Python Program Reference

The following table lists the functions and properties available in the Python programmable interface:

input(id, label)	Requests a numeric input from the user. The value of the “id” parameter uniquely identifies the input, the programmer must make sure this value is only used for this input. The “label” parameter specifies the text that will appear in the UI to prompt the user for this value.
input_date(id, label)	Requests a date input from the user. The value of the “id” parameter uniquely identifies the input, the programmer must make sure this value is only used for this input. The “label” parameter specifies the text that will appear in the UI to prompt the user for this date.
input_string(id, label)	Requests a string input from the user. The value of the “id” parameter uniquely identifies the input, the programmer must make sure this value is only used for this input. The “label” parameter specifies the text that will appear in the UI to prompt the user for this string.
input_query(id, label, entity)	Requests a query input from the user. Queries are special strings that may involve conditional expressions that use the attributes in a dataset. The value of the “id” parameter uniquely identifies the input, the programmer must make sure this value is only used for this input. The “label” parameter specifies the text that will appear in the UI to prompt the user for this query. The “entity” parameter specifies which entity will be queried, this helps the system present the user a list of available attributes to query, which can be extracted from the entity’s metadata.
input_attributes(id, label, type, entity)	Requests the user to select a set of attributes. The value of the “id” parameter uniquely identifies the input, the programmer must make sure this value is only used for this input. The “label” parameter specifies the text that will appear in the UI to prompt the user for the attribute list. The “type” parameter indicates the type of input that will be used. Two values are allowed currently: • voxelfarm.type.value (1) – Used to select a single attribute • voxelfarm.type.set (2) – Used to select multiple attributes

input_entity(id, label, type)	Requests the user to pick an existing entity from the project. The value of the “id” parameter uniquely identifies the input, the programmer must make sure this value is only used for this input. The “label” parameter specifies the text that will appear in the UI to prompt the user for the entity selection. The “type” parameter contains a bit-field that signals which entity types are eligible for the input. The value for this parameter can be composed by performing a bitwise OR with the constants defined for each entity type: • voxelfarm.type.view (1 << 1) • voxelfarm.type.project (1 << 2) • voxelfarm.type.voxel_terrain (1 << 3) • voxelfarm.type.voxel_operations (1 << 4) • voxelfarm.type.block_model (1 << 5) • voxelfarm.type.point_cloud (1 << 6) • voxelfarm.type.voxel_generator (1 << 7) • voxelfarm.type.point_cloud_raw (1 << 8) • voxelfarm.type.heightmap_raw (1 << 9) • voxelfarm.type.block_model_raw (1 << 10) • voxelfarm.type.mesh_raw (1 << 11) • voxelfarm.type.mesh (1 << 12) • voxelfarm.type.ortho (1 << 13) • voxelfarm.type.program (1 << 14) • voxelfarm.type.folder (1 << 15)
set_entity_bounds_x(min, max)	Specifies the entity is contained between “min” and “max” coordinates along the X axis. This function is called only by Voxel Generator programs.
set_entity_bounds_y(min, max)	Specifies the entity is contained between “min” and “max” coordinates along the Y axis. This function is called only by Voxel Generator programs.
set_entity_bounds_z(min, max)	Specifies the entity is contained between “min” and “max” coordinates along the Z axis. This function is called only by Voxel Generator programs.
voxels	A range containing the voxels for the program to read or produce.
field	A range containing the field entries that should be produced. This is used only by Voxel Generator programs.
get_voxel_origin(voxel)	Returns a tuple that contains the (X,Y,Z) coordinates of the voxel’s origin in Project space.
get_field_origin(field_voxel)	Returns a tuple that contains the (X,Y,Z) coordinates of the field origin in Project space.
set_material(voxel, id)	Applies sub-material id to the voxel. This is used only by Voxel Generator programs. Only sub-material 1 is allowed in the current version.
set_vector(x, y, z)	Sets surface vector for the voxel. This is used only by Voxel Generator programs.

set_field(field_voxel, field_value)	Sets field value for the field voxel. This is used only by Voxel Generator programs.
set_query(entity, query)	Sets a query value to the specified entity.
set_attribute_gradient(entity, attribute, min_value, max_value)	Sets cutoff values for an attribute in the specified entity.
start_composite()	Creates a new virtual entity that will be the result of stacking multiple voxel entities as layers.
add_layer(composite, entity)	Adds the specified entity to an existing voxel composite (returned by start_composite).
get_entity_type(entity)	Returns the type of the specified entity. Possible return values are: • voxelfarm.type.view (1 << 1) • voxelfarm.type.project (1 << 2) • voxelfarm.type.voxel_terrain (1 << 3) • voxelfarm.type.voxel_operations (1 << 4) • voxelfarm.type.block_model (1 << 5) • voxelfarm.type.point_cloud (1 << 6) • voxelfarm.type.voxel_generator (1 << 7) • voxelfarm.type.point_cloud_raw (1 << 8) • voxelfarm.type.heightmap_raw (1 << 9) • voxelfarm.type.block_model_raw (1 << 10) • voxelfarm.type.mesh_raw (1 << 11) • voxelfarm.type.mesh (1 << 12) • voxelfarm.type.ortho (1 << 13) • voxelfarm.type.program (1 << 14) • voxelfarm.type.folder (1 << 15)
get_entity_name(entity)	Returns the readable name for the specified entity.
get_entity_date(entity)	Returns the capture date of the specified entity.
folder_contains(folder, entity)	Returns a Boolean indicating whether the entity is inside the specified folder. This function looks inside subfolders as well, so the entity could be nested multiple levels deep from the specified folder.
get_attributes(attributes)	Returns a range to iterate on the specified attribute set.
get_attribute_name(attributes, index)	Returns the attribute identifier for the attribute located at the specified index, from the specified attribute set.
start_sum(item, property)	Begins integration of an RDF triplet. The “item” and “property” parameters define the subject and property items of the triplet. See the “sum()” function to see how to provide a value to the triplet. Returns an integer value that identifies the new sum.
sum(sumId, value)	Adds the specified value to an existing sum. The “sumId” parameter is the identifier returned by the call to “start_sum()”.

get_volume(voxeldata, voxel)	Returns the volume of the specified voxel in the specified voxel data. The voxel data must be loaded with the “load_voxels()” function. The returned value will range from zero to the (voxel size)3
get_attribute_value(voxeldata, attribute, voxel)	Returns the value of the specified attribute for the specified voxel.
load_voxels(entity, attributes, custom_attributes)	Instructs the system to load the voxel data for the specified entity. The “attributes” parameter contains a bit mask that defines which default attributes should be loaded. This can be computed as a bitwise OR of the constants for the desired default attributes. Currently the following default attributes are supported: • voxelfarm.attribute.volume – Returns the volume of each voxel The “custom_attributes” parameter contains an array of strings where each entry corresponds to a custom attribute that may be found in the entity. If no custom attributes are required, it is possible to pass “None” in this parameter.
load_mesh(entity, attributes, custom_attributes)	Instructs the system to load the mesh data for the specified entity. In this version, the “attributes” parameter should be set to zero and the “custom_attributes” to None.
load_points(entity, attributes, custom_attributes)	Instructs the system to load the mesh data for the specified entity. In this version, the “attributes” parameter should be set to zero and the “custom_attributes” to None.
new_material()	Creates a new material instance.
render(component, material)	Instructs the system to render the specified component using the specified material. Only components of type mesh (returned by “load_mesh”) and of type points (returned by “load_points”) are supported by this version.

Revision #1
Created 17 March 2025 19:12:18 by admin
Updated 17 March 2025 19:12:54 by admin