Python Programming
in TopSpin

Copyright (C) 2006 by Bruker BioSpin GmbH
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means without the prior consent of the publisher.
Contents

1. Introduction ................................................................. .................................................. 3
2. Quick Start ................................................................. .................................................. 3
3. Python Functions For TopSpin .................................................. 3
1. Introduction

TopSpin includes the Jython (Java) implementation of the Python programming language. This manual is not a Jython or Python book. In order to learn the language, please refer to the following documents (there might be others available not listed here):

- The online documentation of Barry Feigenbaum

- The book *Jython Essentials* by Samuele Pedroni, Noel Rappin, O'Reilly


Here we describe how you can extend TopSpin with your own functionalities by writing Python modules or scripts. We will use the terms Python module, program, and Python script synonymously.

*Python* is a modern programming language which was embedded in TopSpin so as to provide the user with a sophisticated tool allowing him to add his own functions to TopSpin.

Python programs (modules, scripts) written for TopSpin are capable of:
- executing TopSpin commands
- opening dialog windows for user input or to print messages
- opening NMR data sets for further processing
- fetching and setting NMR parameters
- reading TopSpin NMR data for further manipulation by the Python program
- displaying arrays of data calculated in a Python program
- using the Java *swing* classes which provide a rich set of functions with virtually no limit in user interface and graphics programming
- and much more.

2. Quick Start

Enter the TopSpin command *edpy* in the command line. Result: A window is opened showing the currently available Python programs including the Bruker example programs. Please use those as a reference for your own programs. From the *edpy* window you can create a new Python program and start it up.

Editing a new or an existing Python program without using the *edpy* window: Enter the command *edpy*<program name>.

Executing an existing Python program without using the *edpy* window: Enter the command *xpy*<program name>, or just <program name>. The latter method only works if <program name> is not already the name of a TopSpin internal command or an AU program.

Making an external Python program available in TopSpin: Enter the command *edpy* and choose Import... from the File menu.

3. Python Functions For TopSpin

In your own Python programs you may use all the statements and functions described in the literature listed in chapter 1 of this manual.

In addition you can employ functions provided by TopSpin dealing with NMR commands, data, parameters, etc. They are listed in the following function table.

This table contains examples for each function. It is quite easy to execute them:

Step 1: Enter the command *help python* to open this manual.
Step 2: Enter the command *edpy pytest*. An empty text editor will be opened. Select an example in the
manual by marking the example text and copying it to the clipboard (e.g. with CTRL C, or with Edit-->Copy of the Acrobat Reader menubar). Click into the editor and paste the example text there with CTRL V. Click on the **Execute** button of the editor. That's all.

**Note:**
Many of these functions operate on the *current dataset* (e.g. fetching/storing parameters, processing data). In a Python program, the *current dataset* is defined as follows:

When the script is started, the *current dataset* is the dataset of the currently active TopSpin data window. If no such window is open, the *current dataset* is undefined. In order to define the *current dataset* inside the Python program, use the appropriate dataset functions described the following table, e.g. RE().

<table>
<thead>
<tr>
<th>Python Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output and Confirm Dialogs</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **MSG(message = "", title=None)** | A modal dialog with a *Close* button:  
  *message* = the message to be shown  
  *title* = Optional. The title of the dialog window  
  The Python program will not continue until the *Close* button is clicked. |
| **ERRMSG(message = "", title=None, details=None, modal=0)** | A modal or non-modal dialog with a *Close* button and a *Details* button:  
  *message* = the message to be shown  
  *title* = Optional. The title of the dialog window  
  *details* = Optional. A message that is displayed when clicking of the Details button of the dialog.  
  *modal* = Optional. The Python program will not continue until the *Close* button is clicked if modal = 1. Otherwise it will continue, while the dialog stays on the screen. |
| **value = CONFIRM(title=None, message="")** | A confirm dialog with an *OK* button and a *Cancel* button:  
  *title* = The title of the dialog window  
  *message* = the message to be shown  
  *value* = 1 if OK clicked, otherwise 0 |
| **value = SELECT(title=None, message="", buttons=["OK_M", "CANCEL_M"], mnemonic=None)** | A confirm dialog with an arbitrary number of buttons:  
  *title* = the title of the dialog window  
  *message* = the message to be shown  
  *buttons* = a list of button labels  
  *mnemonic* =Optional. A list of characters, shortcuts for the buttons.  
  *value* = the number of the pressed button (0, 1, ...), or negative if ESCAPE pressed, or the dialog window's close icon in the upper right corner. |

