False)**

Select all UV faces which overlap each other

PARAMETERS:

extend (*boolean, (optional)*) – Extend, Extend selection rather than clearing the existing selection

bpy.ops.uv.**select_pinned()**

Select all pinned UV vertices

bpy.ops.uv.**select_similar(*, type='PIN', compare='EQUAL', threshold=0.0)**

Select similar UVs by property types

PARAMETERS:

- **type** (*enum in ['PIN', 'LENGTH', 'LENGTH_3D', 'AREA', 'AREA_3D', 'MATERIAL', 'OBJECT', 'SIDES', 'WINDING', 'FACE'], (optional)*) – Type
- **compare** (*enum in ['EQUAL', 'GREATER', 'LESS'], (optional)*) – Compare
- **threshold** (*float in [0, 1], (optional)*) – Threshold

bpy.ops.uv.**select_split()**

Select only entirely selected faces

bpy.ops.uv.**shortest_path_pick(*, use_face_step=False, use_topology_distance=False, use_fill=False, skip=0, nth=1, offset=0, object_index=-1, index=-1)**

Select shortest path between two selections

PARAMETERS:

- **use_face_step** (*boolean, (optional)*) – Face Stepping, Traverse connected faces (includes diagonals and edge-rings)
- **use_topology_distance** (*boolean, (optional)*) – Topology Distance, Find the minimum number of steps, ignoring spatial distance
- **use_fill** (*boolean, (optional)*) – Fill Region, Select all paths between the source/destination elements
- **skip** (*int in [0, inf], (optional)*) – Deselected, Number of deselected elements in the repetitive sequence
- **nth** (*int in [1, inf], (optional)*) – Selected, Number of selected elements in the repetitive sequence
- **offset** (*int in [-inf, inf], (optional)*) – Offset, Offset from the starting point

bpy.ops.uv.**shortest_path_select(*, use_face_step=False, use_topology_distance=False, use_fill=False, skip=0, nth=1, offset=0)**

Selected shortest path between two vertices/edges/faces

PARAMETERS:

- **use_face_step** (*boolean, (optional)*) – Face Stepping, Traverse connected faces (includes diagonals and edge-rings)
- **use_topology_distance** (*boolean, (optional)*) – Topology Distance, Find the minimum number of steps, ignoring spatial distance
- **use_fill** (*boolean, (optional)*) – Fill Region, Select all paths between the source/destination elements
- **skip** (*int in [0, inf], (optional)*) – Deselected, Number of deselected elements in the repetitive sequence
- **nth** (*int in [1, inf], (optional)*) – Selected, Number of selected elements in the repetitive sequence
- **offset** (*int in [-inf, inf], (optional)*) – Offset, Offset from the starting point

bpy.ops.uv.**smart_project(*, angle_limit=1.15192, margin_method='SCALED', rotate_method='AXIS_ALIGNED_Y', island_margin=0.0, area_weight=0.0, correct_aspect=True, scale_to_bounds=False)**

Projection unwraps the selected faces of mesh objects

PARAMETERS:

- **angle_limit** (*float in [0, 1.5708], (optional)*) – Angle Limit, Lower for more projection groups, higher for less distortion
- **margin_method** (*enum in ['SCALED', 'ADD', 'FRACTION'], (optional)*) –

  Margin Method

  - `SCALED` Scaled – Use scale of existing UVs to multiply margin.
  - `ADD` Add – Just add the margin, ignoring any UV scale.
  - `FRACTION` Fraction – Specify a precise fraction of final UV output.
- **rotate_method** (*enum in ['AXIS_ALIGNED', 'AXIS_ALIGNED_X', 'AXIS_ALIGNED_Y'], (optional)*) –

  Rotation Method

  - `AXIS_ALIGNED` Axis-aligned – Rotated to a minimal rectangle, either vertical or horizontal.
  - `AXIS_ALIGNED_X` Axis-aligned (Horizontal) – Rotate islands to be aligned horizontally.
  - `AXIS_ALIGNED_Y` Axis-aligned (Vertical) – Rotate islands to be aligned vertically.
- **island_margin** (*float in [0, 1], (optional)*) – Island Margin, Margin to reduce bleed from adjacent islands
- **area_weight** (*float in [0, 1], (optional)*) – Area Weight, Weight projection's vector by faces with larger areas
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**snap_cursor(*, target='PIXELS')**

Snap cursor to target type

PARAMETERS:

target (*enum in ['PIXELS', 'SELECTED', 'ORIGIN'], (optional)*) – Target, Target to snap the selected UVs to

bpy.ops.uv.**snap_selected(*, target='PIXELS')**

Snap selected UV vertices to target type

PARAMETERS:

target (*enum in ['PIXELS', 'CURSOR', 'CURSOR_OFFSET', 'ADJACENT_UNSELECTED'], (optional)*) – Target, Target to snap the selected UVs to

bpy.ops.uv.**sphere_project(*, direction='VIEW_ON_EQUATOR', align='POLAR_ZX', pole='PINCH', seam=False, correct_aspect=True, clip_to_bounds=False, scale_to_bounds=False)**

Project the UV vertices of the mesh over the curved surface of a sphere

PARAMETERS:

- **direction** (*enum in ['VIEW_ON_EQUATOR', 'VIEW_ON_POLES', 'ALIGN_TO_OBJECT'], (optional)*) –

  Direction, Direction of the sphere or cylinder

- `VIEW_ON_EQUATOR` View on Equator – 3D view is on the equator.
- `VIEW_ON_POLES` View on Poles – 3D view is on the poles.
- `ALIGN_TO_OBJECT` Align to Object – Align according to object transform.

- **align** (*enum in ['POLAR_ZX', 'POLAR_ZY'], (optional)*) –
  Align, How to determine rotation around the pole
  - `POLAR_ZX` Polar ZX – Polar 0 is X.
  - `POLAR_ZY` Polar ZY – Polar 0 is Y.
- **pole** (*enum in ['PINCH', 'FAN'], (optional)*) –
  Pole, How to handle faces at the poles
  - `PINCH` Pinch – UVs are pinched at the poles.
  - `FAN` Fan – UVs are fanned at the poles.
- **seam** (*boolean, (optional)*) – Preserve Seams, Separate projections by islands isolated by seams
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun
- **clip_to_bounds** (*boolean, (optional)*) – Clip to Bounds, Clip UV coordinates to bounds after unwrapping
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**stitch(*, use_limit=False, snap_islands=True, limit=0.01, static_island=0, active_object_index=0, midpoint_snap=False, clear_seams=True, mode='VERTEX', stored_mode='VERTEX', selection=None, objects_selection_count=(0, 0, 0, 0, 0, 0))**

Stitch selected UV vertices by proximity

PARAMETERS:

- **use_limit** (*boolean, (optional)*) – Use Limit, Stitch UVs within a specified limit distance
- **snap_islands** (*boolean, (optional)*) – Snap Islands, Snap islands together (on edge stitch mode, rotates the islands too)
- **limit** (*float in [0, inf], (optional)*) – Limit, Limit distance in normalized coordinates
- **static_island** (*int in [0, inf], (optional)*) – Static Island, Island that stays in place when stitching islands
- **active_object_index** (*int in [0, inf], (optional)*) – Active Object, Index of the active object
- **midpoint_snap** (*boolean, (optional)*) – Snap at Midpoint, UVs are stitched at midpoint instead of at static island
- **clear_seams** (*boolean, (optional)*) – Clear Seams, Clear seams of stitched edges
- **mode** (*enum in ['VERTEX', 'EDGE'], (optional)*) – Operation Mode, Use vertex or edge stitching
- **stored_mode** (*enum in ['VERTEX', 'EDGE'], (optional)*) – Stored Operation Mode, Use vertex or edge stitching
- **selection** (`bpy_prop_collection` of `SelectedUvElement`, (optional)) – Selection
- **objects_selection_count** (*int array of 6 items in [0, inf], (optional)*) – Objects Selection Count

bpy.ops.uv.**unwrap(*, method='CONFORMAL', fill_holes=False, correct_aspect=True, use_subsurf_data=False, margin_method='SCALED', margin=0.001, no_flip=False, iterations=10, use_weights=False, weight_group='uv_importance', weight_factor=1.0)**

Unwrap the mesh of the object being edited

PARAMETERS:

- **method** (*enum in ['ANGLE_BASED', 'CONFORMAL', 'MINIMUM_STRETCH'], (optional)*) – Method, Unwrapping method (Angle Based usually gives better results than Conformal, while being somewhat slower)
- **fill_holes** (*boolean, (optional)*) – Fill Holes, Virtually fill holes in mesh before unwrapping, to better avoid overlaps and preserve symmetry
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun
- **use_subsurf_data** (*boolean, (optional)*) – Use Subdivision Surface, Map UVs taking vertex position after Subdivision Surface modifier has been applied
- **margin_method** (*enum in ['SCALED', 'ADD', 'FRACTION'], (optional)*) –
  Margin Method
  - `SCALED` Scaled – Use scale of existing UVs to multiply margin.
  - `ADD` Add – Just add the margin, ignoring any UV scale.

    - `FRACTION` Fraction – Specify a precise fraction of final UV output.

- **margin** (*float in [0, 1], (optional)*) – Margin, Space between islands
- **no_flip** (*boolean, (optional)*) – No Flip, Prevent flipping UV’s, flipping may lower distortion depending on the position of pins
- **iterations** (*int in [0, 10000], (optional)*) – Iterations, Number of iterations when “Minimum Stretch” method is used
- **use_weights** (*boolean, (optional)*) – Importance Weights, Whether to take into account per-vertex importance weights
- **weight_group** (*string, (optional, never None)*) – Weight Group, Vertex group name for importance weights (modulating the deform)
- **weight_factor** (*float in [-10000, 10000], (optional)*) – Weight Factor, How much influence the weightmap has for weighted parameterization, 0 being no influence

bpy.ops.uv.**weld()**

Weld selected UV vertices together

Skip to content

View2D Operators

bpy.ops.view2d.**edge_pan(*, inside_padding=1.0, outside_padding=0.0, speed_ramp=1.0, max_speed=500.0, delay=1.0, zoom_influence=0.0)**

Pan the view when the mouse is held at an edge

PARAMETERS:

- **inside_padding** (*float in [0, 100], (optional)*) – Inside Padding, Inside distance in UI units from the edge of the region within which to start panning
- **outside_padding** (*float in [0, 100], (optional)*) – Outside Padding, Outside distance in UI units from the edge of the region at which to sto panning
- **speed_ramp** (*float in [0, 100], (optional)*) – Speed Ramp, Width of the zone in UI units where speed increases with distance from the edg
- **max_speed** (*float in [0, 10000], (optional)*) – Max Speed, Maximum speed in UI units per second
- **delay** (*float in [0, 10], (optional)*) – Delay, Delay in seconds before maximum speed is reached
- **zoom_influence** (*float in [0, 1], (optional)*) – Zoom Influence, Influence of the zoom factor on scroll speed

bpy.ops.view2d.**ndof()**

Use a 3D mouse device to pan/zoom the view

bpy.ops.view2d.**pan(*, deltax=0, deltay=0)**

Pan the view

PARAMETERS:

- **deltax** (*int in [-inf, inf], (optional)*) – Delta X
- **deltay** (*int in [-inf, inf], (optional)*) – Delta Y

bpy.ops.view2d.**reset()**

Reset the view

bpy.ops.view2d.**scroll_down(*, deltax=0, deltay=0, page=False)**

Scroll the view down

PARAMETERS:

- **deltax** (*int in [-inf, inf], (optional)*) – Delta X
- **deltay** (*int in [-inf, inf], (optional)*) – Delta Y
- **page** (*boolean, (optional)*) – Page, Scroll down one page

bpy.ops.view2d.**scroll_left(*, deltax=0, deltay=0)**

Scroll the view left

PARAMETERS:

- **deltax** (*int in [-inf, inf], (optional)*) – Delta X
- **deltay** (*int in [-inf, inf], (optional)*) – Delta Y

bpy.ops.view2d.**scroll_right(*, deltax=0, deltay=0)**

Scroll the view right

PARAMETERS:

- **deltax** (*int in [-inf, inf], (optional)*) – Delta X
- **deltay** (*int in [-inf, inf], (optional)*) – Delta Y

bpy.ops.view2d.**scroll_up(*, deltax=0, deltay=0, page=False)**

Scroll the view up

PARAMETERS:

- **deltax** (*int in [-inf, inf], (optional)*) – Delta X
- **deltay** (*int in [-inf, inf], (optional)*) – Delta Y
- **page** (*boolean, (optional)*) – Page, Scroll up one page

bpy.ops.view2d.**scroller_activate()**

Scroll view by mouse click and drag

bpy.ops.view2d.**smoothview(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True)**

Undocumented, consider contributing.

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.view2d.**zoom(*, deltax=0.0, deltay=0.0, use_cursor_init=True)**

Zoom in/out the view

PARAMETERS:

- **deltax** (*float in [-inf, inf], (optional)*) – Delta X
- **deltay** (*float in [-inf, inf], (optional)*) – Delta Y
- **use_cursor_init** (*boolean, (optional)*) – Use Mouse Position, Allow the initial mouse position to be used

bpy.ops.view2d.**zoom_border(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, zoom_out=False)**

Zoom in the view to the nearest item contained in the border

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **zoom_out** (*boolean, (optional)*) – Zoom Out

bpy.ops.view2d.**zoom_in(*, zoomfacx=0.0, zoomfacy=0.0)**

Zoom in the view

PARAMETERS:

- **zoomfacx** (*float in [-inf, inf], (optional)*) – Zoom Factor X
- **zoomfacy** (*float in [-inf, inf], (optional)*) – Zoom Factor Y

bpy.ops.view2d.**zoom_out(*, zoomfacx=0.0, zoomfacy=0.0)**

Zoom out the view

PARAMETERS:

- **zoomfacx** (*float in [-inf, inf], (optional)*) – Zoom Factor X
- **zoomfacy** (*float in [-inf, inf], (optional)*) – Zoom Factor Y

View3D Operators

bpy.ops.view3d.**bone_select_menu(*, name='', extend=False, deselect=False, toggle=False)**

Menu bone selection

PARAMETERS:

- **name** (*enum in [], (optional)*) – Bone Name
- **extend** (*boolean, (optional)*) – Extend
- **deselect** (*boolean, (optional)*) – Deselect
- **toggle** (*boolean, (optional)*) – Toggle

bpy.ops.view3d.**camera_background_image_add(*, filepath='', relative_path=True, name='', session_uid=0)**

Add a new background image to the active camera

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – Filepath, Path to image file
- **relative_path** (*boolean, (optional)*) – Relative Path, Select the file relative to the blend file
- **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.view3d.**camera_background_image_remove(*, index=0)**

Remove a background image from the camera

PARAMETERS:

index (*int in [0, inf], (optional)*) – Index, Background image index to remove

bpy.ops.view3d.**camera_to_view()**

Set camera view to active view

bpy.ops.view3d.**camera_to_view_selected()**

Move the camera so selected objects are framed

bpy.ops.view3d.**clear_render_border()**

Clear the boundaries of the border render and disable border render

bpy.ops.view3d.**clip_border(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True)**

Set the view clipping region

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.view3d.**copybuffer()**

Copy the selected objects to the internal clipboard

bpy.ops.view3d.**cursor3d(*, use_depth=True, orientation='VIEW')**

Set the location of the 3D cursor

PARAMETERS:

- **use_depth** (*boolean, (optional)*) – Surface Project, Project onto the surface

- **orientation** (*enum in ['NONE', 'VIEW', 'XFORM', 'GEOM'], (optional)*) – Orientation, Preset viewpoint to use
    - `NONE` None – Leave orientation unchanged.
    - `VIEW` View – Orient to the viewport.
    - `XFORM` Transform – Orient to the current transform setting.
    - `GEOM` Geometry – Match the surface normal.

bpy.ops.view3d.**dolly(*, mx=0, my=0, delta=0, use_cursor_init=True)**

Dolly in/out in the view

PARAMETERS:

- **mx** (*int in [0, inf], (optional)*) – Region Position X
- **my** (*int in [0, inf], (optional)*) – Region Position Y
- **delta** (*int in [-inf, inf], (optional)*) – Delta
- **use_cursor_init** (*boolean, (optional)*) – Use Mouse Position, Allow the initial mouse position to be used

bpy.ops.view3d.**drop_world(*, name='', session_uid=0)**

Drop a world into the scene

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.view3d.**edit_mesh_extrude_individual_move()**

Extrude each individual face separately along local normals

FILE:

startup/bl_operators/view3d.py:31

bpy.ops.view3d.**edit_mesh_extrude_manifold_normal()**

Extrude manifold region along normals

FILE:

startup/bl_operators/view3d.py:202

bpy.ops.view3d.**edit_mesh_extrude_move_normal(*, dissolve_and_intersect=False)**

Extrude region together along the average normal

PARAMETERS:

dissolve_and_intersect (*boolean, (optional)*) – dissolve_and_intersect, Dissolves adjacent faces and intersects new geometry

FILE:

startup/bl_operators/view3d.py:168

bpy.ops.view3d.**edit_mesh_extrude_move_shrink_fatten()**

Extrude region together along local normals

FILE:

startup/bl_operators/view3d.py:185

bpy.ops.view3d.**fly()**

Interactively fly around the scene

bpy.ops.view3d.**interactive_add(*, primitive_type='CUBE', plane_origin_base='EDGE', plane_origin_depth='EDGE', plane_aspect_base='FREE', plane_aspect_depth='FREE', wait_for_input=True)**

Interactively add an object

Interactively add an object

PARAMETERS:

- **primitive_type** (*enum in ['CUBE', 'CYLINDER', 'CONE', 'SPHERE_UV', 'SPHERE_ICO'], (optional)*) – Primitive
- **plane_origin_base** (*enum in ['EDGE', 'CENTER'], (optional)*) –
  Origin, The initial position for placement
  - `EDGE` Edge – Start placing the edge position.
  - `CENTER` Center – Start placing the center position.
- **plane_origin_depth** (*enum in ['EDGE', 'CENTER'], (optional)*) –
  Origin, The initial position for placement
  - `EDGE` Edge – Start placing the edge position.
  - `CENTER` Center – Start placing the center position.
- **plane_aspect_base** (*enum in ['FREE', 'FIXED'], (optional)*) –
  Aspect, The initial aspect setting
  - `FREE` Free – Use an unconstrained aspect.
  - `FIXED` Fixed – Use a fixed 1:1 aspect.
- **plane_aspect_depth** (*enum in ['FREE', 'FIXED'], (optional)*) –
  Aspect, The initial aspect setting
  - `FREE` Free – Use an unconstrained aspect.
  - `FIXED` Fixed – Use a fixed 1:1 aspect.
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.view3d.**localview(*, frame_selected=True)**

Toggle display of selected object(s) separately and centered in view

PARAMETERS:

frame_selected (*boolean, (optional)*) – Frame Selected, Move the view to frame the selected objects

bpy.ops.view3d.**localview_remove_from()**

Move selected objects out of local view

bpy.ops.view3d.**move(*, use_cursor_init=True)**

Move the view

PARAMETERS:

use_cursor_init (*boolean, (optional)*) – Use Mouse Position, Allow the initial mouse position to be used

bpy.ops.view3d.**navigate()**

Interactively navigate around the scene (uses the mode (walk/fly) preference)

bpy.ops.view3d.**ndof_all()**

Pan and rotate the view with the 3D mouse

bpy.ops.view3d.**ndof_orbit()**

Orbit the view using the 3D mouse

bpy.ops.view3d.**ndof_orbit_zoom()**

Orbit and zoom the view using the 3D mouse

bpy.ops.view3d.**ndof_pan()**

Pan the view with the 3D mouse

bpy.ops.view3d.**object_as_camera()**

Set the active object as the active camera for this view or scene

bpy.ops.view3d.**object_mode_pie_or_toggle()**

Undocumented, consider contributing.

bpy.ops.view3d.**pastebuffer(*, autoselect=True, active_collection=True)**

Paste objects from the internal clipboard

PARAMETERS:

- **autoselect** (*boolean, (optional)*) – Select, Select pasted objects
- **active_collection** (*boolean, (optional)*) – Active Collection, Put pasted objects in the active collection

bpy.ops.view3d.**render_border(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True)**

Set the boundaries of the border render and enable border render

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

bpy.ops.view3d.**rotate(*, use_cursor_init=True)**

Rotate the view

PARAMETERS:

use_cursor_init (*boolean, (optional)*) – Use Mouse Position, Allow the initial mouse position to be used

bpy.ops.view3d.**ruler_add()**

Add ruler

bpy.ops.view3d.**ruler_remove()**

Undocumented, consider contributing.

bpy.ops.view3d.**select(*, extend=False, deselect=False, toggle=False, deselect_all=False, select_passthrough=False, center=False, enumerate=False, object=False, location=(0, 0))**

Select and activate item(s)

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **deselect** (*boolean, (optional)*) – Deselect, Remove from selection
- **toggle** (*boolean, (optional)*) – Toggle Selection, Toggle the selection
- **deselect_all** (*boolean, (optional)*) – Deselect On Nothing, Deselect all when nothing under the cursor
- **select_passthrough** (*boolean, (optional)*) – Only Select Unselected, Ignore the select action when the element is already selected
- **center** (*boolean, (optional)*) – Center, Use the object center when selecting, in edit mode used to extend object selection
- **enumerate** (*boolean, (optional)*) – Enumerate, List objects under the mouse (object mode only)
- **object** (*boolean, (optional)*) – Object, Use object selection (edit mode only)
- **location** (*int array of 2 items in [-inf, inf], (optional)*) – Location, Mouse location

bpy.ops.view3d.**select_box(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, mode='SET')**

Select items using box selection

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **mode** (*enum in ['SET', 'ADD', 'SUB', 'XOR', 'AND'], (optional)*) –
  Mode
  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.
  - `XOR` Difference – Invert existing selection.
  - `AND` Intersect – Intersect existing selection.

bpy.ops.view3d.**select_circle(*, x=0, y=0, radius=25, wait_for_input=True, mode='SET')**

Select items using circle selection

PARAMETERS:

- **x** (*int in [-inf, inf], (optional)*) – X
- **y** (*int in [-inf, inf], (optional)*) – Y
- **radius** (*int in [1, inf], (optional)*) – Radius
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **mode** (*enum in ['SET', 'ADD', 'SUB'], (optional)*) –
  Mode
  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.

bpy.ops.view3d.**select_lasso(*, path=None, use_smooth_stroke=False, smooth_stroke_factor=0.75, smooth_stroke_radius=35, mode='SET')**

Select items using lasso selection

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **use_smooth_stroke** (*boolean, (optional)*) – Stabilize Stroke, Selection lags behind mouse and follows a smoother path
- **smooth_stroke_factor** (*float in [0.5, 0.99], (optional)*) – Smooth Stroke Factor, Higher values gives a smoother stroke
- **smooth_stroke_radius** (*int in [10, 200], (optional)*) – Smooth Stroke Radius, Minimum distance from last point before selection continues
- **mode** (*enum in ['SET', 'ADD', 'SUB', 'XOR', 'AND'], (optional)*) –
  Mode
  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.
  - `XOR` Difference – Invert existing selection.
  - `AND` Intersect – Intersect existing selection.

bpy.ops.view3d.**select_menu(*, name='', extend=False, deselect=False, toggle=False)**

Menu object selection

PARAMETERS:

- **name** (*enum in [], (optional)*) – Object Name
- **extend** (*boolean, (optional)*) – Extend
- **deselect** (*boolean, (optional)*) – Deselect

- **deselect** (*boolean, (optional)*) – Deselect
- **toggle** (*boolean, (optional)*) – Toggle

bpy.ops.view3d.**smoothview()**

Undocumented, consider contributing.

bpy.ops.view3d.**snap_cursor_to_active()**

Snap 3D cursor to the active item

bpy.ops.view3d.**snap_cursor_to_center()**

Snap 3D cursor to the world origin

bpy.ops.view3d.**snap_cursor_to_grid()**

Snap 3D cursor to the nearest grid division

bpy.ops.view3d.**snap_cursor_to_selected()**

Snap 3D cursor to the middle of the selected item(s)

bpy.ops.view3d.**snap_selected_to_active()**

Snap selected item(s) to the active item

bpy.ops.view3d.**snap_selected_to_cursor(*, use_offset=True)**

Snap selected item(s) to the 3D cursor

PARAMETERS:

use_offset (*boolean, (optional)*) – Offset, If the selection should be snapped as a whole or by each object center

bpy.ops.view3d.**snap_selected_to_grid()**

Snap selected item(s) to their nearest grid division

bpy.ops.view3d.**toggle_matcap_flip()**

Flip MatCap

bpy.ops.view3d.**toggle_shading(*, type='WIREFRAME')**

Toggle shading type in 3D viewport

PARAMETERS:

type (*enum in ['WIREFRAME', 'SOLID', 'MATERIAL', 'RENDERED'], (optional)*) –

Type, Shading type to toggle

- `WIREFRAME` Wireframe – Toggle wireframe shading.
- `SOLID` Solid – Toggle solid shading.
- `MATERIAL` Material Preview – Toggle material preview shading.
- `RENDERED` Rendered – Toggle rendered shading.

bpy.ops.view3d.**toggle_xray()**

Transparent scene display. Allow selecting through items

bpy.ops.view3d.**transform_gizmo_set(*, extend=False, type={})**

Set the current transform gizmo

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend
- **type** (*enum set in {'TRANSLATE', 'ROTATE', 'SCALE'}, (optional)*) – Type

FILE:

[illegible]

startup/bl_operators/view3d.py:249

bpy.ops.view3d.**view_all(*, use_all_regions=False, center=False)**

View all objects in scene

PARAMETERS:

- **use_all_regions** (*boolean, (optional)*) – All Regions, View selected for all regions
- **center** (*boolean, (optional)*) – Center

bpy.ops.view3d.**view_axis(*, type='LEFT', align_active=False, relative=False)**

Use a preset viewpoint

PARAMETERS:

- **type** (*enum in ['LEFT', 'RIGHT', 'BOTTOM', 'TOP', 'FRONT', 'BACK'], (optional)*) – View, Preset viewpoint to use
  - `LEFT` Left – View from the left.
  - `RIGHT` Right – View from the right.
  - `BOTTOM` Bottom – View from the bottom.
  - `TOP` Top – View from the top.
  - `FRONT` Front – View from the front.
  - `BACK` Back – View from the back.
- **align_active** (*boolean, (optional)*) – Align Active, Align to the active object's axis
- **relative** (*boolean, (optional)*) – Relative, Rotate relative to the current orientation

bpy.ops.view3d.**view_camera()**

Toggle the camera view

bpy.ops.view3d.**view_center_camera()**

Center the camera view, resizing the view to fit its bounds

bpy.ops.view3d.**view_center_cursor()**

Center the view so that the cursor is in the middle of the view

bpy.ops.view3d.**view_center_lock()**

Center the view lock offset

bpy.ops.view3d.**view_center_pick()**

Center the view to the Z-depth position under the mouse cursor

bpy.ops.view3d.**view_lock_clear()**

Clear all view locking

bpy.ops.view3d.**view_lock_to_active()**

Lock the view to the active object/bone

bpy.ops.view3d.**view_orbit(*, angle=0.0, type='ORBITLEFT')**

Orbit the view

PARAMETERS:

- **angle** (*float in [-inf, inf], (optional)*) – Roll
- **type** (*enum in ['ORBITLEFT', 'ORBITRIGHT', 'ORBITUP', 'ORBITDOWN'], (optional)*) – Orbit, Direction of View Orbit
  - `ORBITLEFT` Orbit Left – Orbit the view around to the left.

    - `ORBITRIGHT` Orbit Right – Orbit the view around to the right.
    - `ORBITUP` Orbit Up – Orbit the view up.
    - `ORBITDOWN` Orbit Down – Orbit the view down.

bpy.ops.view3d.**view_pan(*, type='PANLEFT')**

Pan the view in a given direction

PARAMETERS:

type (*enum in ['PANLEFT', 'PANRIGHT', 'PANUP', 'PANDOWN'], (optional)*) –

Pan, Direction of View Pan

- `PANLEFT` Pan Left – Pan the view to the left.
- `PANRIGHT` Pan Right – Pan the view to the right.
- `PANUP` Pan Up – Pan the view up.
- `PANDOWN` Pan Down – Pan the view down.

bpy.ops.view3d.**view_persportho()**

Switch the current view from perspective/orthographic projection

bpy.ops.view3d.**view_roll(*, angle=0.0, type='ANGLE')**

Roll the view

PARAMETERS:

- **angle** (*float in [-inf, inf], (optional)*) – Roll
- **type** (*enum in ['ANGLE', 'LEFT', 'RIGHT'], (optional)*) –
  Roll Angle Source, How roll angle is calculated
    - `ANGLE` Roll Angle – Roll the view using an angle value.
    - `LEFT` Roll Left – Roll the view around to the left.
    - `RIGHT` Roll Right – Roll the view around to the right.

bpy.ops.view3d.**view_selected(*, use_all_regions=False)**

Move the view to the selection center

PARAMETERS:

use_all_regions (*boolean, (optional)*) – All Regions, View selected for all regions

bpy.ops.view3d.**walk()**

Interactively walk around the scene

bpy.ops.view3d.**zoom(*, mx=0, my=0, delta=0, use_cursor_init=True)**

Zoom in/out in the view

PARAMETERS:

- **mx** (*int in [0, inf], (optional)*) – Region Position X
- **my** (*int in [0, inf], (optional)*) – Region Position Y
- **delta** (*int in [-inf, inf], (optional)*) – Delta
- **use_cursor_init** (*boolean, (optional)*) – Use Mouse Position, Allow the initial mouse position to be used

bpy.ops.view3d.**zoom_border(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, zoom_out=False)**

Zoom in the view to the nearest object contained in the border

PARAMETERS:

- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max

- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **zoom_out** (*boolean, (optional)*) – Zoom Out

bpy.ops.view3d.**zoom_camera_1_to_1()**

Match the camera to 1:1 to the render output

Skip to content

Wm Operators

bpy.ops.wm.**alembic_export(*, filepath='', 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=False, filter_collada=False, filter_alembic=True, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', filter_glob='*.abc', start=-2147483648, end=-2147483648, xsamples=1, gsamples=1, sh_open=0.0, sh_close=1.0, selected=False, visible_objects_only=False, flatten=False, collection='', uvs=True, packuv=True, normals=True, vcolors=False, orcos=True, face_sets=False, subdiv_schema=False, apply_subdiv=False, curves_as_mesh=False, use_instancing=True, global_scale=1.0, triangulate=False, quad_method='SHORTEST_DIAGONAL', ngon_method='BEAUTY', export_hair=True, export_particles=True, export_custom_properties=True, as_background_job=False, evaluation_mode='RENDER', init_scene_frame_range=True)**

Export current scene in an Alembic archive

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **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
- **start** (*int in [-inf, inf], (optional)*) – Start Frame, Start frame of the export, use the default value to take the start frame of the current scene
- **end** (*int in [-inf, inf], (optional)*) – End Frame, End frame of the export, use the default value to take the end frame of the current scene
- **xsamples** (*int in [1, 128], (optional)*) – Transform Samples, Number of times per frame transformations are sampled
- **gsamples** (*int in [1, 128], (optional)*) – Geometry Samples, Number of times per frame object data are sampled
- **sh_open** (*float in [-1, 1], (optional)*) – Shutter Open, Time at which the shutter is open
- **sh_close** (*float in [-1, 1], (optional)*) – Shutter Close, Time at which the shutter is closed
- **selected** (*boolean, (optional)*) – Selected Objects Only, Export only selected objects

- **visible_objects_only** (*boolean, (optional)*) – Visible Objects Only, Export only objects that are visible
- **flatten** (*boolean, (optional)*) – Flatten Hierarchy, Do not preserve objects' parent/children relationship
- **collection** (*string, (optional, never None)*) – Collection
- **uvs** (*boolean, (optional)*) – UV Coordinates, Export UV coordinates
- **packuv** (*boolean, (optional)*) – Merge UVs
- **normals** (*boolean, (optional)*) – Normals, Export normals
- **vcolors** (*boolean, (optional)*) – Color Attributes, Export color attributes
- **orcos** (*boolean, (optional)*) – Generated Coordinates, Export undeformed mesh vertex coordinates
- **face_sets** (*boolean, (optional)*) – Face Sets, Export per face shading group assignments
- **subdiv_schema** (*boolean, (optional)*) – Use Subdivision Schema, Export meshes using Alembic's subdivision schema
- **apply_subdiv** (*boolean, (optional)*) – Apply Subdivision Surface, Export subdivision surfaces as meshes
- **curves_as_mesh** (*boolean, (optional)*) – Curves as Mesh, Export curves and NURBS surfaces as meshes
- **use_instancing** (*boolean, (optional)*) – Use Instancing, Export data of duplicated objects as Alembic instances; speeds up the export and c be disabled for compatibility with other software
- **global_scale** (*float in [0.0001, 1000], (optional)*) – Scale, Value by which to enlarge or shrink the objects with respect to the world's orig
- **triangulate** (*boolean, (optional)*) – Triangulate, Export polygons (quads and n-gons) as triangles
- **quad_method** (enum in Modifier Triangulate Quad Method Items, (optional)) – Quad Method, Method for splitting the quads into triangles
- **ngon_method** (enum in Modifier Triangulate Ngon Method Items, (optional)) – N-gon Method, Method for splitting the n-gons into triangles
- **export_hair** (*boolean, (optional)*) – Export Hair, Exports hair particle systems as animated curves
- **export_particles** (*boolean, (optional)*) – Export Particles, Exports non-hair particle systems
- **export_custom_properties** (*boolean, (optional)*) – Export Custom Properties, Export custom properties to Alembic .userProperties
- **as_background_job** (*boolean, (optional)*) – Run as Background Job, Enable this to run the import in the background, disable to block Blender while importing. This option is deprecated; EXECUTE this operator to run in the foreground, and INVOKE it to run as a backgroun job
- **evaluation_mode** (*enum in ['RENDER', 'VIEWPORT'], (optional)*) –

  Settings, Determines visibility of objects, modifier settings, and other areas where there are different settings for viewport and rendering

  - `RENDER` Render – Use Render settings for object visibility, modifier settings, etc.
  - `VIEWPORT` Viewport – Use Viewport settings for object visibility, modifier settings, etc.

bpy.ops.wm.**alembic_import(*, filepath='', directory='', files=None, check_existing=False, 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=False, filter_collada=False, filter_alembic=True, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, relative_path=True, display_type='DEFAULT', sort_method='', filter_glob='*.abc', scale=1.0, set_frame_range=True, validate_meshes=False, always_add_cache_reader=False, is_sequence=False, as_background_job=False)**

Load an Alembic archive

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
- **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_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
- **scale** (*float in [0.0001, 1000], (optional)*) – Scale, Value by which to enlarge or shrink the objects with respect to the world's origin
- **set_frame_range** (*boolean, (optional)*) – Set Frame Range, If checked, update scene's start and end frame to match those of the Alembic archive
- **validate_meshes** (*boolean, (optional)*) – Validate Meshes, Ensure the data is valid (when disabled, data may be imported which causes crashes displaying or editing)
- **always_add_cache_reader** (*boolean, (optional)*) – Always Add Cache Reader, Add cache modifiers and constraints to imported objects even if they are not animated so that they can be updated when reloading the Alembic archive
- **is_sequence** (*boolean, (optional)*) – Is Sequence, Set to true if the cache is split into separate files
- **as_background_job** (*boolean, (optional)*) – Run as Background Job, Enable this to run the export in the background, disable to block Blender while exporting. This option is deprecated; EXECUTE this operator to run in the foreground, and INVOKE it to run as a backgroun job

bpy.ops.wm.**append(*, filepath='', directory='', filename='', files=None, check_existing=False, filter_blender=True, 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=False, filter_folder=True, filter_blenlib=True, filemode=1, display_type='DEFAULT', sort_method='', link=False, do_reuse_local_id=False, clear_asset_data=False, autoselect=True, active_collection=True, instance_collections=False, instance_object_data=True, set_fake=False, use_recursive=True)**

Append from a Library .blend file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **directory** (*string, (optional, never None)*) – Directory, Directory of the file
- **filename** (*string, (optional, never None)*) – File Name, Name of the file
- **files** (`bpy_prop_collection` of `OperatorFileListElement`, (optional)) – Files
- **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
- **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
- **link** (*boolean, (optional)*) – Link, Link the objects or data-blocks rather than appending
- **do_reuse_local_id** (*boolean, (optional)*) – Re-Use Local Data, Try to re-use previously matching appended data-blocks instead of appending a new copy
- **clear_asset_data** (*boolean, (optional)*) – Clear Asset Data, Don't add asset meta-data or tags from the original data-block
- **autoselect** (*boolean, (optional)*) – Select, Select new objects
- **active_collection** (*boolean, (optional)*) – Active Collection, Put new objects on the active collection
- **instance_collections** (*boolean, (optional)*) – Instance Collections, Create instances for collections, rather than adding them directly to the scene
- **instance_object_data** (*boolean, (optional)*) – Instance Object Data, Create instances for object data which are not referenced by any objects
- **set_fake** (*boolean, (optional)*) – Fake User, Set "Fake User" for appended items (except objects and collections)
- **use_recursive** (*boolean, (optional)*) – Localize All, Localize all appended data, including those indirectly linked from other libraries

bpy.ops.wm.**batch_rename(*, data_type='OBJECT', data_source='SELECT', actions=None)**

Rename multiple items at once

PARAMETERS:

- **data_type** (*enum in ['OBJECT', 'COLLECTION', 'MATERIAL', 'MESH', 'CURVE', 'META', 'VOLUME', 'GPENCIL', 'ARMATURE', 'LATTICE', 'LIGHT', 'LIGHT_PROBE', 'CAMERA', 'SPEAKER', 'BONE', 'NODE', 'SEQUENCE_STRIP', 'ACTION_CLIP', 'SCENE', 'BRUSH'], (optional)*) – Type, Type of data to rename
- **data_source** (*enum in ['SELECT', 'ALL'], (optional)*) – Source
- **actions** (`bpy_prop_collection` of `BatchRenameAction`, (optional)) – actions

FILE:

startup/bl_operators/wm.py:3266

bpy.ops.wm.**blend_strings_utf8_validate()**

Check and fix all strings in current .blend file to be valid UTF-8 Unicode (needed for some old, 2.4x area files)

FILE:

startup/bl_operators/file.py:289

bpy.ops.wm.**call_asset_shelf_popover(*, name='')**

Open a predefined asset shelf in a popup

PARAMETERS:

name (*string, (optional, never None)*) – Asset Shelf Name, Identifier of the asset shelf to display

bpy.ops.wm.**call_menu(*, name='')**

Open a predefined menu

PARAMETERS:

name (*string, (optional, never None)*) – Name, Name of the menu

bpy.ops.wm.**call_menu_pie(*, name='')**

Open a predefined pie menu

PARAMETERS:

name (*string, (optional, never None)*) – Name, Name of the pie menu

bpy.ops.wm.**call_panel(*, name='', keep_open=True)**

Open a predefined panel

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the menu
- **keep_open** (*boolean, (optional)*) – Keep Open

bpy.ops.wm.**clear_recent_files(*, remove='ALL')**

Clear the recent files list

PARAMETERS:

remove (*enum in ['ALL', 'MISSING'], (optional)*) – Remove

bpy.ops.wm.**collada_export(*, filepath='', 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=False, filter_collada=True, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', filter_glob='*.dae', prop_bc_export_ui_section='main', apply_modifiers=False, export_mesh_type=0, export_mesh_type_selection='view', export_global_forward_selection='Y', export_global_up_selection='Z', apply_global_orientation=False, selected=False, include_children=False, include_armatures=False, include_shapekeys=False, deform_bones_only=False, include_animations=True, include_all_actions=True, export_animation_type_selection='sample', sampling_rate=1, keep_smooth_curves=False, keep_keyframes=False, keep_flat_curves=False, active_uv_only=False, use_texture_copies=True, triangulate=True, use_object_instantiation=True, use_blender_profile=True, sort_by_name=False, export_object_transformation_type=0, export_object_transformation_type_selection='matrix', export_animation_transformation_type=0, export_animation_transformation_type_selection='matrix', open_sim=False, limit_precision=False, keep_bind_info=False)**

Save a Collada file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **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
- **prop_bc_export_ui_section** (*enum in ['main', 'geometry', 'armature', 'animation', 'collada'], (optional)*) –
  Export Section, Only for User Interface organization
  - `main` Main – Data export section.
  - `geometry` Geom – Geometry export section.
  - `armature` Arm – Armature export section.
  - `animation` Anim – Animation export section.
  - `collada` Extra – Collada export section.
- **apply_modifiers** (*boolean, (optional)*) – Apply Modifiers, Apply modifiers to exported mesh (non destructive)
- **export_mesh_type** (*int in [-inf, inf], (optional)*) – Resolution, Modifier resolution for export
- **export_mesh_type_selection** (*enum in ['view', 'render'], (optional)*) –
  Resolution, Modifier resolution for export
  - `view` Viewport – Apply modifier's viewport settings.
  - `render` Render – Apply modifier's render settings.
- **export_global_forward_selection** (*enum in ['X', 'Y', 'Z', '-X', '-Y', '-Z'], (optional)*) –
  Global Forward Axis, Global Forward axis for export
  - `X` X – Global Forward is positive X Axis.
  - `Y` Y – Global Forward is positive Y Axis.
  - `Z` Z – Global Forward is positive Z Axis.
  - `-X` -X – Global Forward is negative X Axis.
  - `-Y` -Y – Global Forward is negative Y Axis.
  - `-Z` -Z – Global Forward is negative Z Axis.
- **export_global_up_selection** (*enum in ['X', 'Y', 'Z', '-X', '-Y', '-Z'], (optional)*) –
  Global Up Axis, Global Up axis for export
  - `X` X – Global UP is positive X Axis.
  - `Y` Y – Global UP is positive Y Axis.
  - `Z` Z – Global UP is positive Z Axis.
  - `-X` -X – Global UP is negative X Axis.
  - `-Y` -Y – Global UP is negative Y Axis.
  - `-Z` -Z – Global UP is negative Z Axis.

