CoPY – Python-Scripting-Tool in Cedar

CoPY is a simple scripting tool to simplify and automate the creation and editing of architectures. Core functions of architecture editing have been implemented and are explained in this document.

Scripts can be saved and loaded as Python-Files (.py).

The Python Interpreter is initially designed to remember earlier created python objects like variables, functions, or classes for usage in later executions. It can be reset manually (with the “reset”-button) or automatically (“Auto Reset” checked) after every execution. The current python objects the interpreter holds can be monitored on “Show Vars”.

Reading the tips section at the end of the document is highly recommended.

The implemented methods are listed below with the corresponding syntax explanation. As is already common in Cedar, names of the form "[GROUPNAME/S].STEPNAME.PARAMETERNAME" are used to identify steps and parameters.

Methods are held by a predefined Object “py” and are therefore invoked by py.METHOD(). Since methods are finally called in C++ datatypes must be defined.

- string[] create(string cedarClassId, int x_pos, int y_pos, string groupId = “root”, int amount = 1):
  - creates amount elements of Class cedarClassId, in Group with name groupId, at group-relative position (x_pos, y_pos) among each other. Returns the list of element identifiers for further use.
  - **Example Usage:**
    neuralFields = py.create('cedar.dynamics.NeuralField', 0, 0, "root", 10)
    //neuralFields = [“Neural Field 1”, “Neural Field 2”,…,”Neural Field 10”]

- string[] createGroupTemplate(string groupTemplate, int x_pos, int y_pos, string groupId = “root”, int amount = 1):
  - creates amount elements of template groupTemplate, in Group with name groupId, at group-relative position (x_pos, y_pos) among each other. Returns the list of element identifiers for further use.
  - **Example Usage:**
    nodes = py.createGroupTemplate('node', 0, 0, "Group", 10)
    //nodes = [“Group.node 1”, “Group.node 2”,…,”Group.node 10”]

- void connect(string sourceIdentifier, string targetIdentifier, int sourceSlot, int targetSlot):
  - creates a connection from element sourceIdentifier (Output-Slot with Index (starting from 0) or name sourceSlot) to element targetIdentifier (Input-Slot with Index (starting from 0) or name targetSlot). sourceIdentifier and targetIdentifier can also be lists to connect multiple elements at once.
  - **Example Usage:**
    py.connect(“Neural Field”,[”Convolution”, “Convolution 2”], “sigmoided activation”, 1)

- void disconnect(string sourceIdentifier, string targetIdentifier, int sourceSlot, int targetSlot):
  - deletes the connection from element sourceIdentifier (Output-Slot with Index (starting from 0) or name sourceSlot) to element targetIdentifier (Input-Slot with Index(starting from 0) or name targetSlot). sourceIdentifier and targetIdentifier can also be lists to disconnect multiple elements at once.
  - **Example Usage:**
    py.disconnect(“Neural Field”,”Group.Convolution”, “sigmoided activation”, 1)

- void setParameter(string elemIdentifier, string paramName, value):
  - changes the parameter paramName of element elemIdentifier to the new value value. Since there are different types of parameters, value doesn't have to be of a predefined type. Possible types are strings, floats, or bools. For Enum-Parameters (as "sigmoid" in Neural Fields) use the string identifier. For Vector-Parameters (as "sizes" in Neural Fields) use Python Lists (e.g. [50,50])
  - **Example Usage:**
    py.setParameter("Neural Field","lateral kernels[0].amplitude", 2)

- void addObjectList(string elemIdentifier, string paramName, string type):
  - adds a new Object of type type to the ObjectList-Parameter paramName of element elemIdentifier.
  - **Example Usage:**
    py.addObjectList("Neural Field", "lateral kernels", "cedar.aux.kernel.Box")

- void copyAllParameters(string sourceIdentifier, targetIdentifier):
  - given elements are of the same class, this method duplicates property values of element sourceIdentifier to element targetIdentifier. targetIdentifier can also be a list of elements.
  - **Example Usage:**
    py.copyAllParameters ("Neural Field", listOfNeuralFields)

Tips for easier Usage:

- Drag and drop elements from the step panel to the scripting console to automatically insert the command with the corresponding cedarClassId.
- Right click on a parameter in the properties-panel and select "use in Copy" to automatically insert the command.
- Right click on any step (or multiple selected steps) and select "use in copy" to insert the corresponding identifiers(s) as a string or a list.
- Right click anywhere in the architecture area and select "copy coordinates" to copy the group-relative coordinates of the mouse.
- Have a look at the bottom right of the architecture panel to see the current group and group-relative coordinates.