Example 1:
**ERRMSG("Test Message Non Modal")**

Example 2:
**ERRMSG("Test Message Modal", modal=1)**

Example:
```python
if CONFIRM("Delete", "Delete this file?") == 0:
    EXIT()
```

Example 1:
```python
value = SELECT("Delete", "Delete these files?", ["Delete Selected", "Delete All", "Close"])
if value == 2 or value < 0:
    EXIT()
```

Example 2:
```python
value = SELECT("Delete", "Delete these files?", ["Delete Selected", "Delete All", "Cancel", "Close"])
```
<table>
<thead>
<tr>
<th><strong>Python Function</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| All", "Close"], ['s', 'a', 'c']) | A modal or non-modal viewer for big text, e.g. as read in from a file (for short text better use MSG()):

- **title** = the title of the text window
- **header** = a header text near the top of the window
- **text** = the text
- **modal** = Option. 1=modal window (default), 0 = non-modal. |
| VIEWTEXT(title='', header='', text='', modal=1) | **Example:**
mytext=""
This is
my multi-line
text"
VIEWTEXT("MyTitle", "MyHeader", mytext) |
| SHOW_STATUS(message='') | Displays a message in TopSpin's status line. **Example:**
SHOW_STATUS("Script XYZ In Progress.") |
| **Input Dialogs** | |
| result = INPUT_DIALOG(title=None, header=None, items=None, values=None, comments=None, types=None, buttons=None, shortcuts=None, columns=30) | A general input dialog window with an arbitrary number of text input lines, and 2 buttons to press:

- **title** = the dialog title
- **header** = an additional multiline header, lines to be separated by "\n"
- **items** = list of label names for text input fields (or None)
- **values** = list of initial values for the text input fields (or None)
- **comments** = list of comments to be appended to the text input fields (or None), may have several lines to be separated by "\n"
- **types** = the list of text input field sizes (1=single line, >1 multiple line field)
- **buttons** = list of button labels. If None, an OK and Cancel button are shown.
- **shortcuts** = list of button shortcuts.
- **columns** = number of text columns (width of input fields), default = 30.
- **result** = the list of values in the text fields, if the user pressed OK, or None if the user pressed ESC, or Cancel, or the Window's close icon. |

**Example 1:**
result = INPUT_DIALOG("MyTitle", "This is an example\nNumber 1.", ["Solvent = ", "Nucleus = "], ["CDCl3", "1H"], ["", ""], ["1", "1"])
if result <> None:
    MSG(result[0] + "\n" + result[1])

**Example 2:**
result = INPUT_DIALOG("MyTitle", "This is an example\nNumber 1.", ["Solvent = ", "Nucleus = "], ["CDCl3", "1H"], ["", ""], ["1", "1"], - ["Accept","Close"], ['a','c'], 10)
if result <> None:
    MSG(result[0] + "\n" + result[1]) |
| result = DATASET_DIALOG(title=None, values=None) | A dialog window for a user to enter a TopSpin dataset with the following buttons: OK, FIND, BROWSE, CANCEL.