[illegible]

- **apply_global_orientation** (*boolean, (optional)*) – Apply Global Orientation, Rotate all root objects to match the global orientation settings otherwise set the global orientation per Collada asset
- **selected** (*boolean, (optional)*) – Selection Only, Export only selected elements
- **include_children** (*boolean, (optional)*) – Include Children, Export all children of selected objects (even if not selected)
- **include_armatures** (*boolean, (optional)*) – Include Armatures, Export related armatures (even if not selected)
- **include_shapekeys** (*boolean, (optional)*) – Include Shape Keys, Export all Shape Keys from Mesh Objects
- **deform_bones_only** (*boolean, (optional)*) – Deform Bones Only, Only export deforming bones with armatures
- **include_animations** (*boolean, (optional)*) – Include Animations, Export animations if available (exporting animations will enforce the decomposition of node transforms into <translation> <rotation> and <scale> components)
- **include_all_actions** (*boolean, (optional)*) – Include all Actions, Export also unassigned actions (this allows you to export entire animation libraries for your character(s))
- **export_animation_type_selection** (*enum in ['sample', 'keys'], (optional)*) –

  Key Type, Type for exported animations (use sample keys or Curve keys)

  - `sample` Samples – Export Sampled points guided by sampling rate.
  - `keys` Curves – Export Curves (note: guided by curve keys).
- **sampling_rate** (*int in [1, inf], (optional)*) – Sampling Rate, The distance between 2 keyframes (1 to key every frame)
- **keep_smooth_curves** (*boolean, (optional)*) – Keep Smooth curves, Export also the curve handles (if available) (this does only work when the inverse parent matrix is the unity matrix, otherwise you may end up with odd results)
- **keep_keyframes** (*boolean, (optional)*) – Keep Keyframes, Use existing keyframes as additional sample points (this helps when you want t keep manual tweaks)
- **keep_flat_curves** (*boolean, (optional)*) – All Keyed Curves, Export also curves which have only one key or are totally flat
- **active_uv_only** (*boolean, (optional)*) – Only Selected UV Map, Export only the selected UV Map
- **use_texture_copies** (*boolean, (optional)*) – Copy, Copy textures to same folder where the .dae file is exported
- **triangulate** (*boolean, (optional)*) – Triangulate, Export polygons (quads and n-gons) as triangles
- **use_object_instantiation** (*boolean, (optional)*) – Use Object Instances, Instantiate multiple Objects from same Data
- **use_blender_profile** (*boolean, (optional)*) – Use Blender Profile, Export additional Blender specific information (for material, shaders, bon etc.)
- **sort_by_name** (*boolean, (optional)*) – Sort by Object name, Sort exported data by Object name
- **export_object_transformation_type** (*int in [-inf, inf], (optional)*) – Transform, Object Transformation type for translation, scale and rotation
- **export_object_transformation_type_selection** (*enum in ['matrix', 'decomposed'], (optional)*) –

  Transform, Object Transformation type for translation, scale and rotation

  - `matrix` Matrix – Use <matrix> representation for exported transformations.
  - `decomposed` Decomposed – Use <rotate>, <translate> and <scale> representation for exported transformations.
- **export_animation_transformation_type** (*int in [-inf, inf], (optional)*) – Transform, Transformation type for translation, scale and rotation Note: The Animation transformation type in the Anim Tab is always equal to the Object transformation type in the Geom tab
- **export_animation_transformation_type_selection** (*enum in ['matrix', 'decomposed'], (optional)*) –

  Transform, Transformation type for translation, scale and rotation. Note: The Animation transformation type in the Anim Tab is always equal t the Object transformation type in the Geom tab

  - `matrix` Matrix – Use <matrix> representation for exported transformations.
  - `decomposed` Decomposed – Use <rotate>, <translate> and <scale> representation for exported transformations.
- **open_sim** (*boolean, (optional)*) – Export to SL/OpenSim, Compatibility mode for Second Life, OpenSimulator and other compatible online worlds
- **limit_precision** (*boolean, (optional)*) – Limit Precision, Reduce the precision of the exported data to 6 digits
- **keep_bind_info** (*boolean, (optional)*) – Keep Bind Info, Store Bindpose information in custom bone properties for later use during Collada export

bpy.ops.wm.**collada_import(*, filepath='', check_existing=False, 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=False, filter_collada=True, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', filter_glob='*.dae', import_units=False, custom_normals=True, fix_orientation=False, find_chains=False, auto_connect=False, min_chain_length=0, keep_bind_info=False)

Load a Collada file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **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
- **import_units** (*boolean, (optional)*) – Import Units, If disabled match import to Blender's current Unit settings, otherwise use the settings fro the Imported scene
- **custom_normals** (*boolean, (optional)*) – Custom Normals, Import custom normals, if available (otherwise Blender will compute them)
- **fix_orientation** (*boolean, (optional)*) – Fix Leaf Bones, Fix Orientation of Leaf Bones (Collada does only support Joints)
- **find_chains** (*boolean, (optional)*) – Find Bone Chains, Find best matching Bone Chains and ensure bones in chain are connected
- **auto_connect** (*boolean, (optional)*) – Auto Connect, Set use_connect for parent bones which have exactly one child bone
- **min_chain_length** (*int in [0, inf], (optional)*) – Minimum Chain Length, When searching Bone Chains disregard chains of length below this value
- **keep_bind_info** (*boolean, (optional)*) – Keep Bind Info, Store Bindpose information in custom bone properties for later use during Collada export

bpy.ops.wm.**collection_export_all()**

Invoke all configured exporters for all collections

bpy.ops.wm.**context_collection_boolean_set(*, data_path_iter='', data_path_item='', type='TOGGLE')**

Set boolean values for a collection of items

PARAMETERS:

- **data_path_iter** (*string, (optional, never None)*) – data_path_iter, The data path relative to the context, must point to an iterable
- **data_path_item** (*string, (optional, never None)*) – data_path_item, The data path from each iterable to the value (int or float)
- **type** (*enum in ['TOGGLE', 'ENABLE', 'DISABLE'], (optional)*) – Type

FILE:

startup/bl_operators/wm.py:875

bpy.ops.wm.**context_cycle_array(*, data_path='', reverse=False)**

Set a context array value (useful for cycling the active mesh edit mode)

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **reverse** (*boolean, (optional)*) – Reverse, Cycle backwards

FILE:

startup/bl_operators/wm.py:673

bpy.ops.wm.**context_cycle_enum(*, data_path='', reverse=False, wrap=False)**

Toggle a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **reverse** (*boolean, (optional)*) – Reverse, Cycle backwards
- **wrap** (*boolean, (optional)*) – Wrap, Wrap back to the first/last values

FILE:

startup/bl_operators/wm.py:624

bpy.ops.wm.**context_cycle_int(*, data_path='', reverse=False, wrap=False)**

Set a context value (useful for cycling active material, shape keys, groups, etc.)

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **reverse** (*boolean, (optional)*) – Reverse, Cycle backwards
- **wrap** (*boolean, (optional)*) – Wrap, Wrap back to the first/last values

FILE:

startup/bl_operators/wm.py:584

bpy.ops.wm.**context_menu_enum(*, data_path='')**

Undocumented, consider contributing.

PARAMETERS:

data_path (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)

FILE:

startup/bl_operators/wm.py:703

bpy.ops.wm.**context_modal_mouse(*, data_path_iter='', data_path_item='', header_text='', input_scale=0.01, invert=False, initial_x=0)**

Adjust arbitrary values with mouse input

PARAMETERS:

- **data_path_iter** (*string, (optional, never None)*) – data_path_iter, The data path relative to the context, must point to an iterable
- **data_path_item** (*string, (optional, never None)*) – data_path_item, The data path from each iterable to the value (int or float)
- **header_text** (*string, (optional, never None)*) – Header Text, Text to display in header during scale
- **input_scale** (*float in [-inf, inf], (optional)*) – input_scale, Scale the mouse movement by this value before applying the delta
- **invert** (*boolean, (optional)*) – invert, Invert the mouse input
- **initial_x** (*int in [-inf, inf], (optional)*) – initial_x

FILE:

startup/bl_operators/wm.py:1014

bpy.ops.wm.**context_pie_enum(*, data_path='')**

Undocumented, consider contributing.

PARAMETERS:

data_path (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)

FILE:

startup/bl_operators/wm.py:735

bpy.ops.wm.**context_scale_float(*, data_path='', value=1.0)**

Scale a float context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*float in [-inf, inf], (optional)*) – Value, Assign value

FILE:

startup/bl_operators/wm.py:338

bpy.ops.wm.**context_scale_int(*, data_path='', value=1.0, always_step=True)**

Scale an int context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*float in [-inf, inf], (optional)*) – Value, Assign value
- **always_step** (*boolean, (optional)*) – Always Step, Always adjust the value by a minimum of 1 when ‘value’ is not 1.0

FILE:

startup/bl_operators/wm.py:376

bpy.ops.wm.**context_set_boolean(*, data_path='', value=True)**

Set a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*boolean, (optional)*) – Value, Assignment value

FILE:

startup/bl_operators/wm.py:267

bpy.ops.wm.**context_set_enum(*, data_path='', value='')**

Set a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*string, (optional, never None)*) – Value, Assignment value (as a string)

FILE:

startup/bl_operators/wm.py:267

bpy.ops.wm.**context_set_float(*, data_path='', value=0.0, relative=False)**

Set a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*float in [-inf, inf], (optional)*) – Value, Assignment value
- **relative** (*boolean, (optional)*) – Relative, Apply relative to the current value (delta)

FILE:

startup/bl_operators/wm.py:267

bpy.ops.wm.**context_set_id(*, data_path='', value='')**

Set a context value to an ID data-block

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*string, (optional, never None)*) – Value, Assign value

FILE:

startup/bl_operators/wm.py:817

bpy.ops.wm.**context_set_int(*, data_path='', value=0, relative=False)**

Set a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*int in [-inf, inf], (optional)*) – Value, Assign value
- **relative** (*boolean, (optional)*) – Relative, Apply relative to the current value (delta)

FILE:

startup/bl_operators/wm.py:267

bpy.ops.wm.**context_set_string(*, data_path='', value='')**

Set a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*string, (optional, never None)*) – Value, Assign value

FILE:

startup/bl_operators/wm.py:267

bpy.ops.wm.**context_set_value(*, data_path='', value='')**

Set a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value** (*string, (optional, never None)*) – Value, Assignment value (as a string)

FILE:

startup/bl_operators/wm.py:480

bpy.ops.wm.**context_toggle(*, data_path='', module='')**

Toggle a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **module** (*string, (optional, never None)*) – Module, Optionally override the context with a module

FILE:

startup/bl_operators/wm.py:504

bpy.ops.wm.**context_toggle_enum(*, data_path='', value_1='', value_2='')**

Toggle a context value

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Context Attributes, Context data-path (expanded using visible windows in the current .blend file)
- **value_1** (*string, (optional, never None)*) – Value, Toggle enum
- **value_2** (*string, (optional, never None)*) – Value, Toggle enum

FILE:

startup/bl_operators/wm.py:545

bpy.ops.wm.**debug_menu(*, debug_value=0)**

Open a popup to set the debug level

PARAMETERS:

debug_value (*int in [-32768, 32767], (optional)*) – Debug Value

bpy.ops.wm.**doc_view(*, doc_id='')**

Open online reference docs in a web browser

PARAMETERS:

doc_id (*string, (optional, never None)*) – Doc ID

FILE:

startup/bl_operators/wm.py:1358

bpy.ops.wm.**doc_view_manual(*, doc_id='')**

Load online manual

PARAMETERS:

doc_id (*string, (optional, never None)*) – Doc ID

FILE:

startup/bl_operators/wm.py:1331

bpy.ops.wm.**doc_view_manual_ui_context()**

View a context based online manual in a web browser

bpy.ops.wm.**drop_blend_file(*, filepath='')**

Undocumented, consider contributing.

PARAMETERS:

filepath (*string, (optional, never None)*) – filepath

FILE:

startup/bl_operators/wm.py:3627

bpy.ops.wm.**drop_import_file(*, directory='', files=None)**

Operator that allows file handlers to receive file drops

PARAMETERS:

- **directory** (*string, (optional, never None)*) – Directory, Directory of the file
- **files** (`bpy_prop_collection` of `OperatorFileListElement`, (optional)) – Files

bpy.ops.wm.**grease_pencil_export_pdf(*, filepath='', 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=True, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', use_fill=True, selected_object_type='ACTIVE', stroke_sample=0.0, use_uniform_width=False, frame_mode='ACTIVE')**

Export Grease Pencil to PDF

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **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_fill** (*boolean, (optional)*) – Fill, Export strokes with fill enabled
- **selected_object_type** (*enum in ['ACTIVE', 'SELECTED', 'VISIBLE'], (optional)*) – Object, Which objects to include in the export

Object, Which objects to include in the export

  - `ACTIVE` Active – Include only the active object.
  - `SELECTED` Selected – Include selected objects.
  - `VISIBLE` Visible – Include all visible objects.

- **stroke_sample** (*float in [0, 100], (optional)*) – Sampling, Precision of stroke sampling. Low values mean a more precise result, and zero disables sampling
- **use_uniform_width** (*boolean, (optional)*) – Uniform Width, Export strokes with uniform width
- **frame_mode** (*enum in ['ACTIVE', 'SELECTED', 'SCENE'], (optional)*) –

  Frames, Which frames to include in the export

  - `ACTIVE` Active – Include only active frame.
  - `SELECTED` Selected – Include selected frames.
  - `SCENE` Scene – Include all scene frames.

bpy.ops.wm.**grease_pencil_export_svg(*, filepath='', 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=True, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', use_fill=True, selected_object_type='ACTIVE', stroke_sample=0.0, use_uniform_width=False, use_clip_camera=False)**

Export Grease Pencil to SVG

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **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_fill** (*boolean, (optional)*) – Fill, Export strokes with fill enabled

- **selected_object_type** (*enum in ['ACTIVE', 'SELECTED', 'VISIBLE'], (optional)*) –
  Object, Which objects to include in the export
  - `ACTIVE` Active – Include only the active object.
  - `SELECTED` Selected – Include selected objects.
  - `VISIBLE` Visible – Include all visible objects.
- **stroke_sample** (*float in [0, 100], (optional)*) – Sampling, Precision of stroke sampling. Low values mean a more precise result, and zero disables sampling
- **use_uniform_width** (*boolean, (optional)*) – Uniform Width, Export strokes with uniform width
- **use_clip_camera** (*boolean, (optional)*) – Clip Camera, Clip drawings to camera size when exporting in camera view

bpy.ops.wm.**grease_pencil_import_svg(*, filepath='', directory='', files=None, check_existing=False, 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=Tru filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, relative_path=True, display_type='DEFAULT', sort_method='', resolution=10, scale=10.0, use_scene_unit=False)**

Import SVG into Grease Pencil

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
- **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
- **resolution** (*int in [1, 100000], (optional)*) – Resolution, Resolution of the generated strokes

- **scale** (*float in [1e-06, 1e+06], (optional)*) – Scale, Scale of the final strokes
- **use_scene_unit** (*boolean, (optional)*) – Scene Unit, Apply current scene's unit (as defined by unit scale) to imported data

bpy.ops.wm.**interface_theme_preset_add(*, name='', remove_name=False, remove_active=False)**

Add a custom theme to the preset list

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the preset, used to make the path name
- **remove_name** (*boolean, (optional)*) – remove_name
- **remove_active** (*boolean, (optional)*) – remove_active

FILE:

startup/bl_operators/presets.py:119

bpy.ops.wm.**interface_theme_preset_remove(*, name='', remove_name=False, remove_active=True)**

Remove a custom theme from the preset list

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the preset, used to make the path name
- **remove_name** (*boolean, (optional)*) – remove_name
- **remove_active** (*boolean, (optional)*) – remove_active

FILE:

startup/bl_operators/presets.py:119

bpy.ops.wm.**interface_theme_preset_save(*, name='', remove_name=False, remove_active=True)**

Save a custom theme in the preset list

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the preset, used to make the path name
- **remove_name** (*boolean, (optional)*) – remove_name
- **remove_active** (*boolean, (optional)*) – remove_active

FILE:

startup/bl_operators/presets.py:685

bpy.ops.wm.**keyconfig_preset_add(*, name='', remove_name=False, remove_active=False)**

Add a custom keymap configuration to the preset list

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the preset, used to make the path name
- **remove_name** (*boolean, (optional)*) – remove_name
- **remove_active** (*boolean, (optional)*) – remove_active

FILE:

startup/bl_operators/presets.py:119

bpy.ops.wm.**keyconfig_preset_remove(*, name='', remove_name=False, remove_active=True)**

Remove a custom keymap configuration from the preset list

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the preset, used to make the path name
- **remove_name** (*boolean, (optional)*) – remove_name
- **remove_active** (*boolean, (optional)*) – remove_active

FILE:

startup/bl_operators/presets.py:119

bpy.ops.wm.**lib_reload(*, library='', filepath='', directory='', filename='', hide_props_region=True, check_existing=False, filter_blender=True, 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=False, filter_folder=True, filter_blenlib=False, filemode=8, relative_path=True, display_type='DEFAULT', sort_method='')**

Reload the given library

PARAMETERS:

- **library** (*string, (optional, never None)*) – Library, Library to reload
- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **directory** (*string, (optional, never None)*) – Directory, Directory of the file
- **filename** (*string, (optional, never None)*) – File Name, Name of the 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
- **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

bpy.ops.wm.**lib_relocate(*, library='', filepath='', directory='', filename='', files=None, hide_props_region=True, check_existing=False, filter_blender=True, 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=False, filter_folder=True, filter_blenlib=False, filemode=8, relative_path=True, display_type='DEFAULT', sort_method='')**

