TRACE32 and Python

With the improved Python support we want to enhance two main areas:

- The Remote API in Python enables test frame developers to run complex tests via a TRACE32 backend using a native Python script *(see picture below)*.
- The new ability to edit and run Python scripts in TRACE32 gives developers direct access to features from the Python scripting environment *(see picture on the back)*.

Frequently Asked Questions about Python in TRACE32

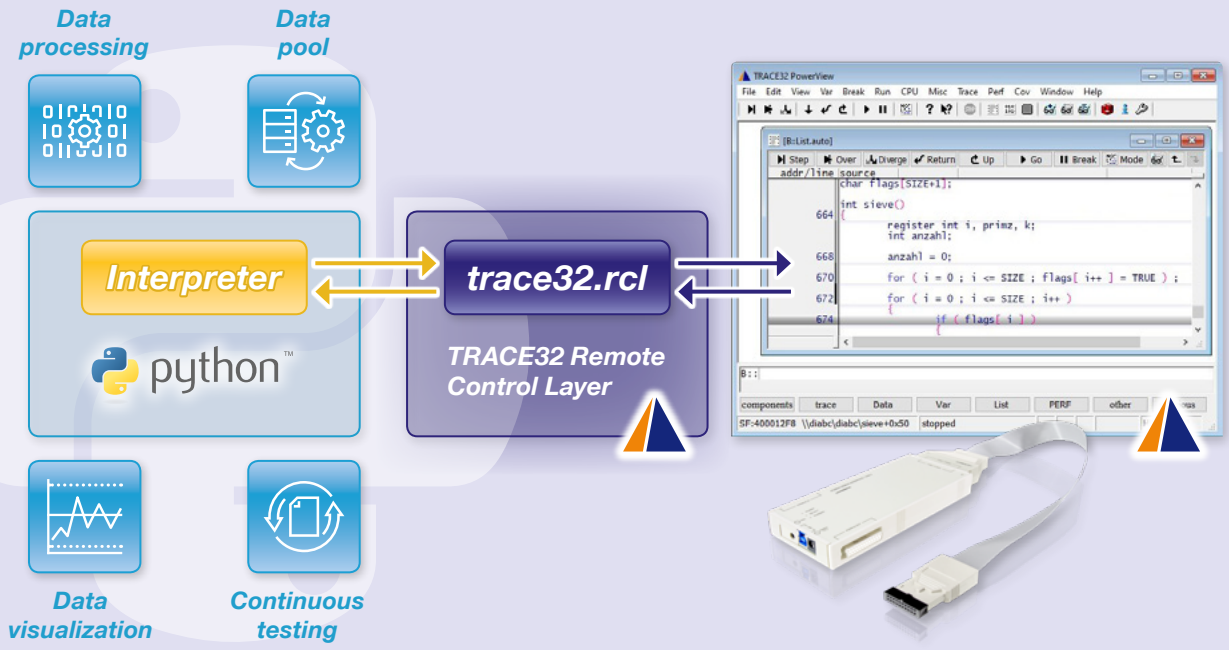
Why Python, why not my other favorite scripting languages?

If you look at the results of any survey of popularity of programming/scripting languages, you will probably find Python in top five or even top three. Other languages come and go; Python endures. Python is a living, evolving language with a colossal user base. This is the reason why we are focusing exclusively on Python. However, you can create your own support package for any language, which can interact with TRACE32 using our remote API.