- **title** = the dialog title
- **values** = Optional. Initial values of the text input fields
- **result** = the list of dataset specifiers [name, expno, |

**Example 1:**
result = DATASET_DIALOG("MyTitle", CURDATA())
if result <> None:
<table>
<thead>
<tr>
<th>Python Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result = FIND_DIALOG()</td>
<td>A „find“ dialog window for a user search for TopSpin data sets</td>
</tr>
<tr>
<td>Example 1: result = FIND_DIALOG() datapaths = &quot;&quot; if result == None: EXIT() for datapath in result: datapaths += datapath + &quot;\n&quot; MSG(datapaths)</td>
<td>result = The list of full dataset paths. The list consists of those data sets selected by the user from the search result. None is returned if nothing found or the user cancelled the dialog.</td>
</tr>
<tr>
<td>CMDTHREAD</td>
<td>Not a function, but a variable of type CmdThread representing the thread in which a Python script executes. Provides access to the thread state.</td>
</tr>
<tr>
<td>result = EXEC_PYSCRIPT(pyscript, arg)</td>
<td>Executes a Python script in its own thread (i.e. in background):</td>
</tr>
<tr>
<td>Example: script = &quot;&quot; f = open(&quot;c:/foo.txt&quot;, 'w') f.write(&quot;test line\n&quot;) f.close() &quot;&quot; EXEC_PYSCRIPT(script)</td>
<td>pyscript = the script text</td>
</tr>
<tr>
<td>arg = an optional argument (any Object) result = the thread of type CmdThread</td>
<td></td>
</tr>
<tr>
<td>result = EXEC_PYFILE(pyfile, arg)</td>
<td>Executes a Python program in its own thread (i.e. in background):</td>
</tr>
<tr>
<td>Example: EXEC_PYSCRIPT(script)</td>
<td>pyfile = the absolute path of the file containing the script text, or the name of the Python program in the data base</td>
</tr>
<tr>
<td>arg = an optional argument (any Object) result = the thread of type CmdThread</td>
<td></td>
</tr>
<tr>
<td>EXIT()</td>
<td>Terminates the current Python script. Not required at the end of the Python program.</td>
</tr>
<tr>
<td>Example: if CONFIRM(&quot;Delete&quot;, &quot;Delete this file?&quot;) == 0: EXIT()</td>
<td></td>
</tr>
<tr>
<td>host = GETHOSTNAME()</td>
<td>Returns the host name of the PC where TopSpin is currently running.</td>
</tr>
<tr>
<td>Example: host = GETHOSTNAME() MSG(&quot;result=\n&quot;+host)</td>
<td></td>
</tr>
<tr>
<td>sys.argv</td>
<td>The arguments passed to a Python script.</td>
</tr>
<tr>
<td>Example: Prints the number of arguments, and the arguments when starting the script.</td>
<td>Assume you start a Python script myscript by typing xpy myscript a b c or just myscript a b c into TopSpin's command line. The 3 arguments a b c (separated by space characters) can be retrieved in the</td>
</tr>
</tbody>
</table>
### Python Function

| args = "number of args = " + str(len(sys.argv)) + "\n" | script by using `sys.argv` This is an array of Strings:
| args += "arglist:\n" | `sys.argv[0]` = the complete path of the script
| for arg in sys.argv: | `sys.argv[1]` = the first argument (if present)
| args += arg + "\n" | `sys.argv[2]` = the second argument (if present)
| MSG(args) | Etc.

---

| SLEEP(seconds) | Pauses execution of the current Python program by the specified number of seconds. Shortcut for Python’s `time.sleep()` function.

### SLEEP(2)

### ct = XCMD(cmd, wait = WAIT_TILL_DONE, arg = None)

| ct = XCMD("ft") | Executes an arbitrary TopSpin command:
| MSG("result="+str(ct.getResult())) | `cmd` = Command name
| Example 1: | `wait` = Optional: `WAIT_TILL_DONE` or `NO_WAIT_TILL_DONE`.
| Prints result as a String: `ft` return 0 if succeeded, -1 if failed. | `arg` = optional argument for the command (depends on the command whether such an argument is evaluated)
| Example 2: | `ct` = an Object of type CmdThread, allowing access to the result with `ct.getResult()`. The result type depends on the command. The result is undefined for `wait` = `NO_WAIT_TILL_DONE`.
| Terminates script if Cancel pressed in SI Dialog. | If `cmd` is a processing command (e.g. "ft"),
| if XCMD("SI").getResult() == None: | XCMD("<command>").getResult() is -1 when execution failed.
| EXIT() | If `cmd` is a parameter name (e.g. "SI"), then usually a dialog window is opened to enter the parameter.
| FT() | XCMD("<parameter>").getResult() is None when the user did not press the OK button of the dialog. Otherwise the result is a String with currently undefined value. This feature allows one to terminate the Python script when Cancel or ESC etc. pressed.