Relocate the given library to one or several others

PARAMETERS:

- **library** (*string, (optional, never None)*) – Library, Library to relocate

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **directory** (*string, (optional, never None)*) – Directory, Directory of the file
- **filename** (*string, (optional, never None)*) – File Name, Name 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

bpy.ops.wm.**link(*, filepath='', directory='', filename='', files=None, check_existing=False, filter_blender=True, 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=False, filter_folder=True, filter_blenlib=True, filemode=1, relative_path=True, display_type='DEFAULT', sort_method='', link=True, do_reuse_local_id=False, clear_asset_data=False, autoselect=True, active_collection=True, instance_collections=True, instance_object_data=True)**

Link from a Library .blend file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **directory** (*string, (optional, never None)*) – Directory, Directory of the file
- **filename** (*string, (optional, never None)*) – File Name, Name of the file
- **files** (`bpy_prop_collection` of `OperatorFileListElement`, (optional)) – Files
- **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
- **link** (*boolean, (optional)*) – Link, Link the objects or data-blocks rather than appending
- **do_reuse_local_id** (*boolean, (optional)*) – Re-Use Local Data, Try to re-use previously matching appended data-blocks instead of appending a new copy
- **clear_asset_data** (*boolean, (optional)*) – Clear Asset Data, Don't add asset meta-data or tags from the original data-block
- **autoselect** (*boolean, (optional)*) – Select, Select new objects
- **active_collection** (*boolean, (optional)*) – Active Collection, Put new objects on the active collection
- **instance_collections** (*boolean, (optional)*) – Instance Collections, Create instances for collections, rather than adding them directly to the scene
- **instance_object_data** (*boolean, (optional)*) – Instance Object Data, Create instances for object data which are not referenced by any objects

bpy.ops.wm.**memory_statistics()**

Print memory statistics to the console

bpy.ops.wm.**obj_export(*, filepath='', 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', export_animation=False, start_frame=-2147483648, end_frame=2147483647, forward_axis='NEGATIVE_Z', up_axis='Y', global_scale=1.0, apply_modifiers=True, export_eval_mode='DAG_EVAL_VIEWPORT', export_selected_objects=False, export_uv=True, export_normals=True, export_colors=False, export_materials=True, export_pbr_extensions=False, path_mode='AUTO', export_triangulated_mesh=False, export_curves_as_nurbs=False, export_object_groups=False, export_material_groups=False, export_vertex_groups=False, export_smooth_groups=False, smooth_group_bitflags=False, filter_glob='*.obj;*.mtl', collection='')**

Save the scene to a Wavefront OBJ file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file

- **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
- **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
- **export_animation** (*boolean, (optional)*) – Export Animation, Export multiple frames instead of the current frame only
- **start_frame** (*int in [-inf, inf], (optional)*) – Start Frame, The first frame to be exported
- **end_frame** (*int in [-inf, inf], (optional)*) – End Frame, The last frame to be exported
- **forward_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –
  Forward Axis
  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.
- **up_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –
  Up Axis
  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.
- **global_scale** (*float in [0.0001, 10000], (optional)*) – Scale, Value by which to enlarge or shrink the objects with respect to the world's origin

- **apply_modifiers** (*boolean, (optional)*) – Apply Modifiers, Apply modifiers to exported meshes
- **export_eval_mode** (*enum in ['DAG_EVAL_RENDER', 'DAG_EVAL_VIEWPORT'], (optional)*) –

  Object Properties, Determines properties like object visibility, modifiers etc., where they differ for Render and Viewport

  - `DAG_EVAL_RENDER` Render – Export objects as they appear in render.
  - `DAG_EVAL_VIEWPORT` Viewport – Export objects as they appear in the viewport.
- **export_selected_objects** (*boolean, (optional)*) – Export Selected Objects, Export only selected objects instead of all supported objects
- **export_uv** (*boolean, (optional)*) – Export UVs
- **export_normals** (*boolean, (optional)*) – Export Normals, Export per-face normals if the face is flat-shaded, per-face-per-loop normals if smooth-shaded
- **export_colors** (*boolean, (optional)*) – Export Colors, Export per-vertex colors
- **export_materials** (*boolean, (optional)*) – Export Materials, Export MTL library. There must be a Principled-BSDF node for image textures to be exported to the MTL file
- **export_pbr_extensions** (*boolean, (optional)*) – Export Materials with PBR Extensions, Export MTL library using PBR extensions (roughness, metallic, sheen, coat, anisotropy, transmission)
- **path_mode** (*enum in ['AUTO', 'ABSOLUTE', 'RELATIVE', 'MATCH', 'STRIP', 'COPY'], (optional)*) –

  Path Mode, Method used to reference paths

  - `AUTO` Auto – Use relative paths with subdirectories only.
  - `ABSOLUTE` Absolute – Always write absolute paths.
  - `RELATIVE` Relative – Write relative paths where possible.
  - `MATCH` Match – Match absolute/relative setting with input path.
  - `STRIP` Strip – Write filename only.
  - `COPY` Copy – Copy the file to the destination path.
- **export_triangulated_mesh** (*boolean, (optional)*) – Export Triangulated Mesh, All ngons with four or more vertices will be triangulated. Meshes in the scene will not be affected. Behaves like Triangulate Modifier with ngon-method: “Beauty”, quad-method: “Shortest Diagonal”, min vertices: 4
- **export_curves_as_nurbs** (*boolean, (optional)*) – Export Curves as NURBS, Export curves in parametric form instead of exporting as mes
- **export_object_groups** (*boolean, (optional)*) – Export Object Groups, Append mesh name to object name, separated by a ‘_’
- **export_material_groups** (*boolean, (optional)*) – Export Material Groups, Generate an OBJ group for each part of a geometry using a different material
- **export_vertex_groups** (*boolean, (optional)*) – Export Vertex Groups, Export the name of the vertex group of a face. It is approximated by choosing the vertex group with the most members among the vertices of a face
- **export_smooth_groups** (*boolean, (optional)*) – Export Smooth Groups, Every smooth-shaded face is assigned group “1” and every flat-shaded face “off”
- **smooth_group_bitflags** (*boolean, (optional)*) – Generate Bitflags for Smooth Groups
- **filter_glob** (*string, (optional, never None)*) – Extension Filter
- **collection** (*string, (optional, never None)*) – Collection

bpy.ops.wm.**obj_import(*, filepath='', directory='', files=None, check_existing=False, 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', global_scale=1.0, clamp_size=0.0, forward_axis='NEGATIVE_Z', up_axis='Y', use_split_objects=True, use_split_groups=False, import_vertex_groups=False, validate_meshes=True, close_spline_loops=True, collection_separator='', filter_glob='*.obj;*.mtl')**

Load a Wavefront OBJ scene

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
- **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
- **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
- **global_scale** (*float in [0.0001, 10000], (optional)*) – Scale, Value by which to enlarge or shrink the objects with respect to the world's origin
- **clamp_size** (*float in [0, 1000], (optional)*) – Clamp Bounding Box, Resize the objects to keep bounding box under this value. Value 0 disables clamping
- **forward_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –

  Forward Axis

  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.
- **up_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –

  Up Axis

  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.
- **use_split_objects** (*boolean, (optional)*) – Split By Object, Import each OBJ 'o' as a separate object
- **use_split_groups** (*boolean, (optional)*) – Split By Group, Import each OBJ 'g' as a separate object

- **import_vertex_groups** (*boolean, (optional)*) – Vertex Groups, Import OBJ groups as vertex groups
- **validate_meshes** (*boolean, (optional)*) – Validate Meshes, Ensure the data is valid (when disabled, data may be imported which causes crashes displaying or editing)
- **close_spline_loops** (*boolean, (optional)*) – Detect Cyclic Curves, Join curve endpoints if overlapping control points are detected(if disablec no curves will be cyclic)
- **collection_separator** (*string, (optional, never None)*) – Path Separator, Character used to separate objects name into hierarchical structur
- **filter_glob** (*string, (optional, never None)*) – Extension Filter

bpy.ops.wm.**open_mainfile(*, filepath='', hide_props_region=True, check_existing=False, filter_blender=True, 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=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', load_ui=True, use_scripts=True, display_file_selector=True, state=0)**

Open a Blender 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
- **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
- **load_ui** (*boolean, (optional)*) – Load UI, Load user interface setup in the .blend file
- **use_scripts** (*boolean, (optional)*) – Trusted Source, Allow .blend file to execute scripts automatically, default available from system preferences
- **display_file_selector** (*boolean, (optional)*) – Display File Selector
- **state** (*int in [-inf, inf], (optional)*) – State

bpy.ops.wm.**operator_cheat_sheet()**

List all the operators in a text-block, useful for scripting

FILE:

startup/bl_operators/wm.py:2246

bpy.ops.wm.**operator_defaults()**

Set the active operator to its default values

bpy.ops.wm.**operator_pie_enum(*, data_path='', prop_string='')**

Undocumented, consider contributing.

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Operator, Operator name (in Python as string)
- **prop_string** (*string, (optional, never None)*) – Property, Property name (as a string)

FILE:

startup/bl_operators/wm.py:777

bpy.ops.wm.**operator_preset_add(*, name='', remove_name=False, remove_active=False, operator='')**

Add or remove an Operator Preset

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name, Name of the preset, used to make the path name
- **remove_name** (*boolean, (optional)*) – remove_name
- **remove_active** (*boolean, (optional)*) – remove_active
- **operator** (*string, (optional, never None)*) – Operator

FILE:

startup/bl_operators/presets.py:119

bpy.ops.wm.**operator_presets_cleanup(*, operator='', properties=None)**

Remove outdated operator properties from presets that may cause problems

PARAMETERS:

- **operator** (*string, (optional, never None)*) – operator
- **properties** (`bpy_prop_collection` of `OperatorFileListElement`, (optional)) – properties

FILE:

startup/bl_operators/presets.py:882

bpy.ops.wm.**owner_disable(*, owner_id='')**

Disable add-on for workspace

PARAMETERS:

owner_id (*string, (optional, never None)*) – UI Tag

FILE:

startup/bl_operators/wm.py:2294

bpy.ops.wm.**owner_enable(*, owner_id='')**

Enable add-on for workspace

PARAMETERS:

owner_id (*string, (optional, never None)*) – UI Tag

FILE:

startup/bl_operators/wm.py:2279

bpy.ops.wm.**path_open(*, filepath='')**

Open a path in a file browser

PARAMETERS:

filepath (*string, (optional, never None)*) – filepath

FILE:

startup/bl_operators/wm.py:1167

bpy.ops.wm.**ply_export(*, filepath='', 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', forward_axis='Y', up_axis='Z', global_scale=1.0, apply_modifiers=True, export_selected_objects=False, collection='', export_uv=True, export_normals=False, export_colors='SRGB', export_attributes=True, export_triangulated_mesh=False, ascii_format=False, filter_glob='*.ply')**

Save the scene to a PLY file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **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
- **forward_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) – Forward Axis
  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.

- `NEGATIVE_X` -X – Negative X axis.
- `NEGATIVE_Y` -Y – Negative Y axis.
- `NEGATIVE_Z` -Z – Negative Z axis.

- **up_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) – Up Axis

  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.

- **global_scale** (*float in [0.0001, 10000], (optional)*) – Scale, Value by which to enlarge or shrink the objects with respect to the world's origin
- **apply_modifiers** (*boolean, (optional)*) – Apply Modifiers, Apply modifiers to exported meshes
- **export_selected_objects** (*boolean, (optional)*) – Export Selected Objects, Export only selected objects instead of all supported objects
- **collection** (*string, (optional, never None)*) – Source Collection, Export only objects from this collection (and its children)
- **export_uv** (*boolean, (optional)*) – Export UVs
- **export_normals** (*boolean, (optional)*) – Export Vertex Normals, Export specific vertex normals if available, export calculated normals otherwise
- **export_colors** (*enum in ['NONE', 'SRGB', 'LINEAR'], (optional)*) – Export Vertex Colors, Export vertex color attributes

  - `NONE` None – Do not import/export color attributes.
  - `SRGB` sRGB – Vertex colors in the file are in sRGB color space.
  - `LINEAR` Linear – Vertex colors in the file are in linear color space.

- **export_attributes** (*boolean, (optional)*) – Export Vertex Attributes, Export custom vertex attributes
- **export_triangulated_mesh** (*boolean, (optional)*) – Export Triangulated Mesh, All ngons with four or more vertices will be triangulated. Meshes in the scene will not be affected. Behaves like Triangulate Modifier with ngon-method: "Beauty", quad-method: "Shortest Diagonal", min vertices: 4
- **ascii_format** (*boolean, (optional)*) – ASCII Format, Export file in ASCII format, export as binary otherwise
- **filter_glob** (*string, (optional, never None)*) – Extension Filter

bpy.ops.wm.**ply_import(*, filepath='', directory='', files=None, check_existing=False, 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', global_scale=1.0, use_scene_unit=False, forward_axis='Y', up_axis='Z', merge_verts=False, import_colors='SRGB', import_attributes=True, filter_glob='*.ply')**

Import an PLY file as an object

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
- **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
- **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
- **global_scale** (*float in [1e-06, 1e+06], (optional)*) – Scale
- **use_scene_unit** (*boolean, (optional)*) – Scene Unit, Apply current scene's unit (as defined by unit scale) to imported data
- **forward_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –
  Forward Axis
  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.
- **up_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –
  Up Axis
  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.
- **merge_verts** (*boolean, (optional)*) – Merge Vertices, Merges vertices by distance
- **import_colors** (*enum in ['NONE', 'SRGB', 'LINEAR'], (optional)*) –
  Vertex Colors, Import vertex color attributes
  - `NONE` None – Do not import/export color attributes.
  - `SRGB` sRGB – Vertex colors in the file are in sRGB color space.
  - `LINEAR` Linear – Vertex colors in the file are in linear color space.
- **import_attributes** (*boolean, (optional)*) – Vertex Attributes, Import custom vertex attributes
- **filter_glob** (*string, (optional, never None)*) – Extension Filter

bpy.ops.wm.**previews_batch_clear(*, files=None, directory='', filter_blender=True, filter_folder=True, use_scenes=True, use_collections=True, use_objects=True, use_intern_data=True, use_trusted=False, use_backups=True)**

Clear selected .blend file's previews

PARAMETERS:

- **files** (`bpy_prop_collection` of `OperatorFileListElement`, (optional)) – files
- **directory** (*string, (optional, never None)*) – directory
- **filter_blender** (*boolean, (optional)*) – filter_blender
- **filter_folder** (*boolean, (optional)*) – filter_folder
- **use_scenes** (*boolean, (optional)*) – Scenes, Clear scenes' previews
- **use_collections** (*boolean, (optional)*) – Collections, Clear collections' previews
- **use_objects** (*boolean, (optional)*) – Objects, Clear objects' previews
- **use_intern_data** (*boolean, (optional)*) – Materials & Textures, Clear 'internal' previews (materials, textures, images, etc.)
- **use_trusted** (*boolean, (optional)*) – Trusted Blend Files, Enable Python evaluation for selected files
- **use_backups** (*boolean, (optional)*) – Save Backups, Keep a backup (.blend1) version of the files when saving with cleared previews

FILE:

startup/bl_operators/file.py:204

bpy.ops.wm.**previews_batch_generate(*, files=None, directory='', filter_blender=True, filter_folder=True, use_scenes=True, use_collections=True, use_objects=True, use_intern_data=True, use_trusted=False, use_backups=True)**

Generate selected .blend file's previews

PARAMETERS:

- **files** (`bpy_prop_collection` of `OperatorFileListElement`, (optional)) – Collection of file paths with common `directory` root
- **directory** (*string, (optional, never None)*) – Root path of all files listed in `files` collection
- **filter_blender** (*boolean, (optional)*) – Show Blender files in the File Browser
- **filter_folder** (*boolean, (optional)*) – Show folders in the File Browser
- **use_scenes** (*boolean, (optional)*) – Scenes, Generate scenes' previews
- **use_collections** (*boolean, (optional)*) – Collections, Generate collections' previews
- **use_objects** (*boolean, (optional)*) – Objects, Generate objects' previews
- **use_intern_data** (*boolean, (optional)*) – Materials & Textures, Generate 'internal' previews (materials, textures, images, etc.)
- **use_trusted** (*boolean, (optional)*) – Trusted Blend Files, Enable Python evaluation for selected files
- **use_backups** (*boolean, (optional)*) – Save Backups, Keep a backup (.blend1) version of the files when saving with generated previews

FILE:

startup/bl_operators/file.py:95

bpy.ops.wm.**previews_clear(*, id_type={})**

Clear data-block previews (only for some types like objects, materials, textures, etc.)

PARAMETERS:

id_type (*enum set in {'ALL', 'GEOMETRY', 'SHADING', 'SCENE', 'COLLECTION', 'OBJECT', 'MATERIAL', 'LIGHT', 'WORLD', 'TEXTURE', 'IMAGE'}, (optional)*) –

Data-Block Type, Which data-block previews to clear

- `ALL` All Types.
- `GEOMETRY` All Geometry Types – Clear previews for scenes, collections and objects.
- `SHADING` All Shading Types – Clear previews for materials, lights, worlds, textures and images.
- `SCENE` Scenes.
- `COLLECTION` Collections.
- `OBJECT` Objects.

