Python useful features for experiment control systems programming

Frank Laboratory of Neutron Physics, Joint Institute for Nuclear Research, 141980 Dubna, Moscow Region, Russia.
E-mails: akirilov@ñi.jinr.ru, akirilov@jinr.ru

Python programming language is popular, in particular for the programming and control systems as a basic programming language and as a language for programming user interfaces (GUI). In the programming complex Sonix+ the Python successfully used for both of these. The presentation will be devoted to some nice language features useful for programming instrument control software.

Use the Python introspection for the GUI universalization

There is a widget for manual device control in the Sonix+ GUI. This widget allows the user to select device from list and then select one of the available command for this device. It is important to emphasize that necessary information is obtained automatically as the widget code is instrument independent. In order to organize a unified device independent manual interface it is necessary to obtain list of available devices and a list of commands available for each device.

In the Sonix+ these information is concentrated in several Python files. The instrument configuration file contains full list of commands for each device. As the library of instruments commands can be appended or modified at any moment, so self.

The diagram of communication of the interpreter and the executed script is given above.

The interpretation of this code is performed by special module "Interpreter of script" (PyGui) and the interpreter is called via the interpreter interface, which is necessary in the

The interpretation of this code is performed by special module "Interpreter of script" (PyGui) and the interpreter is called via the interpreter interface, which is necessary in the

The interpretation of this code is performed by special module "Interpreter of script" (PyGui) and the interpreter is called via the interpreter interface, which is necessary in the

The interpretation of this code is performed by special module "Interpreter of script" (PyGui) and the interpreter is called via the interpreter interface, which is necessary in the

For saving normal speed execution it is offered to enter quantization of actions in a script by means of so called break points. To do this, the special procedure module cont_points was developed. Each instrument command source begins with call of

>>> gsm[2][1]
5

>>> inspect.getmembers(g, inspect.ismethod) \n
>>> import inspect

Get a list of object's methods, i.e. a list of device commands

>>> dir(g)

In the Sonix+, these information is concentrated in several Python files. The information is organized in a form of Python classes and objects. Each instrument file contains a Python class representing this instrument and a set of functions and methods.

>>> for i in dir(y):

Thus, from the configuration file, we can get a list of all announced devices, for each device - a list of its commands, for each command - prompt the name of the parameters.

>>> d = getattr(y, i)

>>> cont_points.SetCPoint (1, "Set Collimator 40 (1), wait ...")

In case of assembling user interfaces (GUI) from a set of components on there is a problem of data access arrangement, being, in other words, a structural tree. These data can be parameter values, links to methods (functions), etc. Especially as a part of components (widgets) can be used as independently, so as a part of other components. For example, the Sonix+ universal GUI-converter the widget for direct stepper motor control. This opportunity is enabled before measurement start, but must be loaded in measurement script is executing.

To implement the approach Python modules and protocols were used. The first container class class that is very convenient for the realization of specialized protocols based on the Python. - implements some useful functions to obtain information on existing objects, such as modules, classes, methods, etc.

The next class Python module has been designed. New features the command to go the next break point, abort the execution, disable/enable flags and parameters were added. The possibility of input of ordinary commands and execute it in current context was preserved.

>>> py_cmd[base:original_cemf and inspected were used. New features the command to go the next break point, abort the execution, disable/enable flags and parameters were added. The possibility of input of ordinary commands and execute it in current context was preserved.

The cont_points module is supplied with a flag of DisableConfFlag which allows to check-control script execution. It is necessary, for example, when checking a correctness of a script.