#### Please Note:
For most TopSpin processing and acquisition commands there is a Python equivalent, e.g.
FT(), APK(), EFP(), ...
XCMD should only be used when no suitable Python function is available.

---

### Defining The Current NMR Dataset

| RE(dataset = None, show = "y") | Makes the specified data set the current dataset:
| Example 1: | `dataset` = list of TopSpin dataset specifiers [name, expno, procno, directory, user]
| RE(["exam1d_13C", "2", "1", "c:/bruker/topspin", "guest"])) | `show` = Optional: `y` = dataset is displayed (default), `n` = dataset is not displayed.
<table>
<thead>
<tr>
<th>Python Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example 2:</strong></td>
<td>RE([&quot;exam1d_13C&quot;, &quot;2&quot;, &quot;1&quot;, &quot;c:/bruker/topspin&quot;, &quot;guest&quot;], &quot;n&quot;)&lt;br&gt;<strong>Example 3:</strong>&lt;br&gt;<strong>Reads 7 data sets (expno = 1...7) into a new window.</strong>&lt;br&gt;for i in range(7):&lt;br&gt;data = [&quot;exam1d_13C&quot;, str(i+1), &quot;1&quot;, &quot;c:/x y&quot;, &quot;guest&quot;]&lt;br&gt;NEWWIN()&lt;br&gt;RE(data)</td>
</tr>
<tr>
<td><strong>RE_PATH(dataset = None, show = &quot;y&quot;)</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RE(&quot;c:/bruker/topspin/data/guest/nmr/exam1d_13C/2/pdata/1&quot;)</td>
<td>Returns the current dataset of a Python script:&lt;br&gt;<strong>cmdthread</strong> = Optional: The thread in which the script is running. Default is the current script.&lt;br&gt;<strong>value</strong> = the list of dataset specifiers [name, expno, procno, directory, user]</td>
</tr>
<tr>
<td><strong>value = CURDATA(cmdthread = None)</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;curdat = CURDATA()&lt;br&gt;MSG(&quot;name=&quot;+curdat[0] + &quot;, procno=&quot;+curdat[2])</td>
<td>Returns the current dataset of a Python script:&lt;br&gt;<strong>cmdthread</strong> = Optional: The thread in which the script is running. Default is the current script.&lt;br&gt;<strong>value</strong> = the list of dataset specifiers [name, expno, procno, directory, user]</td>
</tr>
<tr>
<td><strong>RE_IEXNO(dataset = None, show = &quot;y&quot;)</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;MSG(&quot;expno-before =&quot;+CURDATA()[1])&lt;br&gt;RE_IEXPNO()&lt;br&gt;MSG(&quot;expno-after =&quot;+CURDATA()[1])</td>
<td>Increments the EXPNO of the specified dataset and makes it the current one:&lt;br&gt;<strong>dataset</strong> = Optional: list of TopSpin dataset specifiers [name, expno, procno, directory, user]&lt;br&gt;Default is the current dataset.&lt;br&gt;<strong>show</strong> = Optional: y = dataset is displayed (default), n = dataset is not displayed.&lt;br&gt;Does not create a new EXPNO if the new one doesn't exist!</td>
</tr>
<tr>
<td><strong>RE_IPROCNO(dataset = None, show = &quot;y&quot;)</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;MSG(&quot;procno-before =&quot;+CURDATA()[2])&lt;br&gt;RE_IPROCN0()&lt;br&gt;MSG(&quot;procno-after =&quot;+CURDATA()[2])</td>
<td>Increments the PROCNO of the specified dataset and makes it the current one:&lt;br&gt;<strong>dataset</strong> = Optional: list of TopSpin dataset specifiers [name, expno, procno, directory, user]&lt;br&gt;Default is the current dataset.&lt;br&gt;<strong>show</strong> = Optional: y = dataset is displayed (default), n = dataset is not displayed.&lt;br&gt;Does not create a new EXPNO if the new one doesn't exist!</td>
</tr>
<tr>
<td><strong>RSER(fidnum = None, expno = None, show = &quot;y&quot;)</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RSER(&quot;27&quot;, &quot;777&quot;)&lt;br&gt;FT()</td>
<td>Reads the specified fid number from the current data set (must be a 2D/3D/... data set with raw data), stores it under the specified EXPNO, and makes this the current data set.&lt;br&gt;<strong>fidnum</strong> = the fid number as a String (1, 2, ...)&lt;br&gt;<strong>expno</strong> = the destination expno as a String&lt;br&gt;<strong>show</strong> = Optional: y = expno is displayed (default), n = expno is not displayed.</td>
</tr>
<tr>
<td><strong>Python Function</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| **RSR(row = None, procno = None, show = "y")** | Reads the specified row from the current data set (must be 2D data set with processed data), stores it under the specified PROCNO, and makes this the current data set.  
row = the row number as a String  
procno = the destination procno as a String  
show = Optional: y = procno is displayed (default), n = expno is not displayed. |
| **Example:** | RSR("27", "999") |
| **RSC(col = None, procno = None, show = "y")** | Reads the specified column from the current data set (must be 2D data set with processed data), stores it under the specified PROCNO, and makes this the current data set.  
col = the column number as a String  
procno = the destination procno as a String  
show = Optional: y = procno is displayed (default), n = expno is not displayed. |
| **Example:** | RSC("27", "999") |