- `MATERIAL` Materials.
- `LIGHT` Lights.
- `WORLD` Worlds.
- `TEXTURE` Textures.
- `IMAGE` Images.

bpy.ops.wm.**previews_ensure()**

Ensure data-block previews are available and up-to-date (to be saved in .blend file, only for some types like materials, textures, etc.)

bpy.ops.wm.**properties_add(*, data_path='')**

Add your own property to the data-block

PARAMETERS:

data_path (*string, (optional, never None)*) – Property Edit, Property data_path edit

FILE:

startup/bl_operators/wm.py:2128

bpy.ops.wm.**properties_context_change(*, context='')**

Jump to a different tab inside the properties editor

PARAMETERS:

context (*string, (optional, never None)*) – Context

FILE:

startup/bl_operators/wm.py:2171

bpy.ops.wm.**properties_edit(*, data_path='', property_name='', property_type='FLOAT', is_overridable_library=False, description='', use_soft_limits=False, array_length=3, default_int=(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), min_int=-10000, max_int=10000, soft_min_int=-10000, soft_max_int=10000, step_int=1, default_bool=(False, Fals False, False, False, False, False, False, False, False), default_float=(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, 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), min_float=-10000.0, max_float=-10000.0, soft_min_float=-10000.0, soft_max_float=-10000.0, precision=3, step_float=0.1, subtype='', default_string='', id_type='OBJECT', eval_string='')**

Change a custom property's type, or adjust how it is displayed in the interface

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Property Edit, Property data_path edit
- **property_name** (*string, (optional, never None)*) – Property Name, Property name edit
- **property_type** (*enum in ['FLOAT', 'FLOAT_ARRAY', 'INT', 'INT_ARRAY', 'BOOL', 'BOOL_ARRAY', 'STRING', 'DATA_BLOCK', 'PYTHON'], (optional)*) –

  Type

  - `FLOAT` Float – A single floating-point value.
  - `FLOAT_ARRAY` Float Array – An array of floating-point values.
  - `INT` Integer – A single integer.
  - `INT_ARRAY` Integer Array – An array of integers.
  - `BOOL` Boolean – A true or false value.
  - `BOOL_ARRAY` Boolean Array – An array of true or false values.
  - `STRING` String – A string value.
  - `DATA_BLOCK` Data-Block – A data-block value.
  - `PYTHON` Python – Edit a Python value directly, for unsupported property types.
- **is_overridable_library** (*boolean, (optional)*) – Library Overridable, Allow the property to be overridden when the data-block is linked
- **description** (*string, (optional, never None)*) – Description
- **use_soft_limits** (*boolean, (optional)*) – Soft Limits, Limits the Property Value slider to a range, values outside the range must be inputted

numerically

- **array_length** (*int in [1, 32], (optional)*) – Array Length
- **default_int** (*int array of 32 items in [-inf, inf], (optional)*) – Default Value
- **min_int** (*int in [-inf, inf], (optional)*) – Min
- **max_int** (*int in [-inf, inf], (optional)*) – Max
- **soft_min_int** (*int in [-inf, inf], (optional)*) – Soft Min
- **soft_max_int** (*int in [-inf, inf], (optional)*) – Soft Max
- **step_int** (*int in [1, inf], (optional)*) – Step
- **default_bool** (*boolean array of 32 items, (optional)*) – Default Value
- **default_float** (*float array of 32 items in [-inf, inf], (optional)*) – Default Value
- **min_float** (*float in [-inf, inf], (optional)*) – Min
- **max_float** (*float in [-inf, inf], (optional)*) – Max
- **soft_min_float** (*float in [-inf, inf], (optional)*) – Soft Min
- **soft_max_float** (*float in [-inf, inf], (optional)*) – Soft Max
- **precision** (*int in [0, 8], (optional)*) – Precision
- **step_float** (*float in [0.001, inf], (optional)*) – Step
- **subtype** (*enum in [], (optional)*) – Subtype
- **default_string** (*string, (optional, never None)*) – Default Value
- **id_type** (*enum in ['ACTION', 'ARMATURE', 'BRUSH', 'CACHEFILE', 'CAMERA', 'COLLECTION', 'CURVE', 'CURVES', 'FONT', 'GREASEPENCIL', 'GREASEPENCIL_V3', 'IMAGE', 'KEY', 'LATTICE', 'LIBRARY', 'LIGHT', 'LIGHT_PROBE', 'LINESTYLE', 'MAS 'MATERIAL', 'MESH', 'META', 'MOVIECLIP', 'NODETREE', 'OBJECT', 'PAINTCURVE', 'PALETTE', 'PARTICLE', 'POINTCLOUD 'SCENE', 'SCREEN', 'SOUND', 'SPEAKER', 'TEXT', 'TEXTURE', 'VOLUME', 'WINDOWMANAGER', 'WORKSPACE', 'WORLD'], (optional)*) – ID Type
- **eval_string** (*string, (optional, never None)*) – Value, Python value for unsupported custom property types

FILE:

startup/bl_operators/wm.py:1861

bpy.ops.wm.**properties_edit_value(*, data_path='', property_name='', eval_string='')**

Edit the value of a custom property

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Property Edit, Property data_path edit
- **property_name** (*string, (optional, never None)*) – Property Name, Property name edit
- **eval_string** (*string, (optional, never None)*) – Value, Value for custom property types that can only be edited as a Python expression

FILE:

startup/bl_operators/wm.py:2085

bpy.ops.wm.**properties_remove(*, data_path='', property_name='')**

Internal use (edit a property data_path)

PARAMETERS:

- **data_path** (*string, (optional, never None)*) – Property Edit, Property data_path edit
- **property_name** (*string, (optional, never None)*) – Property Name, Property name edit

FILE:

startup/bl_operators/wm.py:2185

bpy.ops.wm.**quit_blender()**

Quit Blender

bpy.ops.wm.**radial_control(*, data_path_primary='', data_path_secondary='', use_secondary='', rotation_path='', color_path='', fill_color_path='', fill_color_override_path='', fill_color_override_test_path='', zoom_path='', image_id='', secondary_tex=False,**

release_confirm=False)

Set some size property (e.g. brush size) with mouse wheel

PARAMETERS:

- **data_path_primary** (*string, (optional, never None)*) – Primary Data Path, Primary path of property to be set by the radial control
- **data_path_secondary** (*string, (optional, never None)*) – Secondary Data Path, Secondary path of property to be set by the radial control
- **use_secondary** (*string, (optional, never None)*) – Use Secondary, Path of property to select between the primary and secondary data patl
- **rotation_path** (*string, (optional, never None)*) – Rotation Path, Path of property used to rotate the texture display
- **color_path** (*string, (optional, never None)*) – Color Path, Path of property used to set the color of the control
- **fill_color_path** (*string, (optional, never None)*) – Fill Color Path, Path of property used to set the fill color of the control
- **fill_color_override_path** (*string, (optional, never None)*) – Fill Color Override Path
- **fill_color_override_test_path** (*string, (optional, never None)*) – Fill Color Override Test
- **zoom_path** (*string, (optional, never None)*) – Zoom Path, Path of property used to set the zoom level for the control
- **image_id** (*string, (optional, never None)*) – Image ID, Path of ID that is used to generate an image for the control
- **secondary_tex** (*boolean, (optional)*) – Secondary Texture, Tweak brush secondary/mask texture
- **release_confirm** (*boolean, (optional)*) – Confirm On Release, Finish operation on key release

bpy.ops.wm.**read_factory_settings(*, use_factory_startup_app_template_only=False, app_template='Template', use_empty=False)**

Load factory default startup file and preferences. To make changes permanent, use “Save Startup File” and “Save Preferences”

PARAMETERS:

- **use_factory_startup_app_template_only** (*boolean, (optional)*) – Factory Startup App-Template Only
- **use_empty** (*boolean, (optional)*) – Empty, After loading, remove everything except scenes, windows, and workspaces. This makes it possil to load the startup file with its scene configuration and window layout intact, but no objects, materials, animations, …

bpy.ops.wm.**read_factory_userpref(*, use_factory_startup_app_template_only=False)**

Load factory default preferences. To make changes to preferences permanent, use “Save Preferences”

PARAMETERS:

use_factory_startup_app_template_only (*boolean, (optional)*) – Factory Startup App-Template Only

bpy.ops.wm.**read_history()**

Reloads history and bookmarks

bpy.ops.wm.**read_homefile(*, filepath='', load_ui=True, use_splash=False, use_factory_startup=False, use_factory_startup_app_template_only=False, app_template='Template', use_empty=False)**

Open the default file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to an alternative start-up file
- **load_ui** (*boolean, (optional)*) – Load UI, Load user interface setup from the .blend file
- **use_splash** (*boolean, (optional)*) – Splash
- **use_factory_startup** (*boolean, (optional)*) – Factory Startup, Load the default (‘factory startup’) blend file. This is independent of the norm start-up file that the user can save
- **use_factory_startup_app_template_only** (*boolean, (optional)*) – Factory Startup App-Template Only
- **use_empty** (*boolean, (optional)*) – Empty, After loading, remove everything except scenes, windows, and workspaces. This makes it possil to load the startup file with its scene configuration and window layout intact, but no objects, materials, animations, …

bpy.ops.wm.**read_userpref()**

Load last saved preferences

bpy.ops.wm.**recover_auto_save(*, filepath='', hide_props_region=True, check_existing=False, filter_blender=True, 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=False, filter_folder=False, filter_blenlib=False, filemode=8, display_type='LIST_VERTICAL', sort_method='', use_scripts=True)

Open an automatically saved file to recover it

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
- **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_scripts** (*boolean, (optional)*) – Trusted Source, Allow .blend file to execute scripts automatically, default available from system preferences

bpy.ops.wm.**recover_last_session(*, use_scripts=True)**

Open the last closed file ("quit.blend")

PARAMETERS:

use_scripts (*boolean, (optional)*) – Trusted Source, Allow .blend file to execute scripts automatically, default available from system preferences

bpy.ops.wm.**redraw_timer(*, type='DRAW', iterations=10, time_limit=0.0)**

Simple redraw timer to test the speed of updating the interface

PARAMETERS:

- **type** (*enum in ['DRAW', 'DRAW_SWAP', 'DRAW_WIN', 'DRAW_WIN_SWAP', 'ANIM_STEP', 'ANIM_PLAY', 'UNDO'], (optional)*) – Type
  - `DRAW` Draw Region – Draw region.
  - `DRAW_SWAP` Draw Region & Swap – Draw region and swap.

    - `DRAW_SWAP` Draw Region & Swap – Draw region and swap.
    - `DRAW_WIN` Draw Window – Draw window.
    - `DRAW_WIN_SWAP` Draw Window & Swap – Draw window and swap.
    - `ANIM_STEP` Animation Step – Animation steps.
    - `ANIM_PLAY` Animation Play – Animation playback.
    - `UNDO` Undo/Redo – Undo and redo.

- **iterations** (*int in [1, inf], (optional)*) – Iterations, Number of times to redraw
- **time_limit** (*float in [0, inf], (optional)*) – Time Limit, Seconds to run the test for (override iterations)

bpy.ops.wm.**revert_mainfile(*, use_scripts=True)**

Reload the saved file

PARAMETERS:

use_scripts (*boolean, (optional)*) – Trusted Source, Allow .blend file to execute scripts automatically, default available from system preferences

bpy.ops.wm.**save_as_mainfile(*, filepath='', hide_props_region=True, check_existing=True, filter_blender=True, 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=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', compress=False, relative_remap=True, copy=False)**

Save the current file in the desired location

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
- **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
- **compress** (*boolean, (optional)*) – Compress, Write compressed .blend file
- **relative_remap** (*boolean, (optional)*) – Remap Relative, Remap relative paths when saving to a different directory
- **copy** (*boolean, (optional)*) – Save Copy, Save a copy of the actual working state but does not make saved file active

bpy.ops.wm.**save_homefile()**

Make the current file the default startup file

bpy.ops.wm.**save_mainfile(*, filepath='', hide_props_region=True, check_existing=True, filter_blender=True, 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=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', compress=False, relative_remap=False, exit=False, incremental=False)**

Save the current Blender 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 fil
- **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
- **compress** (*boolean, (optional)*) – Compress, Write compressed .blend file
- **relative_remap** (*boolean, (optional)*) – Remap Relative, Remap relative paths when saving to a different directory
- **exit** (*boolean, (optional)*) – Exit, Exit Blender after saving
- **incremental** (*boolean, (optional)*) – Incremental, Save the current Blender file with a numerically incremented name that does not overwrite any existing files

bpy.ops.wm.**save_userpref()**

Make the current preferences default

bpy.ops.wm.**search_menu()**

Pop-up a search over all menus in the current context

bpy.ops.wm.**search_operator()**

Pop-up a search over all available operators in current context

bpy.ops.wm.**search_single_menu(*, menu_idname='', initial_query='')**

Pop-up a search for a menu in current context

PARAMETERS:

- **menu_idname** (*string, (optional, never None)*) – Menu Name, Menu to search in
- **initial_query** (*string, (optional, never None)*) – Initial Query, Query to insert into the search box

bpy.ops.wm.**set_stereo_3d(*, display_mode='ANAGLYPH', anaglyph_type='RED_CYAN', interlace_type='ROW_INTERLEAVED', use_interlace_swap=False, use_sidebyside_crosseyed=False)**

Toggle 3D stereo support for current window (or change the display mode)

PARAMETERS:

- **display_mode** (enum in Stereo3D Display Items, (optional)) – Display Mode
- **anaglyph_type** (enum in Stereo3D Anaglyph Type Items, (optional)) – Anaglyph Type
- **interlace_type** (enum in Stereo3D Interlace Type Items, (optional)) – Interlace Type
- **use_interlace_swap** (*boolean, (optional)*) – Swap Left/Right, Swap left and right stereo channels
- **use_sidebyside_crosseyed** (*boolean, (optional)*) – Cross-Eyed, Right eye should see left image and vice versa

bpy.ops.wm.**splash()**

Open the splash screen with release info

bpy.ops.wm.**splash_about()**

Open a window with information about Blender

bpy.ops.wm.**stl_export(*, filepath='', 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', ascii_format=False, use_batch=False, export_selected_objects=False, collection='', global_scale=1.0, use_scene_unit=False, forward_axis='Y', up_axis='Z', apply_modifiers=True, filter_glob='*.stl')**

Save the scene to an STL file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **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
- **ascii_format** (*boolean, (optional)*) – ASCII Format, Export file in ASCII format, export as binary otherwise
- **use_batch** (*boolean, (optional)*) – Batch Export, Export each object to a separate file
- **export_selected_objects** (*boolean, (optional)*) – Export Selected Objects, Export only selected objects instead of all supported objects
- **collection** (*string, (optional, never None)*) – Source Collection, Export only objects from this collection (and its children)
- **global_scale** (*float in [1e-06, 1e+06], (optional)*) – Scale
- **use_scene_unit** (*boolean, (optional)*) – Scene Unit, Apply current scene's unit (as defined by unit scale) to exported data
- **forward_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –
  Forward Axis
  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.
- **up_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –
  Up Axis
  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.
- **apply_modifiers** (*boolean, (optional)*) – Apply Modifiers, Apply modifiers to exported meshes
- **filter_glob** (*string, (optional, never None)*) – Extension Filter

bpy.ops.wm.**stl_import(*, filepath='', directory='', files=None, check_existing=False, 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=False, filter_collada=False, filter_alembic=False, filter_usd=False, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', global_scale=1.0, use_scene_unit=False, use_facet_normal=False, forward_axis='Y', up_axis='Z', use_mesh_validate=True, filter_glob='*.stl')**

Import an STL file as an object

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
- **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
- **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
- **global_scale** (*float in [1e-06, 1e+06], (optional)*) – Scale
- **use_scene_unit** (*boolean, (optional)*) – Scene Unit, Apply current scene's unit (as defined by unit scale) to imported data
- **use_facet_normal** (*boolean, (optional)*) – Facet Normals, Use (import) facet normals (note that this will still give flat shading)
- **forward_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –
  Forward Axis

  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.

- **up_axis** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –
  Up Axis

  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.

- `NEGATIVE_Y` -Y – Negative Y axis.
- `NEGATIVE_Z` -Z – Negative Z axis.

- **use_mesh_validate** (*boolean, (optional)*) – Validate Mesh, Ensure the data is valid (when disabled, data may be imported which causes crashes displaying or editing)
- **filter_glob** (*string, (optional, never None)*) – Extension Filter

bpy.ops.wm.**sysinfo(*, filepath='')**

Generate system information, saved into a text file

PARAMETERS:

filepath (*string, (optional, never None)*) – filepath

FILE:

startup/bl_operators/wm.py:2214

bpy.ops.wm.**tool_set_by_brush_type(*, brush_type='', space_type='EMPTY')**

Look up the most appropriate tool for the given brush type and activate that

PARAMETERS:

- **brush_type** (*string, (optional, never None)*) – Brush Type, Brush type identifier for which the most appropriate tool will be looked up
- **space_type** (*enum in ['EMPTY', 'VIEW_3D', 'IMAGE_EDITOR', 'NODE_EDITOR', 'SEQUENCE_EDITOR', 'CLIP_EDITOR', 'DOPESHEET_EDITOR', 'GRAPH_EDITOR', 'NLA_EDITOR', 'TEXT_EDITOR', 'CONSOLE', 'INFO', 'TOPBAR', 'STATUSBAR', 'OUTLINER', 'PROPERTIES', 'FILE_BROWSER', 'SPREADSHEET', 'PREFERENCES'], (optional)*) – Type

FILE:

startup/bl_operators/wm.py:2428

bpy.ops.wm.**tool_set_by_id(*, name='', cycle=False, as_fallback=False, space_type='EMPTY')**

Set the tool by name (for key-maps)

PARAMETERS:

- **name** (*string, (optional, never None)*) – Identifier, Identifier of the tool
- **cycle** (*boolean, (optional)*) – Cycle, Cycle through tools in this group
- **as_fallback** (*boolean, (optional)*) – Set Fallback, Set the fallback tool instead of the primary tool
- **space_type** (*enum in ['EMPTY', 'VIEW_3D', 'IMAGE_EDITOR', 'NODE_EDITOR', 'SEQUENCE_EDITOR', 'CLIP_EDITOR', 'DOPESHEET_EDITOR', 'GRAPH_EDITOR', 'NLA_EDITOR', 'TEXT_EDITOR', 'CONSOLE', 'INFO', 'TOPBAR', 'STATUSBAR', 'OUTLINER', 'PROPERTIES', 'FILE_BROWSER', 'SPREADSHEET', 'PREFERENCES'], (optional)*) – Type

FILE:

startup/bl_operators/wm.py:2337

bpy.ops.wm.**tool_set_by_index(*, index=0, cycle=False, expand=True, as_fallback=False, space_type='EMPTY')**

Set the tool by index (for key-maps)

PARAMETERS:

- **index** (*int in [-inf, inf], (optional)*) – Index in Toolbar
- **cycle** (*boolean, (optional)*) – Cycle, Cycle through tools in this group
- **expand** (*boolean, (optional)*) – expand, Include tool subgroups
- **as_fallback** (*boolean, (optional)*) – Set Fallback, Set the fallback tool instead of the primary
- **space_type** (*enum in ['EMPTY', 'VIEW_3D', 'IMAGE_EDITOR', 'NODE_EDITOR', 'SEQUENCE_EDITOR', 'CLIP_EDITOR', 'DOPESHEET_EDITOR', 'GRAPH_EDITOR', 'NLA_EDITOR', 'TEXT_EDITOR', 'CONSOLE', 'INFO', 'TOPBAR', 'STATUSBAR', 'OUTLINER', 'PROPERTIES', 'FILE_BROWSER', 'SPREADSHEET', 'PREFERENCES'], (optional)*) – Type

FILE:

startup/bl_operators/wm.py:2387

bpy.ops.wm.**toolbar()**

Undocumented, consider contributing.

FILE:

startup/bl_operators/wm.py:2495

bpy.ops.wm.**toolbar_fallback_pie()**

Undocumented, consider contributing.

FILE:

startup/bl_operators/wm.py:2519

bpy.ops.wm.**toolbar_prompt()**

Leader key like functionality for accessing tools

FILE:

startup/bl_operators/wm.py:2619

bpy.ops.wm.**url_open(*, url='')**

Open a website in the web browser

PARAMETERS:

url (*string, (optional, never None)*) – URL, URL to open

FILE:

startup/bl_operators/wm.py:1074

bpy.ops.wm.**url_open_preset(*, type='')**

Open a preset website in the web browser

PARAMETERS:

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

FILE:

startup/bl_operators/wm.py:1144

bpy.ops.wm.**usd_export(*, filepath='', 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=False, filter_collada=False, filter_alembic=False, filter_usd=True, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, display_type='DEFAULT', sort_method='', filter_glob='*.usd', selected_objects_only=False, visible_objects_only=True, collection='', export_animation=False, export_hair=False, export_uvmaps=True, rename_uvmaps=True, export_mesh_colors=True, export_normals=True, export_materials=True, export_subdivision='BEST_MATCH', export_armatures=True, only_deform_bones=False, export_shapekeys=True, use_instancing=False, evaluation_mode='RENDER', generate_preview_surface=True, generate_materialx_network=False, convert_orientation=False, export_global_forward_selection='NEGATIVE_Z', export_global_up_selection='Y', export_textures=False, export_textures_mode='NEW', overwrite_textures=False, relative_paths=True, xform_op_mode='TRS', root_prim_path='/root', export_custom_properties=True, custom_properties_namespace='userProperties', author_blender_name=True, convert_world_material=True, allow_unicode=False, export_meshes=True, export_lights=True, export_cameras=True, export_curves=True, export_points=True, export_volumes=True, triangulate_meshes=False, quad_method='SHORTEST_DIAGONAL', ngon_method='BEAUTY', usdz_downscale_size='KEEP', usdz_downscale_custom_size=128, merge_parent_xform=False, convert_scene_units='METERS', meters_per_unit=1.0)**

Export current scene in a USD archive

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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_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
- **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
- **selected_objects_only** (*boolean, (optional)*) – Selection Only, Only export selected objects. Unselected parents of selected objects are exported as empty transform
- **visible_objects_only** (*boolean, (optional)*) – Visible Only, Only export visible objects. Invisible parents of exported objects are exported a empty transforms
- **collection** (*string, (optional, never None)*) – Collection
- **export_animation** (*boolean, (optional)*) – Animation, Export all frames in the render frame range, rather than only the current frame
- **export_hair** (*boolean, (optional)*) – Hair, Export hair particle systems as USD curves
- **export_uvmaps** (*boolean, (optional)*) – UV Maps, Include all mesh UV maps in the export
- **rename_uvmaps** (*boolean, (optional)*) – Rename UV Maps, Rename active render UV map to "st" to match USD conventions
- **export_mesh_colors** (*boolean, (optional)*) – Color Attributes, Include mesh color attributes in the export
- **export_normals** (*boolean, (optional)*) – Normals, Include normals of exported meshes in the export
- **export_materials** (*boolean, (optional)*) – Materials, Export viewport settings of materials as USD preview materials, and export material assignments as geometry subsets
- **export_subdivision** (*enum in ['IGNORE', 'TESSELLATE', 'BEST_MATCH'], (optional)*) –

  Subdivision, Choose how subdivision modifiers will be mapped to the USD subdivision scheme during export

  - `IGNORE` Ignore – Scheme = None. Export base mesh without subdivision.
  - `TESSELLATE` Tessellate – Scheme = None. Export subdivided mesh.
  - `BEST_MATCH` Best Match – Scheme = Catmull-Clark, when possible. Reverts to exporting the subdivided mesh for the Simple subdivision type.