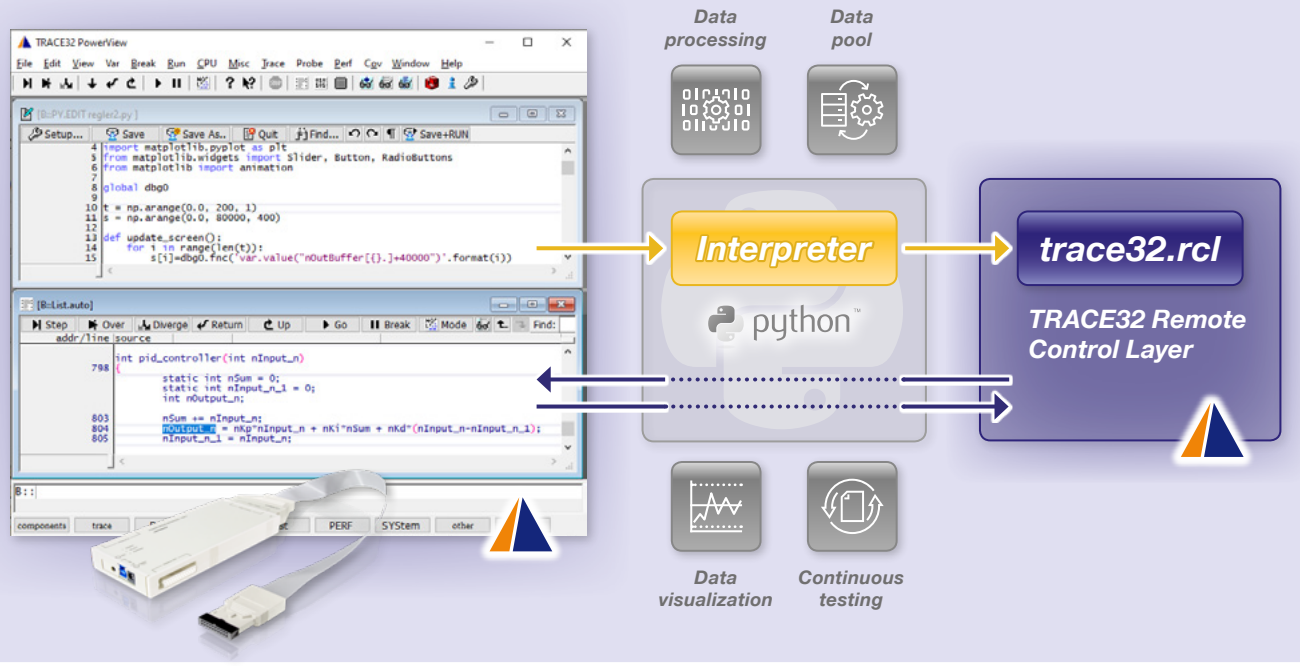
What is the benefit of using Python for the automation of repetitive tasks?

Python offers great performance for complex tasks and calculations. External libraries are available for almost anything you can imagine. As your scripts become increasingly complex, you will experience the benefits of the Python ecosystem.

Python-Controlled Scripting



TRACE32-Controlled Scripting



Which Python interpreter is used and which Python versions are supported?

We support existing installations of Python 3.5 and higher and provide a Python package, which can be installed into your Python environment. To make the setup process easier, future TRACE32 installers will include an option to install a compatible version of Python.

How can I use Python in TRACE32?

You can just type `PYthon.EDIT hello_python.py` at the TRACE32 command line and start writing your script. Three simple lines, like this, are a good starting point:

```
import lauterbach.trace32.rcl as t32
with t32.autoconnect() as dbg:
  dbg.print("Hello Python")
```

Then launch it via `PYthon.RUN hello_python.py`.

What happens when I do PYthon.RUN myscript.py?

We do not interpret Python directly in TRACE32. Instead, we fork a new process for the Python interpreter and pass myscript.py to this Python instance. In this new instance, stdin and stdout are redirected to a Python window in the TRACE32 PowerView GUI so the user does not need to continually switch between the debugger and external windows.

Can I mix Python and the standard TRACE32 script language PRACTICE?

Basically yes: because each of the two languages has its specific strengths. Since you can use TRACE32 commands directly in PRACTICE, it is easier to set-up TRACE32 itself and the target for specific test cases. It makes little sense to wrap longer sequences of TRACE32 commands in Python.

If you are setting up a test framework from scratch, we would recommend doing the flow control in Python and calling PRACTICE scripts for necessary TRACE32 and target setups. This way your Python scripts remain portable as ‘pure Python’ and use PRACTICE for its advanced control of TRACE32.

If you maintain and extend a test framework written in PRACTICE, you can now perform access to external databases and data pools, perform complex calculations, or display graphical evaluations much more easily by calling a Python script.

I am interested in this topic and would like to participate in the TRACE32 Beta Tester Program for Python.

To get a TRACE32 software update and more details please send an email to:

python-support@lauterbach.com

Lauterbach GmbH · Altlaufstr. 40 · D-85635 Höhenkirchen-Sieg. · +49 8102 9876-0 · marketing@lauterbach.com · www.lauterbach.com