**Saving The Current NMR Dataset**

| **WR(dataset = None, override = "y")** | Writes (copies) the current dataset to the specified destination data set:  
dataset = list of TopSpin dataset specifiers [name, expno, procno, directory, user] for the destination, must be different from the source  
override = Optional: y = new destination dataset is overridden silently if existing (default), n = a confirm dialog is displayed.  
The destination does NOT become the current data set. Use RE to make it the current if needed. |
| **Example 1:** | WR(["exam1d_13C", "2", "1", "c:/", "guest"]) |
| **Example 2:** | WR(["exam1d_13C", "2", "1", "c:/", "guest"], "n") |

| **WR_PATH(dataset = None, override = "y")** | Saves (copies) the current dataset to the specified destination data set:  
dataset = the complete disk path name of a TopSpin dataset, must be different from the source  
override = Optional: y = new destination dataset is overridden silently if existing (default), n = a confirm dialog is displayed.  
The destination does NOT become the current data set. Use RE to make it the current if needed. |
| **Example 1:** | WR("c:/data/guest/nmr/exam1d_13C/2/pdata/1") |
| **Example 2:** | WR("c:/data/guest/nmr/exam1d_13C/2/pdata/1", "n") |

**Fetching And Storing NMR Parameters**

| **value = GETPAR(name, axis = 0)** | Gets the value of a TopSpin parameter from the current data set:  
name = the parameter name  
axis = Optional. 0 (acquisition axis = default) 1, 2, 3, ... (F1, F2, F3, ... axis for nD data)  
value = parameter value as a String  
Note: In order to perform calculations with value, value must be converted from String to its proper type, e.g. using the Python functions int() or float(). |

**Example 1:**

computes 2*SI and prints result:

SIstring = GETPAR("SI")  
twoTimesSI = 2*int(SIstring)  
MSG("result="+str(twoTimesSI))