- **export_armatures** (*boolean, (optional)*) – Armatures, Export armatures and meshes with armature modifiers as USD skeletons and skinne meshes
- **only_deform_bones** (*boolean, (optional)*) – Only Deform Bones, Only export deform bones and their parents
- **export_shapekeys** (*boolean, (optional)*) – Shape Keys, Export shape keys as USD blend shapes
- **use_instancing** (*boolean, (optional)*) – Instancing, Export instanced objects as references in USD rather than real objects
- **evaluation_mode** (*enum in ['RENDER', 'VIEWPORT'], (optional)*) –

- **evaluation_mode** (*enum in ['RENDER', 'VIEWPORT'], (optional)*) –

  Use Settings for, Determines visibility of objects, modifier settings, and other areas where there are different settings for viewport and renderin

  - `RENDER` Render – Use Render settings for object visibility, modifier settings, etc.
  - `VIEWPORT` Viewport – Use Viewport settings for object visibility, modifier settings, etc.

- **generate_preview_surface** (*boolean, (optional)*) – USD Preview Surface Network, Generate an approximate USD Preview Surface shac representation of a Principled BSDF node network
- **generate_materialx_network** (*boolean, (optional)*) – MaterialX Network, Generate a MaterialX network representation of the materials
- **convert_orientation** (*boolean, (optional)*) – Convert Orientation, Convert orientation axis to a different convention to match other applications
- **export_global_forward_selection** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –

  Forward Axis

  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.

- **export_global_up_selection** (*enum in ['X', 'Y', 'Z', 'NEGATIVE_X', 'NEGATIVE_Y', 'NEGATIVE_Z'], (optional)*) –

  Up Axis

  - `X` X – Positive X axis.
  - `Y` Y – Positive Y axis.
  - `Z` Z – Positive Z axis.
  - `NEGATIVE_X` -X – Negative X axis.
  - `NEGATIVE_Y` -Y – Negative Y axis.
  - `NEGATIVE_Z` -Z – Negative Z axis.

- **export_textures** (*boolean, (optional)*) – Export Textures, If exporting materials, export textures referenced by material nodes to a 'textures directory in the same directory as the USD file
- **export_textures_mode** (*enum in ['KEEP', 'PRESERVE', 'NEW'], (optional)*) –

  Export Textures, Texture export method

  - `KEEP` Keep – Use original location of textures.
  - `PRESERVE` Preserve – Preserve file paths of textures from already imported USD files. Export remaining textures to a 'textures' folder next to the USD file.
  - `NEW` New Path – Export textures to a 'textures' folder next to the USD file.

- **overwrite_textures** (*boolean, (optional)*) – Overwrite Textures, Overwrite existing files when exporting textures
- **relative_paths** (*boolean, (optional)*) – Relative Paths, Use relative paths to reference external files (i.e. textures, volumes) in USD, otherwis use absolute paths
- **xform_op_mode** (*enum in ['TRS', 'TOS', 'MAT'], (optional)*) –

  Xform Ops, The type of transform operators to write

  - `TRS` Translate, Rotate, Scale – Export with translate, rotate, and scale Xform operators.
  - `TOS` Translate, Orient, Scale – Export with translate, orient quaternion, and scale Xform operators.
  - `MAT` Matrix – Export matrix operator.

- **root_prim_path** (*string, (optional, never None)*) – Root Prim, If set, add a transform primitive with the given path to the stage as the parent of all exported data
- **export_custom_properties** (*boolean, (optional)*) – Custom Properties, Export custom properties as USD attributes
- **custom_properties_namespace** (*string, (optional, never None)*) – Namespace, If set, add the given namespace as a prefix to exported custom property names. This only applies to property names that do not already have a prefix (e.g., it would apply to name 'bar' but not 'foo:bar') and does not apply to blender object and data names which are always exported in the 'userProperties:blender' namespace

- **author_blender_name** (*boolean, (optional)*) – Blender Names, Author USD custom attributes containing the original Blender object and object data names
- **convert_world_material** (*boolean, (optional)*) – World Dome Light, Convert the world material to a USD dome light. Currently works for simple materials, consisting of an environment texture connected to a background shader, with an optional vector multiply of the texture color
- **allow_unicode** (*boolean, (optional)*) – Allow Unicode, Preserve UTF-8 encoded characters when writing USD prim and property names (requires software utilizing USD 24.03 or greater when opening the resulting files)
- **export_meshes** (*boolean, (optional)*) – Meshes, Export all meshes
- **export_lights** (*boolean, (optional)*) – Lights, Export all lights
- **export_cameras** (*boolean, (optional)*) – Cameras, Export all cameras
- **export_curves** (*boolean, (optional)*) – Curves, Export all curves
- **export_points** (*boolean, (optional)*) – Point Clouds, Export all point clouds
- **export_volumes** (*boolean, (optional)*) – Volumes, Export all volumes
- **triangulate_meshes** (*boolean, (optional)*) – Triangulate Meshes, Triangulate meshes during export
- **quad_method** (enum in Modifier Triangulate Quad Method Items, (optional)) – Quad Method, Method for splitting the quads into triangles
- **ngon_method** (enum in Modifier Triangulate Ngon Method Items, (optional)) – N-gon Method, Method for splitting the n-gons into triangles
- **usdz_downscale_size** (*enum in ['KEEP', '256', '512', '1024', '2048', '4096', 'CUSTOM'], (optional)*) –
  USDZ Texture Downsampling, Choose a maximum size for all exported textures
  - `KEEP` Keep – Keep all current texture sizes.
  - `256` 256 – Resize to a maximum of 256 pixels.
  - `512` 512 – Resize to a maximum of 512 pixels.
  - `1024` 1024 – Resize to a maximum of 1024 pixels.
  - `2048` 2048 – Resize to a maximum of 2048 pixels.
  - `4096` 4096 – Resize to a maximum of 4096 pixels.
  - `CUSTOM` Custom – Specify a custom size.
- **usdz_downscale_custom_size** (*int in [64, 16384], (optional)*) – USDZ Custom Downscale Size, Custom size for downscaling exported textures
- **merge_parent_xform** (*boolean, (optional)*) – Merge parent Xform, Merge USD primitives with their Xform parent if possible. USD does ı allow nested UsdGeomGprims, intermediary Xform prims will be defined to keep the USD file valid when encountering object hierarchies.
- **convert_scene_units** (*enum in ['METERS', 'KILOMETERS', 'CENTIMETERS', 'MILLIMETERS', 'INCHES', 'FEET', 'YARDS', 'CUSTOM'], (optional)*) –
  Units, Set the USD Stage meters per unit to the chosen measurement, or a custom value
  - `METERS` Meters – Scene meters per unit to 1.0.
  - `KILOMETERS` Kilometers – Scene meters per unit to 1000.0.
  - `CENTIMETERS` Centimeters – Scene meters per unit to 0.01.
  - `MILLIMETERS` Millimeters – Scene meters per unit to 0.001.
  - `INCHES` Inches – Scene meters per unit to 0.0254.
  - `FEET` Feet – Scene meters per unit to 0.3048.
  - `YARDS` Yards – Scene meters per unit to 0.9144.
  - `CUSTOM` Custom – Specify a custom scene meters per unit value.
- **meters_per_unit** (*float in [0.0001, 1000], (optional)*) – Meters Per Unit, Custom value for meters per unit in the USD Stage

bpy.ops.wm.**usd_import(*, filepath='', check_existing=False, 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=False, filter_collada=False, filter_alembic=False, filter_usd=True, filter_obj=False, filter_volume=False, filter_folder=True, filter_blenlib=False, filemode=8, relative_path=True, display_type='DEFAULT', sort_method='', filter_glob='*.usd', scale=1.0, set_frame_range=True, import_cameras=True, import_curves=True, import_lights=True, import_materials=True, import_meshes=True, import_volumes=True, import_shapes=True, import_skeletons=True, import_blendshapes=True, import_points=True, import_subdiv=False, support_scene_instancing=True, import_visible_only=True, create_collection=False, read_mesh_uvs=True, read_mesh_colors=True, read_mesh_attributes=True, prim_path_mask='', import_guide=False, import_proxy=False, import_render=True, import_all_materials=False, import_usd_preview=True,**

set_material_blend=True, light_intensity_scale=1.0, mtl_purpose='MTL_FULL', mtl_name_collision_mode='MAKE_UNIQUE', import_textures_mode='IMPORT_PACK', import_textures_dir='//textures/', tex_name_collision_mode='USE_EXISTING', attr_import_mode='ALL', validate_meshes=False, create_world_material=True, import_defined_only=True, merge_parent_xform=True, apply_unit_conversion_scale=True)

Import USD stage into current scene

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – File Path, Path to file
- **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
- **scale** (*float in [0.0001, 1000], (optional)*) – Scale, Value by which to enlarge or shrink the objects with respect to the world's origin
- **set_frame_range** (*boolean, (optional)*) – Set Frame Range, Update the scene's start and end frame to match those of the USD archive
- **import_cameras** (*boolean, (optional)*) – Cameras
- **import_curves** (*boolean, (optional)*) – Curves
- **import_lights** (*boolean, (optional)*) – Lights
- **import_materials** (*boolean, (optional)*) – Materials
- **import_meshes** (*boolean, (optional)*) – Meshes
- **import_volumes** (*boolean, (optional)*) – Volumes
- **import_shapes** (*boolean, (optional)*) – USD Shapes
- **import_skeletons** (*boolean, (optional)*) – Armatures
- **import_blendshapes** (*boolean, (optional)*) – Shape Keys
- **import_points** (*boolean, (optional)*) – Point Clouds
- **import_subdiv** (*boolean, (optional)*) – Import Subdivision Scheme, Create subdivision surface modifiers based on the USD SubdivisionScheme attribute

- **support_scene_instancing** (*boolean, (optional)*) – Scene Instancing, Import USD scene graph instances as collection instances
- **import_visible_only** (*boolean, (optional)*) – Visible Primitives Only, Do not import invisible USD primitives. Only applies to primitives with non-animated visibility attribute. Primitives with animated visibility will always be imported
- **create_collection** (*boolean, (optional)*) – Create Collection, Add all imported objects to a new collection
- **read_mesh_uvs** (*boolean, (optional)*) – UV Coordinates, Read mesh UV coordinates
- **read_mesh_colors** (*boolean, (optional)*) – Color Attributes, Read mesh color attributes
- **read_mesh_attributes** (*boolean, (optional)*) – Mesh Attributes, Read USD Primvars as mesh attributes
- **prim_path_mask** (*string, (optional, never None)*) – Path Mask, Import only the primitive at the given path and its descendants. Multiple paths may be specified in a list delimited by commas or semicolons
- **import_guide** (*boolean, (optional)*) – Guide, Import guide geometry
- **import_proxy** (*boolean, (optional)*) – Proxy, Import proxy geometry
- **import_render** (*boolean, (optional)*) – Render, Import final render geometry
- **import_all_materials** (*boolean, (optional)*) – Import All Materials, Also import materials that are not used by any geometry. Note that whe this option is false, materials referenced by geometry will still be imported
- **import_usd_preview** (*boolean, (optional)*) – Import USD Preview, Convert UsdPreviewSurface shaders to Principled BSDF shader networks
- **set_material_blend** (*boolean, (optional)*) – Set Material Blend, If the Import USD Preview option is enabled, the material blend method w automatically be set based on the shader's opacity and opacityThreshold inputs
- **light_intensity_scale** (*float in [0.0001, 10000], (optional)*) – Light Intensity Scale, Scale for the intensity of imported lights
- **mtl_purpose** (*enum in ['MTL_ALL_PURPOSE', 'MTL_PREVIEW', 'MTL_FULL'], (optional)*) –

  Material Purpose, Attempt to import materials with the given purpose. If no material with this purpose is bound to the primitive, fall back on loading any other bound material

  - `MTL_ALL_PURPOSE` All Purpose – Attempt to import 'allPurpose' materials..
  - `MTL_PREVIEW` Preview – Attempt to import 'preview' materials. Load 'allPurpose' materials as a fallback.
  - `MTL_FULL` Full – Attempt to import 'full' materials. Load 'allPurpose' or 'preview' materials, in that order, as a fallback.
- **mtl_name_collision_mode** (*enum in ['MAKE_UNIQUE', 'REFERENCE_EXISTING'], (optional)*) –

  Material Name Collision, Behavior when the name of an imported material conflicts with an existing material

  - `MAKE_UNIQUE` Make Unique – Import each USD material as a unique Blender material.
  - `REFERENCE_EXISTING` Reference Existing – If a material with the same name already exists, reference that instead of importing.
- **import_textures_mode** (*enum in ['IMPORT_NONE', 'IMPORT_PACK', 'IMPORT_COPY'], (optional)*) –

  Import Textures, Behavior when importing textures from a USDZ archive

  - `IMPORT_NONE` None – Don't import textures.
  - `IMPORT_PACK` Packed – Import textures as packed data.
  - `IMPORT_COPY` Copy – Copy files to textures directory.
- **import_textures_dir** (*string, (optional, never None)*) – Textures Directory, Path to the directory where imported textures will be copied
- **tex_name_collision_mode** (*enum in ['USE_EXISTING', 'OVERWRITE'], (optional)*) –

  File Name Collision, Behavior when the name of an imported texture file conflicts with an existing file

  - `USE_EXISTING` Use Existing – If a file with the same name already exists, use that instead of copying.
  - `OVERWRITE` Overwrite – Overwrite existing files.
- **attr_import_mode** (*enum in ['NONE', 'USER', 'ALL'], (optional)*) –

  Custom Properties, Behavior when importing USD attributes as Blender custom properties

  - `NONE` None – Do not import USD custom attributes.
  - `USER` User – Import USD attributes in the 'userProperties' namespace as Blender custom properties. The namespace will be stripped from the property names.
  - `ALL` All Custom – Import all USD custom attributes as Blender custom properties. Namespaces will be retained in the property names.
- **validate_meshes** (*boolean, (optional)*) – Validate Meshes, Ensure the data is valid (when disabled, data may be imported which causes crashes displaying or editing)

- **create_world_material** (*boolean, (optional)*) – World Dome Light, Convert the first discovered USD dome light to a world background shader
- **import_defined_only** (*boolean, (optional)*) – Defined Primitives Only, Import only defined USD primitives. When disabled this allows importing USD primitives which are not defined, such as those with an override specifier
- **merge_parent_xform** (*boolean, (optional)*) – Merge parent Xform, Allow USD primitives to merge with their Xform parent if they are the only child in the hierarchy
- **apply_unit_conversion_scale** (*boolean, (optional)*) – Apply Unit Conversion Scale, Scale the scene objects by the USD stage's meters p unit value. This scaling is applied in addition to the value specified in the Scale option

bpy.ops.wm.**window_close()**

Close the current window

bpy.ops.wm.**window_fullscreen_toggle()**

Toggle the current window full-screen

bpy.ops.wm.**window_new()**

Create a new window

bpy.ops.wm.**window_new_main()**

Create a new main window with its own workspace and scene selection

bpy.ops.wm.**xr_navigation_fly(*, mode='VIEWER_FORWARD', lock_location_z=False, lock_direction=False, speed_frame_based=True speed_min=0.018, speed_max=0.054, speed_interpolation0=(0.0, 0.0), speed_interpolation1=(1.0, 1.0))**

Move/turn relative to the VR viewer or controller

PARAMETERS:

- **mode** (*enum in ['FORWARD', 'BACK', 'LEFT', 'RIGHT', 'UP', 'DOWN', 'TURNLEFT', 'TURNRIGHT', 'VIEWER_FORWARD', 'VIEWER_BACK', 'VIEWER_LEFT', 'VIEWER_RIGHT', 'CONTROLLER_FORWARD'], (optional)*) –

  Mode, Fly mode

  - `FORWARD` Forward – Move along navigation forward axis.
  - `BACK` Back – Move along navigation back axis.
  - `LEFT` Left – Move along navigation left axis.
  - `RIGHT` Right – Move along navigation right axis.
  - `UP` Up – Move along navigation up axis.
  - `DOWN` Down – Move along navigation down axis.
  - `TURNLEFT` Turn Left – Turn counter-clockwise around navigation up axis.
  - `TURNRIGHT` Turn Right – Turn clockwise around navigation up axis.
  - `VIEWER_FORWARD` Viewer Forward – Move along viewer's forward axis.
  - `VIEWER_BACK` Viewer Back – Move along viewer's back axis.
  - `VIEWER_LEFT` Viewer Left – Move along viewer's left axis.
  - `VIEWER_RIGHT` Viewer Right – Move along viewer's right axis.
  - `CONTROLLER_FORWARD` Controller Forward – Move along controller's forward axis.
- **lock_location_z** (*boolean, (optional)*) – Lock Elevation, Prevent changes to viewer elevation
- **lock_direction** (*boolean, (optional)*) – Lock Direction, Limit movement to viewer's initial direction
- **speed_frame_based** (*boolean, (optional)*) – Frame Based Speed, Apply fixed movement deltas every update
- **speed_min** (*float in [0, 1000], (optional)*) – Minimum Speed, Minimum move (turn) speed in meters (radians) per second or frame
- **speed_max** (*float in [0, 1000], (optional)*) – Maximum Speed, Maximum move (turn) speed in meters (radians) per second or frame
- **speed_interpolation0** (`mathutils.Vector` of 2 items in [0, 1], (optional)) – Speed Interpolation 0, First cubic spline control point between min/max speeds
- **speed_interpolation1** (`mathutils.Vector` of 2 items in [0, 1], (optional)) – Speed Interpolation 1, Second cubic spline control poi between min/max speeds

bpy.ops.wm.**xr_navigation_grab(*, lock_location=False, lock_location_z=False, lock_rotation=False, lock_rotation_z=False, lock_scale=False)**

Navigate the VR scene by grabbing with controllers

PARAMETERS:

- **lock_location** (*boolean, (optional)*) – Lock Location, Prevent changes to viewer location
- **lock_location_z** (*boolean, (optional)*) – Lock Elevation, Prevent changes to viewer elevation
- **lock_rotation** (*boolean, (optional)*) – Lock Rotation, Prevent changes to viewer rotation
- **lock_rotation_z** (*boolean, (optional)*) – Lock Up Orientation, Prevent changes to viewer up orientation
- **lock_scale** (*boolean, (optional)*) – Lock Scale, Prevent changes to viewer scale

bpy.ops.wm.**xr_navigation_reset(*, location=True, rotation=True, scale=True)**

Reset VR navigation deltas relative to session base pose

PARAMETERS:

- **location** (*boolean, (optional)*) – Location, Reset location deltas
- **rotation** (*boolean, (optional)*) – Rotation, Reset rotation deltas
- **scale** (*boolean, (optional)*) – Scale, Reset scale deltas

bpy.ops.wm.**xr_navigation_teleport(*, teleport_axes=(True, True, True), interpolation=1.0, offset=0.0, selectable_only=True, distance=1.70141e+38, from_viewer=False, axis=(0.0, 0.0, -1.0), color=(0.35, 0.35, 1.0, 1.0))**

Set VR viewer location to controller raycast hit location

PARAMETERS:

- **teleport_axes** (*boolean array of 3 items, (optional)*) – Teleport Axes, Enabled teleport axes in navigation space
- **interpolation** (*float in [0, 1], (optional)*) – Interpolation, Interpolation factor between viewer and hit locations
- **offset** (*float in [0, inf], (optional)*) – Offset, Offset along hit normal to subtract from final location
- **selectable_only** (*boolean, (optional)*) – Selectable Only, Only allow selectable objects to influence raycast result
- **distance** (*float in [0, inf], (optional)*) – Maximum raycast distance
- **from_viewer** (*boolean, (optional)*) – From Viewer, Use viewer pose as raycast origin
- **axis** (`mathutils.Vector` of 3 items in [-1, 1], (optional)) – Axis, Raycast axis in controller/viewer space
- **color** (*float array of 4 items in [0, 1], (optional)*) – Color, Raycast color

bpy.ops.wm.**xr_session_toggle()**

Open a view for use with virtual reality headsets, or close it if already opened

Workspace Operators

bpy.ops.workspace.**add()**

Add a new workspace by duplicating the current one or appending one from the user configuration

bpy.ops.workspace.**append_activate(*, idname='', filepath='')**

Append a workspace and make it the active one in the current window

PARAMETERS:

- **idname** (*string, (optional, never None)*) – Identifier, Name of the workspace to append and activate
- **filepath** (*string, (optional, never None)*) – Filepath, Path to the library

bpy.ops.workspace.**delete()**

Delete the active workspace

bpy.ops.workspace.**duplicate()**

Add a new workspace

bpy.ops.workspace.**reorder_to_back()**

Reorder workspace to be last in the list

bpy.ops.workspace.**reorder_to_front()**

Reorder workspace to be first in the list

bpy.ops.workspace.**scene_pin_toggle()**

Remember the last used scene for the current workspace and switch to it whenever this workspace is activated again

World Operators

bpy.ops.world.**convert_volume_to_mesh()**

Convert the volume of a world to a mesh. The world's volume used to be rendered by EEVEE Legacy. Conversion is needed for it to render properly

FILE:

startup/bl_operators/world.py:26

bpy.ops.world.**new()**

Create a new world Data-Block

Skip to content

Path Utilities (bpy.path)

This module has a similar scope to os.path, containing utility functions for dealing with paths in Blender.

bpy.path.**abspath(path, *, start=None, library=None)**

Returns the absolute path relative to the current blend file using the "//" prefix.

