Attribute(bpy_struct)

Attributes are used to store data that corresponds to geometry elements. Geometry elements are items in one of the geometry domains like points, curve
or faces.

An attribute has a `name` , a `type` , and is stored on a `domain` .

`name`
: The name of this attribute. Names have to be unique within the same geometry. If the name starts with a `.` , the attribute is hidden from the UI.

`type`
: The type of data that this attribute stores, e.g. a float, integer, color, etc. See Attribute Type Items.

`domain`
: The geometry domain that the attribute is stored on. See Attribute Domain Items.

Using Attributes

Attributes can be stored on geometries like `Mesh` , `Curves` , `PointCloud` , etc. These geometries have attribute groups (usually called `attributes`). Using the groups, attributes can then be accessed by their name:

```
radii = curves.attributes["radius"]
```

Creating and storing custom attributes is done using the `attributes.new` function:

```
# Add a new attribute named `my_attribute_name` of type `float` on the point domain of the
my_attribute = curves.attributes.new("my_attribute_name", 'FLOAT', 'POINT')
```

Removing attributes can be done like so:

```
attribute = drawing.attributes["some_attribute"]
drawing.attributes.remove(attribute)
```

Note

Some attributes are required and cannot be removed, like `"position"` .

Attribute values are read by accessing their `attribute.data` collection property. However, in cases where multiple values should be read at onc
it is better to use the `bpy_prop_collection.foreach_get` function and read the values into a `numpy` buffer.

```
import numpy as np

# Get the radius attribute.
radii = curves.attributes["radius"]
# Print the radius of the first point.
print(radii.data[0].value)
# Output: 0.005

# Get the total number of points.
num_points = attributes.domain_size('POINT')
# Create an empty buffer to read all the radii into.
radii_data = np.zeros(num_points, dtype=np.float32)
# Read all the radii of the curves into `radii_data` at once.
radii.data.foreach_get('value', radii_data)
# Print all the radii.
```

```
print(radii_data)
# Output: [0.1, 0.2, 0.3, 0.4, ... ]
```

Note

Some attribute types use different named properties to access their value. Instead of `value`, vectors use `vector`, and colors use `color`.

Writing to different attribute types is very similar. You can simply assign to a value directly. Again, when writing to multiple values, it is recommended to use the `bpy_prop_collection.foreach_set` function to write the values from a `numpy` buffer.

```
import numpy as np

radii = curves.attributes["radius"]
# Write a radius with a value of 0.5 to the first point.
radii.data[0].value = 0.5
print(radii.data[0].value)
# Output: 0.5

num_points = attributes.domain_size('POINT')
# Generate random radii with values between 0.001 and 0.05 using numpy.
new_radii = np.random.uniform(0.001, 0.05, num_points)
# Write the new radii to the radius attribute.
radii.data.foreach_set('value', new_radii)
```

The `bpy_prop_collection.foreach_get` / `bpy_prop_collection.foreach_set` methods require a flat array. This is sometimes not desirable, e.g. when reading/writing positions, which are 3D vectors. In these cases, it's possible to use `np.ravel` to pass the data as flat array:

```
num_points = attributes.domain_size('POINT')
positions = curves.attributes['position']
# Here, we're using a numpy array with shape (num_points, 3) so that each
# element is a 3d vector.
positions_data = np.zeros((num_points, 3), dtype=np.float32)
# The `np.ravel` function will pass the `positions_data` as a flat array
# without changing the original shape.
positions.data.foreach_get('vector', np.ravel(positions_data))
print(positions_data)
# Output: [[0.1, 0.2, 0.3], [0.4, 0.5, 0.6], ...]
```

base class — `bpy_struct`

subclasses — `BoolAttribute`, `ByteColorAttribute`, `ByteIntAttribute`, `Float2Attribute`, `Float4x4Attribute`, `FloatAttribute`, `FloatColorAttribute`, `FloatVectorAttribute`, `Int2Attribute`, `IntAttribute`, `QuaternionAttribute`, `Short2Attribute`, `StringAttribute`

class bpy.types.**Attribute(bpy_struct)**

Geometry attribute

data_type

Type of data stored in attribute

TYPE:

enum in Attribute Type Items, default 'FLOAT', (readonly)

domain

Domain of the Attribute

TYPE:

enum in Attribute Domain Items, default 'POINT', (readonly)

is_internal

The attribute is meant for internal use by Blender

TYPE:

boolean, default False, (readonly)

is_required

Whether the attribute can be removed or renamed

TYPE:

boolean, default False, (readonly)

name

Name of the Attribute

TYPE:

string, default "", (never None)

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.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.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- AttributeGroupCurves.active
- AttributeGroupCurves.new
- AttributeGroupCurves.remove
- AttributeGroupGreasePencil.active
- AttributeGroupGreasePencil.new
- AttributeGroupGreasePencil.remove
- AttributeGroupGreasePencilDrawing.active
- AttributeGroupGreasePencilDrawing.new
- AttributeGroupGreasePencilDrawing.remove
- AttributeGroupMesh.active
- AttributeGroupMesh.active_color
- AttributeGroupMesh.new
- AttributeGroupMesh.remove
- AttributeGroupPointCloud.active
- AttributeGroupPointCloud.new
- AttributeGroupPointCloud.remove
- Curves.attributes
- Curves.color_attributes
- GreasePencilDrawing.attributes
- GreasePencilDrawing.color_attributes
- GreasePencilv3.attributes
- GreasePencilv3.color_attributes
- Mesh.attributes
- Mesh.color_attributes
- PointCloud.attributes
- PointCloud.color_attributes