**Example 2:**

computes 1/(2*D1) and prints result:

d1String = GETPAR("D 1")
<table>
<thead>
<tr>
<th>Python Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| ```
j = 1.0 / 2*float(d1String)
MSG("result="+str(j))
``` | **Example 3:**
computes 1/(2*D1) and prints result:
d1String = GETPAR("D 1")
j = 1.0 / 2*float(d1String)
MSG("result="+str(j))

**Alternate method of invoking GETPAR():**
Instead of specifying the axis as the second argument, the axis can be encoded into the parameter name:
GETPAR("1 SI") corresponds to GETPAR("SI", 1)
This is similar to the TopSpin command line syntax. It also allows one to get access to status parameters:
GETPAR("2s SI") corresponds to GETPARSTAT("SI", 2)

| value = GETPARSTAT(name, axis = 0) | Gets the value of a TopSpin status parameter (i.e. as stored in the status parameters files acqus/procs).
Usage: See GETPAR above. |
|-----------------------------------|-----------------------------------------------------------------|
| **result = GETACQU DIM()**        | Gets the dimension of the acquisition data of the current data set.
result = the dimension as an integer 1, 2, ..., or -1 in case of an error (no acquisition data present) |
| **Example:**
MSG(str(GETACQU DIM()))          |                                                                 |
| **result = GETPROCDIM()**        | Gets the dimension of the processed data of the current data set.
result = the dimension as an integer 1, 2, ... or -1 in case of an error (no processed data present) |
| **Example:**
MSG(str(GETPROCDIM()))          |                                                                 |
| **PUTPAR(name, value)**          | Sets the value of a TopSpin parameter for the current data set:
name = the parameter name, possibly including axis (for data dimension > 1) and status, similar to the TopSpin command line syntax
value = the parameter's value as a String |
| **Example 1:**
Sets the size for the acquisition direction
PUTPAR("SI", "2048")           |                                                                 |
| **Example 2:**
Sets the status parameter for the acquisition direction
PUTPAR("status SI", "2048")    |                                                                 |
| **Example 3:**
Sets the size for the F1 axis
PUTPAR("1 SI", "2048")         |                                                                 |
| **Example 4:**
Sets the status parameter for the F1 direction
PUTPAR("1s SI", "2048")        |                                                                 |
| **Example 5:**
Sets the array parameter P1
PUTPAR("P 1", "0.5")           |                                                                 |

**TopSpin Internal Windows**

10
<table>
<thead>
<tr>
<th><strong>Python Function</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>result = NEWWIN(width = -1, height = -1)</code></td>
<td>Opens an empty new internal TopSpin window.</td>
</tr>
<tr>
<td><strong>Example 1:</strong></td>
<td></td>
</tr>
<tr>
<td><code>NEWWIN()</code></td>
<td></td>
</tr>
<tr>
<td><code>RE([&quot;exam1d_13C&quot;, &quot;2&quot;, &quot;1&quot;, &quot;c:/bruker/topspin&quot;, &quot;guest&quot;])</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example 2:</strong></td>
<td></td>
</tr>
<tr>
<td><code>NEWWIN(300, 400)</code></td>
<td></td>
</tr>
<tr>
<td><code>result = GETWINID(userPanel = None)</code></td>
<td>Gets the unique ID String of an internal TopSpin window.</td>
</tr>
<tr>
<td><strong>Example 1:</strong></td>
<td></td>
</tr>
<tr>
<td>Prints the id of a new window</td>
<td></td>
</tr>
<tr>
<td><code>MSG(GETWINID(NEWWIN()))</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example 2:</strong></td>
<td></td>
</tr>
<tr>
<td>Prints the id of the default window.</td>