PARAMETERS:

- **start** (*str* | *bytes*) – Relative to this path, when not set the current filename is used.
- **library** (`bpy.types.Library`) – The library this path is from. This is only included for convenience, when the library is not None its path replaces *start*.

RETURNS:

The absolute path.

RETURN TYPE:

str

bpy.path.**basename(path)**

Equivalent to `os.path.basename`, but skips a "//" prefix.

Use for Windows compatibility.

RETURNS:

The base name of the given path.

RETURN TYPE:

str

bpy.path.**clean_name(name, *, replace='_')**

Returns a name with characters replaced that may cause problems under various circumstances, such as writing to a file.

All characters besides A-Z/a-z, 0-9 are replaced with "_" or the *replace* argument if defined.

PARAMETERS:

- **name** (*str* | *bytes*) – The path name.
- **replace** (*str*) – The replacement for non-valid characters.

RETURNS:

The cleaned name.

RETURN TYPE:

str

bpy.path.**display_name(name, *, has_ext=True, title_case=True)**

Creates a display string from name to be used menus and the user interface. Intended for use with filenames and module names.

PARAMETERS:

- **name** (*str*) – The name to be used for displaying the user interface.
- **has_ext** (*bool*) – Remove file extension from name.
- **title_case** (*bool*) – Convert lowercase names to title case.

RETURNS:

The display string.

RETURN TYPE:

str

bpy.path.**display_name_to_filepath(name)**

Performs the reverse of display_name using literal versions of characters which aren't supported in a filepath.

PARAMETERS:

name (*str*) – The display name to convert.

RETURNS:

The file path.

RETURN TYPE:

str

bpy.path.**display_name_from_filepath(name)**

Returns the path stripped of directory and extension, ensured to be utf8 compatible.

PARAMETERS:

name (*str*) – The file path to convert.

RETURNS:

The display name.

RETURN TYPE:

str

bpy.path.**ensure_ext(filepath, ext, *, case_sensitive=False)**

Return the path with the extension added if it is not already set.

PARAMETERS:

- **filepath** (*str*) – The file path.
- **ext** (*str*) – The extension to check for, can be a compound extension. Should start with a dot, such as '.blend' or '.tar.gz'.
- **case_sensitive** (*bool*) – Check for matching case when comparing extensions.

RETURNS:

The file path with the given extension.

RETURN TYPE:

str

bpy.path.**is_subdir(path, directory)**

Returns true if *path* in a subdirectory of *directory*. Both paths must be absolute.

PARAMETERS:

path (*str* | *bytes*) – An absolute path.

RETURNS:

Whether or not the path is a subdirectory.

RETURN TYPE:

bool

bpy.path.**module_names(path, *, recursive=False, package='')**

Return a list of modules which can be imported from *path*.

PARAMETERS:

- **path** (*str*) – a directory to scan.
- **recursive** (*bool*) – Also return submodule names for packages.
- **package** (*str*) – Optional string, used as the prefix for module names (without the trailing ".").

RETURNS:

a list of string pairs (module_name, module_file).

RETURN TYPE:

list[str]

bpy.path.**native_pathsep(path)**

Replace the path separator with the systems native `os.sep`.

PARAMETERS:

path (*str*) – The path to replace.

RETURNS:

The path with system native separators.

RETURN TYPE:

str

bpy.path.**reduce_dirs(dirs)**

Given a sequence of directories, remove duplicates and any directories nested in one of the other paths. (Useful for recursive path searching).

PARAMETERS:

dirs (*Sequence[str]*) – Sequence of directory paths.

RETURNS:

A unique list of paths.

RETURN TYPE:

list[str]

bpy.path.**relpath(path, *, start=None)**

Returns the path relative to the current blend file using the "//" prefix.

PARAMETERS:

- **path** (*str* | *bytes*) – An absolute path.
- **start** (*str* | *bytes*) – Relative to this path, when not set the current filename is used.

RETURNS:

The relative path.

RETURN TYPE:

str

bpy.path.**resolve_ncase(path)**

Resolve a case insensitive path on a case sensitive system, returning a string with the path if found else return the original path.

PARAMETERS:

path (*str*) – The path name to resolve.

RETURNS:

The resolved path.

RETURN TYPE:

str

Skip to content

Property Definitions (bpy.props)

This module defines properties to extend Blender's internal data. The result of these functions is used to assign properties to classes registered with Blender and can't be used directly.

Note

All parameters to these functions must be passed as keywords.

Assigning to Existing Classes

Custom properties can be added to any subclass of an `ID`, `Bone` and `PoseBone`.

These properties can be animated, accessed by the user interface and python like Blender's existing properties.

Warning

Access to these properties might happen in threaded context, on a per-data-block level. This has to be carefully considered when using accessors or update callbacks.

Typically, these callbacks should not affect any other data that the one owned by their data-block. When accessing external non-Blender data, thread safety mechanisms should be considered.

```
import bpy

# Assign a custom property to an existing type.
bpy.types.Material.custom_float = bpy.props.FloatProperty(name="Test Property")

# Test the property is there.
bpy.data.materials[0].custom_float = 5.0
```

Operator Example

A common use of custom properties is for python based `Operator` classes. Test this code by running it in the text editor, or by clicking the button in the 3D View-port's Tools panel. The latter will show the properties in the Redo panel and allow you to change them.

```
import bpy


class OBJECT_OT_property_example(bpy.types.Operator):
    bl_idname = "object.property_example"
    bl_label = "Property Example"
    bl_options = {'REGISTER', 'UNDO'}

    my_float: bpy.props.FloatProperty(name="Some Floating Point")
    my_bool: bpy.props.BoolProperty(name="Toggle Option")
    my_string: bpy.props.StringProperty(name="String Value")

    def execute(self, context):
        self.report(
            {'INFO'}, "F: {:.2f}  B: {:s}  S: {!r}".format(
                self.my_float, self.my_bool, self.my_string,
            )
        )
        print('My float:', self.my_float)
        print('My bool:', self.my_bool)
        print('My string:', self.my_string)
        return {'FINISHED'}


class OBJECT_PT_property_example(bpy.types.Panel):
    bl_idname = "object_PT_property_example"
    bl_label = "Property Example"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = "Tool"

    def draw(self, context):
        # You can set the property values that should be used when the user
        # presses the button in the UI.
        props = self.layout.operator('object.property_example')
        props.my_bool = True
        props.my_string = "Shouldn't that be 47?"

        # You can set properties dynamically:
        if context.object:
            props.my_float = context.object.location.x
        else:
            props.my_float = 327


bpy.utils.register_class(OBJECT_OT_property_example)
bpy.utils.register_class(OBJECT_PT_property_example)

# Demo call. Be sure to also test in the 3D Viewport.
bpy.ops.object.property_example(
    my_float=47,
    my_bool=True,
    my_string="Shouldn't that be 327?",
)
```

PropertyGroup Example

PropertyGroups can be used for collecting custom settings into one value to avoid many individual settings mixed in together.

```
import bpy


class MaterialSettings(bpy.types.PropertyGroup):
    my_int: bpy.props.IntProperty()
    my_float: bpy.props.FloatProperty()
    my_string: bpy.props.StringProperty()


bpy.utils.register_class(MaterialSettings)

bpy.types.Material.my_settings = bpy.props.PointerProperty(type=MaterialSettings)

# test the new settings work
```

```
material = bpy.data.materials[0]

material.my_settings.my_int = 5
material.my_settings.my_float = 3.0
material.my_settings.my_string = "Foo"
```

Collection Example

Custom properties can be added to any subclass of an `ID`, `Bone` and `PoseBone`.

```
import bpy


# Assign a collection.
class SceneSettingItem(bpy.types.PropertyGroup):
    name: bpy.props.StringProperty(name="Test Property", default="Unknown")
    value: bpy.props.IntProperty(name="Test Property", default=22)


bpy.utils.register_class(SceneSettingItem)

bpy.types.Scene.my_settings = bpy.props.CollectionProperty(type=SceneSettingItem)

# Assume an armature object selected.
print("Adding 2 values!")

my_item = bpy.context.scene.my_settings.add()
my_item.name = "Spam"
my_item.value = 1000

my_item = bpy.context.scene.my_settings.add()
my_item.name = "Eggs"
my_item.value = 30

for my_item in bpy.context.scene.my_settings:
    print(my_item.name, my_item.value)
```

Update Example

It can be useful to perform an action when a property is changed and can be used to update other properties or synchronize with external data.

All properties define update functions except for CollectionProperty.

Warning

Remember that these callbacks may be executed in threaded context.

Warning

If the property belongs to an Operator, the update callback’s first parameter will be an OperatorProperties instance, rather than an instance of the operator itself. This means you can’t access other internal functions of the operator, only its other properties.

```
import bpy


def update_func(self, context):
    print("my test function", self)


bpy.types.Scene.testprop = bpy.props.FloatProperty(update=update_func)

bpy.context.scene.testprop = 11.0

# >>> my test function <bpy_struct, Scene("Scene")>
```

Getter/Setter Example

Getter/setter functions can be used for boolean, int, float, string and enum properties. If these callbacks are defined the property will not be stored in the ID properties automatically. Instead, the `get` and `set` functions will be called when the property is respectively read or written from the API.

Warning

Remember that these callbacks may be executed in threaded context.

```
import bpy


# Simple property reading/writing from ID properties.
# This is what the RNA would do internally.
def get_float(self):
    return self["testprop"]


def set_float(self, value):
    self["testprop"] = value


bpy.types.Scene.test_float = bpy.props.FloatProperty(get=get_float, set=set_float)


# Read-only string property, returns the current date
def get_date(self):
    import datetime
    return str(datetime.datetime.now())


bpy.types.Scene.test_date = bpy.props.StringProperty(get=get_date)


# Boolean array. Set function stores a single boolean value, returned as the second compon
# Array getters must return a list or tuple
# Array size must match the property vector size exactly
def get_array(self):
    return (True, self["somebool"])


def set_array(self, values):
    self["somebool"] = values[0] and values[1]


bpy.types.Scene.test_array = bpy.props.BoolVectorProperty(size=2, get=get_array, set=set_a
# Enum property.
# Note: the getter/setter callback must use integer identifiers!
test_items = [
    ("RED", "Red", "", 1),
    ("GREEN", "Green", "", 2),
    ("BLUE", "Blue", "", 3),
    ("YELLOW", "Yellow", "", 4),
]


def get_enum(self):
    import random
    return random.randint(1, 4)


def set_enum(self, value):
    print("setting value", value)


bpy.types.Scene.test_enum = bpy.props.EnumProperty(items=test_items, get=get_enum, set=set


# Testing the properties:
scene = bpy.context.scene

scene.test_float = 12.34
print('test_float:', scene.test_float)

scene.test_array = (True, False)
print('test_array:', tuple(scene.test_array))

# scene.test_date = "blah"   # this would fail, property is read-only
print('test_date:', scene.test_date)

scene.test_enum = 'BLUE'
print('test_enum:', scene.test_enum)

# The above outputs:
# test_float: 12.34000015258789
# test_array: (True, False)
# test_date: 2018-03-14 11:36:53.158653
# setting value 3
# test_enum: GREEN
```

bpy.props.**BoolProperty(*, name='', description='', translation_context='*', default=False, options={'ANIMATABLE'}, override=set(), tags=set(), subtype='NONE', update=None, get=None, set=None)**

Returns a new boolean property definition.

PARAMETERS:

- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.

- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **subtype** (*str*) – Enumerator in Property Subtype Number Items.
- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.
- **get** (Callable[[`bpy.types.bpy_struct`], bool]) – Function to be called when this value is 'read', This function must take 1 value (se and return the value of the property.
- **set** (Callable[[`bpy.types.bpy_struct`, bool], None]) – Function to be called when this value is 'written', This function must take 2 values (self, value) and return None.

bpy.props.**BoolVectorProperty(*, name='', description='', translation_context='*', default=(False, False, False), options={'ANIMATABLE'}, override=set(), tags=set(), subtype='NONE', size=3, update=None, get=None, set=None)**

Returns a new vector boolean property definition.

PARAMETERS:

- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **default** (*Sequence[bool]*) – sequence of booleans the length of *size*.
- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **subtype** (*str*) – Enumerator in Property Subtype Number Array Items.
- **size** (*int* | *Sequence[int]*) – Vector dimensions in [1, 32]. An int sequence can be used to define multi-dimension arrays.
- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.
- **get** (Callable[[`bpy.types.bpy_struct`], Sequence[bool]]) – Function to be called when this value is 'read', This function must take value (self) and return the value of the property.
- **set** (Callable[[`bpy.types.bpy_struct`, tuple[bool, ...]], None]) – Function to be called when this value is 'written', This function must take 2 values (self, value) and return None.

bpy.props.**CollectionProperty(type=None, *, name='', description='', translation_context='*', options={'ANIMATABLE'}, override=set(), tags=set())**

Returns a new collection property definition.

PARAMETERS:

- **type** (type[`bpy.types.PropertyGroup`]) – A subclass of a property group.
- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Collection Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.

bpy.props.**EnumProperty(items, *, name='', description='', translation_context='*', default=None, options={'ANIMATABLE'}, override=set(), tags=set(), update=None, get=None, set=None)**

Returns a new enumerator property definition.

PARAMETERS:

- **items** (Sequence[tuple[str, str, str] | tuple[str, str, str, int] | tuple[str, str, str, int, int] | None] | Callable[[`bpy.types.bpy_struct`, `bpy.types.Context` | None], Sequence[tuple[str, str, str] | tuple[str, str, str, int] | tuple[str, str, str, int, int] | None]]) – sequence of enum items formatted: `[(identifier, name, description, icon, number), ...]`.

[illegible]

The first three elements of the tuples are mandatory.

IDENTIFIER:
: The identifier is used for Python access. An empty identifier means that the item is a separator

NAME:
: Name for the interface.

DESCRIPTION:
: Used for documentation and tooltips.

ICON:
: An icon string identifier or integer icon value (e.g. returned by `bpy.types.UILayout.icon`)

NUMBER:
: Unique value used as the identifier for this item (stored in file data). Use when the identifier may need to change. If the *ENUM_FLAG* option is used, the values are bit-masks and should be powers of two.

When an item only contains 4 items they define `(identifier, name, description, number)`.

Separators may be added using either None (nameless separator), or a regular item tuple with an empty identifier string, in which case the nan if non-empty, will be displayed in the UI above the separator line. For dynamic values a callback can be passed which returns a list in the sam format as the static list. This function must take 2 arguments `(self, context)`, **context may be None**.

> Warning
>
> There is a known bug with using a callback, Python must keep a reference to the strings returned by the callback or Blender will misbehave or even crash.

- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **default** (*str* | *int* | *set[str]*) – The default value for this enum, a string from the identifiers used in *items*, or integer matching an item number. I the *ENUM_FLAG* option is used this must be a set of such string identifiers instead. WARNING: Strings cannot be specified for dynamic enums (i.e. if a callback function is given as *items* parameter).
- **options** (*set[str]*) – Enumerator in Property Flag Enum Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.
- **get** (Callable[[`bpy.types.bpy_struct`], int]) – Function to be called when this value is 'read', This function must take 1 value (self, and return the value of the property.
- **set** (Callable[[`bpy.types.bpy_struct`, int], None]) – Function to be called when this value is 'written', This function must take 2 values (self, value) and return None.

bpy.props.**FloatProperty(*, name='', description='', translation_context='*', default=0.0, min=-3.402823e+38, max=3.402823e+38, soft_min=-3.402823e+38, soft_max=3.402823e+38, step=3, precision=2, options={'ANIMATABLE'}, override=set(), tags=set(), subtype='NONE', unit='NONE', update=None, get=None, set=None)**

Returns a new float (single precision) property definition.

PARAMETERS:

- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **min** (*float*) – Hard minimum, trying to assign a value below will silently assign this minimum instead.
- **max** (*float*) – Hard maximum, trying to assign a value above will silently assign this maximum instead.
- **soft_min** (*float*) – Soft minimum (>= *min*), user won't be able to drag the widget below this value in the UI.

- **soft_max** (*float*) – Soft maximum (<= *max*), user won't be able to drag the widget above this value in the UI.
- **step** (*int*) – Step of increment/decrement in UI, in [1, 100], defaults to 3 (WARNING: actual value is /100).
- **precision** (*int*) – Maximum number of decimal digits to display, in [0, 6]. Fraction is automatically hidden for exact integer values of fields wit unit 'NONE' or 'TIME' (frame count) and step divisible by 100.
- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **subtype** (*str*) – Enumerator in Property Subtype Number Items.
- **unit** (*str*) – Enumerator in Property Unit Items.
- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.
- **get** (Callable[[`bpy.types.bpy_struct`], float]) – Function to be called when this value is 'read', This function must take 1 value (se and return the value of the property.
- **set** (Callable[[`bpy.types.bpy_struct`, float], None]) – Function to be called when this value is 'written', This function must take 2 values (self, value) and return None.

bpy.props.**FloatVectorProperty(*, name='', description='', translation_context='*', default=(0.0, 0.0, 0.0), min=sys.float_info.min, max=sys.float_info.max, soft_min=sys.float_info.min, soft_max=sys.float_info.max, step=3, precision=2, options={'ANIMATABLE'}, override=set(), tags=set(), subtype='NONE', unit='NONE', size=3, update=None, get=None, set=None)**

Returns a new vector float property definition.