<td></td>
</tr>
<tr>
<td><code>MSG(GETWINID())</code></td>
<td></td>
</tr>
<tr>
<td><code>CLOSEWIN(dataset = None)</code></td>
<td>Closes the internal TopSpin window containing the specified dataset. Has no effect if no such window exists.</td>
</tr>
<tr>
<td><strong>Example 1:</strong></td>
<td></td>
</tr>
<tr>
<td><code>CLOSEWIN([&quot;exam1d_13C&quot;, &quot;2&quot;, &quot;1&quot;, &quot;c:/bruker/topspin&quot;, &quot;guest&quot;])</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example 2:</strong></td>
<td></td>
</tr>
<tr>
<td>Closes the window containing the current dataset.</td>
<td></td>
</tr>
<tr>
<td><code>CLOSEWIN(CURDATA())</code></td>
<td></td>
</tr>
<tr>
<td><code>ARRANGE(mode = &quot;$v$&quot;)</code></td>
<td>Arranges all open internal TopSpin windows.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>ARRANGE()</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example 2:</strong></td>
<td></td>
</tr>
<tr>
<td><code>ARRANGE(&quot;$h&quot;)</code></td>
<td></td>
</tr>
<tr>
<td><code>ARRANGE_WIN(winid, x, y, width, height)</code></td>
<td>Arranges the internal window with the specified ID.</td>
</tr>
<tr>
<td><strong>Example 1:</strong></td>
<td></td>
</tr>
<tr>
<td><code>id = GETWINID(NEWWIN())</code></td>
<td></td>
</tr>
<tr>
<td><code>ARRANGE_WIN(id, 20, 30, 300, 400)</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example 2:</strong></td>
<td></td>
</tr>
<tr>
<td>Loads the data set into the current window (creates a new one if none exists) and arranges the window</td>
<td></td>
</tr>
<tr>
<td><code>RE([&quot;exam1d_13C&quot;, &quot;2&quot;, &quot;1&quot;, &quot;c:/bruker/topspin&quot;, &quot;guest&quot;])</code></td>
<td></td>
</tr>
<tr>
<td><code>id = GETWINID()</code></td>
<td></td>
</tr>
<tr>
<td><code>ARRANGE_WIN(id, 20, 30, 300, 400)</code></td>
<td></td>
</tr>
<tr>
<td><code>result = ALL_WINDOWS()</code></td>
<td>Returns the all currently open internal windows (in their creation order)</td>
</tr>
<tr>
<td><strong>Python Function</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;Prints the window Ids of all open windows.&lt;br&gt;for win in ALL_WINDOWS():&lt;br&gt; MSG(GETWINID(win.getUserPanelData()))</td>
<td>result = list of windows which have type InFrame. In order to get the user panel of an InFrame (which is required e.g. by GETWINID()), use the function getUserPanelData() of InFrame.</td>
</tr>
<tr>
<td><strong>result = SELECTED_WINDOW()</strong></td>
<td>Returns the currently selected window, which is the window on which the user recently clicked.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;Prints the window Id of the selected window.&lt;br&gt;win = SELECTED_WINDOW()&lt;br&gt;MSG(GETWINID(win.getUserPanelData()))</td>
<td>result = a window of type InFrame. In order to get the user panel of an InFrame (which is required e.g. by GETWINID()), use the function getUserPanelData() of InFrame.</td>
</tr>
<tr>
<td><strong>SET_SELECTED_WIN(winid)</strong></td>
<td>Makes the window with the specified ID the current window of this script. All further actions will happen in this window.</td>
</tr>
<tr>
<td><strong>Example 1:</strong>&lt;br&gt;SET_SELECTED_WIN(&quot;2&quot;)&lt;br&gt;FT()</td>
<td>winid = the unique window ID. The script is cancelled if the id does not exist.</td>
</tr>
<tr>
<td><strong>Array Functions</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SAVE_ARRAY_AS_1R1I(reals, imags)</strong></td>
<td>Replaces the real or imaginary part of a 1D spectrum (= the contents of the 1r and/or 1i files) by new values.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;Fills 1r with a ramp, leaves 1i as it is.&lt;br&gt;curdat = [&quot;examld_13C&quot;, &quot;1&quot;, &quot;1&quot;, &quot;c:/bruker/topspin&quot;, &quot;guest&quot;]&lt;br&gt;RE(curdat)&lt;br&gt;reals = []&lt;br&gt;imags = None&lt;br&gt;for i in range(int(GETPAR(&quot;status SI&quot;))):&lt;br&gt; real.append(i)&lt;br&gt;SAVE_ARRAY_AS_1R1I(reals, imags)&lt;br&gt;RE(curdat)</td>