PARAMETERS:

- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **default** (*Sequence[float]*) – Sequence of floats the length of *size*.
- **min** (*float*) – Hard minimum, trying to assign a value below will silently assign this minimum instead.
- **max** (*float*) – Hard maximum, trying to assign a value above will silently assign this maximum instead.
- **soft_min** (*float*) – Soft minimum (>= *min*), user won't be able to drag the widget below this value in the UI.
- **soft_max** (*float*) – Soft maximum (<= *max*), user won't be able to drag the widget above this value in the UI.
- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **step** (*int*) – Step of increment/decrement in UI, in [1, 100], defaults to 3 (WARNING: actual value is /100).
- **precision** (*int*) – Maximum number of decimal digits to display, in [0, 6]. Fraction is automatically hidden for exact integer values of fields wit unit 'NONE' or 'TIME' (frame count) and step divisible by 100.
- **subtype** (*str*) – Enumerator in Property Subtype Number Array Items.
- **unit** (*str*) – Enumerator in Property Unit Items.
- **size** (*int* | *Sequence[int]*) – Vector dimensions in [1, 32]. An int sequence can be used to define multi-dimension arrays.
- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.
- **get** (Callable[[`bpy.types.bpy_struct`], Sequence[float]]) – Function to be called when this value is 'read', This function must take value (self) and return the value of the property.
- **set** (Callable[[`bpy.types.bpy_struct`, tuple[float, …]], None]) – Function to be called when this value is 'written', This function must take 2 values (self, value) and return None.

bpy.props.**IntProperty(*, name='', description='', translation_context='*', default=0, min=-2**31, max=2**31 - 1, soft_min=-2**31, soft_max=2**31 - 1, step=1, options={'ANIMATABLE'}, override=set(), tags=set(), subtype='NONE', update=None, get=None, set=None)**

Returns a new int property definition.

PARAMETERS:

- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **min** (*int*) – Hard minimum, trying to assign a value below will silently assign this minimum instead.
- **max** (*int*) – Hard maximum, trying to assign a value above will silently assign this maximum instead.
- **soft_min** (*int*) – Soft minimum (>= *min*), user won't be able to drag the widget below this value in the UI.
- **soft_max** (*int*) – Soft maximum (<= *max*), user won't be able to drag the widget above this value in the UI.
- **step** (*int*) – Step of increment/decrement in UI, in [1, 100], defaults to 1 (WARNING: unused currently!).
- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **subtype** (*str*) – Enumerator in Property Subtype Number Items.
- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.
- **get** (Callable[[`bpy.types.bpy_struct`], int]) – Function to be called when this value is 'read', This function must take 1 value (self and return the value of the property.
- **set** (Callable[[`bpy.types.bpy_struct`, int], None]) – Function to be called when this value is 'written', This function must take 2 values (self, value) and return None.

bpy.props.**IntVectorProperty(*, name='', description='', translation_context='*', default=(0, 0, 0), min=-2**31, max=2**31 - 1, soft_min=-2**31, soft_max=2**31 - 1, step=1, options={'ANIMATABLE'}, override=set(), tags=set(), subtype='NONE', size=3, update=Non get=None, set=None)**

Returns a new vector int property definition.

PARAMETERS:

- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **default** (*Sequence[int]*) – sequence of ints the length of *size*.
- **min** (*int*) – Hard minimum, trying to assign a value below will silently assign this minimum instead.
- **max** (*int*) – Hard maximum, trying to assign a value above will silently assign this maximum instead.
- **soft_min** (*int*) – Soft minimum (>= *min*), user won't be able to drag the widget below this value in the UI.
- **soft_max** (*int*) – Soft maximum (<= *max*), user won't be able to drag the widget above this value in the UI.
- **step** (*int*) – Step of increment/decrement in UI, in [1, 100], defaults to 1 (WARNING: unused currently!).
- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **subtype** (*str*) – Enumerator in Property Subtype Number Array Items.
- **size** (*int* | *Sequence[int]*) – Vector dimensions in [1, 32]. An int sequence can be used to define multi-dimension arrays.
- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.
- **get** (Callable[[`bpy.types.bpy_struct`], Sequence[int]]) – Function to be called when this value is 'read', This function must take 1 value (self) and return the value of the property.
- **set** (Callable[[`bpy.types.bpy_struct`, tuple[int, …]], None]) – Function to be called when this value is 'written', This function mu take 2 values (self, value) and return None.

bpy.props.**PointerProperty(type=None, *, name='', description='', translation_context='*', options={'ANIMATABLE'}, override=set(), tags=set(), poll=None, update=None)**

Returns a new pointer property definition.

PARAMETERS:

PARAMETERS:

- **type** (type[`bpy.types.PropertyGroup` | `bpy.types.ID`]) – A subclass of a property group or ID types.
- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **poll** (Callable[[`bpy.types.bpy_struct`, `bpy.types.ID`], bool]) –

  Function that determines whether an item is valid for this property. The function must take 2 values (self, object) and return a boolean.

  > Note
  >
  > The return value will be checked only when assigning an item from the UI, but it is still possible to assign an "invalid" item to the property directly.

- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.

> Note
>
> Pointer properties do not support storing references to embedded IDs (e.g. *bpy.types.Scene.collection*, *bpy.types.Material.node_tree*). These shoul
> exclusively be referenced and accessed through their owner ID (e.g. the scene or material).

bpy.props.**RemoveProperty(cls, attr)**

Removes a dynamically defined property.

PARAMETERS:

- **cls** (*type*) – The class containing the property (must be a positional argument).
- **attr** (*str*) – Property name (must be passed as a keyword).

> Note
>
> Typically this function doesn't need to be accessed directly. Instead use `del cls.attr`

bpy.props.**StringProperty(*, name='', description='', translation_context='*', default='', maxlen=0, options={'ANIMATABLE'}, override=set(), tags=set(), subtype='NONE', update=None, get=None, set=None, search=None, search_options={'SUGGESTION'})**

Returns a new string property definition.

PARAMETERS:

- **name** (*str*) – Name used in the user interface.
- **description** (*str*) – Text used for the tooltip and api documentation.
- **translation_context** (*str*) – Text used as context to disambiguate translations.
- **default** (*str*) – initializer string.
- **maxlen** (*int*) – maximum length of the string.
- **options** (*set[str]*) – Enumerator in Property Flag Items.
- **override** (*set[str]*) – Enumerator in Property Override Flag Items.
- **tags** (*set[str]*) – Enumerator of tags that are defined by parent class.
- **subtype** (*str*) – Enumerator in Property Subtype String Items.
- **update** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`], None]) – Function to be called when this value is modified, This function must take 2 values (self, context) and return None. *Warning* there are no safety checks to avoid infinite recursion.
- **get** (Callable[[`bpy.types.bpy_struct`], str]) – Function to be called when this value is 'read', This function must take 1 value (self, and return the value of the property.
- **set** (Callable[[`bpy.types.bpy_struct`, str], None]) – Function to be called when this value is 'written', This function must take 2

values (self, value) and return None.

- **search** (Callable[[`bpy.types.bpy_struct`, `bpy.types.Context`, str], Iterable[str | tuple[str, str]]]) –

  Function to be called to show candidates for this string (shown in the UI). This function must take 3 values (self, context, edit_text) and return sequence, iterator or generator where each item must be:

  - A single string (representing a candidate to display).
  - A tuple-pair of strings, where the first is a candidate and the second is additional information about the candidate.

- **search_options** (*set[str]*) –

  Set of strings in:

  - 'SORT' sorts the resulting items.
  - 'SUGGESTION' lets the user enter values not found in search candidates. **WARNING** disabling this flag causes the search callback to ru on redraw, so only disable this flag if it's not likely to cause performance issues.

Skip to content
Action(ID)

base classes — `bpy_struct`, `ID`

class bpy.types.**Action(ID)**

A collection of F-Curves for animation

curve_frame_range

The combined frame range of all F-Curves within this action

TYPE:

`mathutils.Vector` of 2 items in [-inf, inf], default (0.0, 0.0), (readonly)

fcurves

Legacy API, for backward compatibility with code that does not handle slotted actions yet. This collection contains the F-Curves for the action's first slot

TYPE:

`ActionFCurves` `bpy_prop_collection` of `FCurve`, (readonly)

frame_end

The end frame of the manually set intended playback range

TYPE:

float in [-1.04857e+06, 1.04857e+06], default 0.0

frame_range

The intended playback frame range of this action, using the manually set range if available, or the combined frame range of all F-Curves within this action if not (assigning sets the manual frame range)

TYPE:

`mathutils.Vector` of 2 items in [-inf, inf], default (0.0, 0.0)

frame_start

The start frame of the manually set intended playback range

TYPE:

float in [-1.04857e+06, 1.04857e+06], default 0.0

groups

Legacy API, for backward compatibility with code that does not handle slotted actions yet. This collection contains the F-Curve groups for th
action's first slot

TYPE:

`ActionGroups` `bpy_prop_collection` of `ActionGroup`, (readonly)

id_root

Legacy API, for backward compatibility with code that does not handle slotted actions yet. Type of data-block that the action's first slot can
used on. Do not change unless you know what you are doing

- `ACTION` Action.
- `ARMATURE` Armature.
- `BRUSH` Brush.
- `CACHEFILE` Cache File.
- `CAMERA` Camera.

- `COLLECTION` Collection.
- `CURVE` Curve.
- `CURVES` Curves.
- `FONT` Font.
- `GREASEPENCIL` Grease Pencil.
- `GREASEPENCIL_V3` Grease Pencil v3.
- `IMAGE` Image.
- `KEY` Key.
- `LATTICE` Lattice.
- `LIBRARY` Library.
- `LIGHT` Light.
- `LIGHT_PROBE` Light Probe.
- `LINESTYLE` Line Style.
- `MASK` Mask.
- `MATERIAL` Material.
- `MESH` Mesh.
- `META` Metaball.
- `MOVIECLIP` Movie Clip.
- `NODETREE` Node Tree.
- `OBJECT` Object.
- `PAINTCURVE` Paint Curve.
- `PALETTE` Palette.
- `PARTICLE` Particle.
- `POINTCLOUD` Point Cloud.
- `SCENE` Scene.
- `SCREEN` Screen.
- `SOUND` Sound.
- `SPEAKER` Speaker.
- `TEXT` Text.
- `TEXTURE` Texture.
- `VOLUME` Volume.
- `WINDOWMANAGER` Window Manager.
- `WORKSPACE` Workspace.
- `WORLD` World.
- `UNSPECIFIED` Unspecified – Not yet specified. When this slot is first assigned to a data-block, this will be set to the type of that data-block.

TYPE:

enum in [‘ACTION’, ‘ARMATURE’, ‘BRUSH’, ‘CACHEFILE’, ‘CAMERA’, ‘COLLECTION’, ‘CURVE’, ‘CURVES’, ‘FONT’, ‘GREASEPENCIL’, ‘GREASEPENCIL_V3’, ‘IMAGE’, ‘KEY’, ‘LATTICE’, ‘LIBRARY’, ‘LIGHT’, ‘LIGHT_PROBE’, ‘LINESTYLE’, ‘MASK’, ‘MATERIAL’, ‘MESH’, ‘META’, ‘MOVIECLIP’, ‘NODETREE’, ‘OBJECT’, ‘PAINTCURVE’, ‘PALETTE’, ‘PARTICLE’, ‘POINTCLOUD’, ‘SCENE’, ‘SCREEN’, ‘SOUND’, ‘SPEAKER’, ‘TEXT’, ‘TEXTURE’, ‘VOLUME’ ‘WINDOWMANAGER’, ‘WORKSPACE’, ‘WORLD’, ‘UNSPECIFIED’], default ‘UNSPECIFIED’

is_action_layered

Return whether this is a layered Action. An empty Action considered as both a ‘layered’ and a ‘layered’ Action.

TYPE:

boolean, default False, (readonly)

is_action_legacy

Return whether this is a legacy Action. Legacy Actions have no layers or slots. An empty Action considered as both a 'legacy' and a 'layered' Action. Since Blender 4.4 actions are automatically updated to layered actions, and thus this will only return True when the action is empty

TYPE:

boolean, default False, (readonly)

is_empty

False when there is any Layer, Slot, or legacy F-Curve

TYPE:

boolean, default False, (readonly)

layers

The list of layers that make up this Action

TYPE:

`ActionLayers` `bpy_prop_collection` of `ActionLayer`, (readonly)

pose_markers

Markers specific to this action, for labeling poses

TYPE:

`ActionPoseMarkers` `bpy_prop_collection` of `TimelineMarker`, (readonly)

slots

The list of slots in this Action

TYPE:

`ActionSlots` `bpy_prop_collection` of `ActionSlot`, (readonly)

use_cyclic

The action is intended to be used as a cycle looping over its manually set playback frame range (enabling this doesn't automatically make it loop)

TYPE:

boolean, default False

use_frame_range

Manually specify the intended playback frame range for the action (this range is used by some tools, but does not affect animation evaluation)

TYPE:

boolean, default False

deselect_keys()

Deselects all keys of the Action. The selection status of F-Curves is unchanged.

fcurve_ensure_for_datablock(datablock, data_path, *, index=0)

Ensure that an F-Curve exists, with the given data path and array index, for the given data-block. This action must already be assigned to the data-block. This function will also create the layer, keyframe strip, and action slot if necessary, and take care of assigning the action slot too

PARAMETERS:

- **datablock** (`ID`, (never None)) – The data-block animated by this action, for which to ensure the F-Curve exists. This action must already be assigned to the data-block
- **data_path** (*string, (never None)*) – Data Path, F-Curve data path
- **index** (*int in [0, inf], (optional)*) – Index, Array index

RETURNS:

The found or created F-Curve

RETURN TYPE:

`FCurve`

flip_with_pose(object)

Flip the action around the X axis using a pose

PARAMETERS:

object (`Object`, (never None)) – The reference armature object to use when flipping

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`
- `ID.name`
- `ID.name_full`
- `ID.id_type`
- `ID.session_uid`
- `ID.is_evaluated`
- `ID.original`
- `ID.users`
- `ID.use_fake_user`
- `ID.use_extra_user`
- `ID.is_embedded_data`
- `ID.is_missing`
- `ID.is_runtime_data`
- `ID.is_editable`
- `ID.tag`
- `ID.is_library_indirect`
- `ID.library`
- `ID.library_weak_reference`
- `ID.asset_data`
- `ID.override_library`
- `ID.preview`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.driver_add`
- `bpy_struct.driver_remove`
- `bpy_struct.get`
- `bpy_struct.id_properties_clear`
- `bpy_struct.id_properties_ensure`
- `bpy_struct.id_properties_ui`
- `bpy_struct.type_recast`
- `bpy_struct.values`
- `ID.rename`
- `ID.evaluated_get`
- `ID.copy`
- `ID.asset_mark`
- `ID.asset_clear`

- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- ID.asset_generate_preview
- ID.override_create
- ID.override_hierarchy_create
- ID.user_clear
- ID.user_remap
- ID.make_local
- ID.user_of_id
- ID.animation_data_create
- ID.animation_data_clear
- ID.update_tag
- ID.preview_ensure
- ID.bl_rna_get_subclass
- ID.bl_rna_get_subclass_py

References

- bpy.context.active_action
- bpy.context.selected_editable_actions
- bpy.context.selected_visible_actions
- ActionConstraint.action
- AnimData.action
- AnimData.action_tweak_storage
- BlendData.actions
- BlendDataActions.new
- BlendDataActions.remove
- GLTF2_filter_action.action
- NlaStrip.action
- NlaStrips.new
- Pose.apply_pose_from_action
- Pose.backup_create
- Pose.blend_pose_from_action
- SpaceDopeSheetEditor.action
- WindowManager.poselib_previous_action

Skip to content

ActionChannelbag(bpy_struct)

base class — `bpy_struct`

class bpy.types.**ActionChannelbag(bpy_struct)**

Collection of animation channels, typically associated with an action slot

fcurves

The individual F-Curves that animate the slot

TYPE:

`ActionChannelbagFCurves` `bpy_prop_collection` of `FCurve`, (readonly)

groups

Groupings of F-Curves for display purposes, in e.g. the dopesheet and graph editor

TYPE:

`ActionChannelbagGroups` `bpy_prop_collection` of `ActionGroup`, (readonly)

slot

The Slot that the Channelbag's animation data is for

TYPE:

`ActionSlot`, (readonly)

slot_handle

TYPE:

int in [-inf, inf], default 0, (readonly)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.driver_add`
- `bpy_struct.driver_remove`
- `bpy_struct.get`
- `bpy_struct.id_properties_clear`
- `bpy_struct.id_properties_ensure`
- `bpy_struct.id_properties_ui`
- `bpy_struct.is_property_hidden`
- `bpy_struct.is_property_overridable_library`
- `bpy_struct.is_property_readonly`
- `bpy_struct.is_property_set`
- `bpy_struct.items`
- `bpy_struct.keyframe_delete`
- `bpy_struct.keyframe_insert`
- `bpy_struct.keys`
- `bpy_struct.path_from_id`
- `bpy_struct.path_resolve`
- `bpy_struct.pop`
- `bpy_struct.property_overridable_library_set`
- `bpy_struct.property_unset`
- `bpy_struct.type_recast`
- `bpy_struct.values`

References

- `ActionChannelbags.new`
- `ActionChannelbags.remove`
- `ActionKeyframeStrip.channelbag`
- `ActionKeyframeStrip.channelbags`

Skip to content

ActionChannelbagFCurves(bpy_struct)

base class — `bpy_struct`

class bpy.types.**ActionChannelbagFCurves(bpy_struct)**

Collection of F-Curves for a specific action slot, on a specific strip

new(data_path, *, index=0)

Add an F-Curve to the channelbag

PARAMETERS:

- **data_path** (*string, (never None)*) – Data Path, F-Curve data path to use
- **index** (*int in [0, inf], (optional)*) – Index, Array index

RETURNS:

Newly created F-Curve

RETURN TYPE:

`FCurve`

find(data_path, *, index=0)

Find an F-Curve. Note that this function performs a linear scan of all F-Curves in the channelbag.

PARAMETERS:

- **data_path** (*string, (never None)*) – Data Path, F-Curve data path
- **index** (*int in [0, inf], (optional)*) – Index, Array index

RETURNS:

The found F-Curve, or None if it doesn't exist

RETURN TYPE:

`FCurve`

remove(fcurve)

Remove F-Curve

PARAMETERS:

fcurve (`FCurve` , (never None)) – F-Curve to remove

clear()

Remove all F-Curves from this channelbag

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.driver_add`
- `bpy_struct.driver_remove`
- `bpy_struct.get`
- `bpy_struct.id_properties_clear`
- `bpy_struct.id_properties_ensure`
- `bpy_struct.id_properties_ui`
- `bpy_struct.is_property_hidden`
- `bpy_struct.is_property_overridable_library`
- `bpy_struct.is_property_readonly`
- `bpy_struct.is_property_set`
- `bpy_struct.items`
- `bpy_struct.keyframe_delete`
- `bpy_struct.keyframe_insert`
- `bpy_struct.keys`
- `bpy_struct.path_from_id`
- `bpy_struct.path_resolve`
- `bpy_struct.pop`
- `bpy_struct.property_overridable_library_set`
- `bpy_struct.property_unset`
- `bpy_struct.type_recast`
- `bpy_struct.values`

References

- `ActionChannelbag.fcurves`

Skip to content

ActionChannelbagGroups(bpy_struct)

base class — `bpy_struct`

class bpy.types.**ActionChannelbagGroups(bpy_struct)**

Collection of f-curve groups

new(name)

Create a new action group and add it to the action

PARAMETERS:

name (*string, (never None)*) – New name for the action group

RETURNS:

Newly created action group

RETURN TYPE:

`ActionGroup`

remove(action_group)

Remove action group

PARAMETERS:

action_group (`ActionGroup` , (never None)) – Action group to remove

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.driver_add`
- `bpy_struct.driver_remove`
- `bpy_struct.items`
- `bpy_struct.keyframe_delete`
- `bpy_struct.keyframe_insert`

- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- ActionChannelbag.groups