<td>reals = list of float numbers to replace 1r&lt;br&gt;imags = list of float numbers to replace 1i&lt;br&gt;Note: The array may have a size different from the existing file sizes. The status SI parameter is automatically updated.</td>
</tr>
<tr>
<td><strong>result = GETPROCDATA(fromppm, toppm, type = None)</strong></td>
<td>Reads a region of the current processed data set into a Python list (for 1D data only):&lt;br&gt;fromppm = region start in ppm&lt;br&gt;toppm = region end in ppm&lt;br&gt;type = Optional: dataconst.PROCDATA_IMAG to read imaginary data. Default is real.&lt;br&gt;result = list of float numbers or None if data file not found (files 1r or 1i doesn't exist). The list is always ordered from left to right, where “left” is the file start, regardless in which order the region limits are specified .&lt;br&gt;To read the entire spectrum, specify large limits, e.g. From -500 to 500 ppm.</td>
</tr>
</tbody>
</table>
| **Example 1:**<br>result = GETPROCDATA(-0.5, 0.5)<br>text = ""
for i in range(len(result)):<br> text += str(i) + " " + str(result[i]) + "\n"
VIEWTEXT(“GETPROCDATA Test”, “Read Real Data”, text) | |
<p>| <strong>Example 2:</strong>&lt;br&gt;result = GETPROCDATA(-0.5, 0.5, dataconst.PROCDATA_IMAG) | |
| <strong>result = GETPROCDATA2D(from1, to1, from2, to2, type = None)</strong> | Reads a region of the current processed data set into a Python list (for 2D data only):&lt;br&gt;from1 = region start in ppm&lt;br&gt;to1 = region end in ppm&lt;br&gt;from2 = region start in ppm&lt;br&gt;to2 = region end in ppm&lt;br&gt;type = Optional: dataconst.PROCDATA_IMAG to read imaginary data. Default is real.&lt;br&gt;result = list of float numbers or None if data file not found (files 1r or 1i doesn't exist). The list is always ordered from left to right, where “left” is the file start, regardless in which order the region limits are specified .&lt;br&gt;To read the entire spectrum, specify large limits, e.g. From -500 to 500 ppm. |</p>
<table>
<thead>
<tr>
<th>Python Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1:</td>
<td>result = GETPROCDATA(6.0, 8.0, 40.0 60.0)</td>
</tr>
<tr>
<td>Example 2:</td>
<td>Displays 2 ramps without subwindows.</td>
</tr>
<tr>
<td>Example 3:</td>
<td>Displays 2 ramps using display properties</td>
</tr>
<tr>
<td><strong>DISPLAY_DATALIST(ydata, props = None, title = &quot;&quot;, subwindows = 0)</strong></td>
<td>Displays a list of data arrays. Note that TopSpin supports printing and exporting the graphs generated with this function.</td>
</tr>
<tr>
<td><strong>DISPLAY_DATALIST_XY(ydata, xdata, props = None, title = &quot;&quot;, subwindows = 0)</strong></td>
<td>Displays a list of data arrays where y and x values are given.</td>
</tr>
</tbody>
</table>

Example 1:
result = GETPROCDATA(6.0, 8.0, 40.0 60.0)
text = ""
for i in range(len(result)):
    text += str(i) + " " + str(result[i]) + "\n"
VIEWTEXT("GETPROCDATA Test", "Read Real Data", text)

(UNTESTED, PRELIMINARY)

text = ""
for i in range(len(result)):
    text += str(i) + " " + str(result[i]) + "\n"
VIEWTEXT("GETPROCDATA Test", "Read Real Data", text)
Python Function

```
array2 = []
xvalues1 = []
for i in range(100):
    xvalues1.append(float(i*i))
for i in range(100):
    array1.append(float(i))
    array2.append(float(i*i))
DISPLAY_DATALIST_XY([array1, array2], [xvalues1, xvalues1])
```

Description

`props` = Optional. A list of display properties (axis and other), one for each array. See GET_DISPLAY_PROPS() function. If `props = None`, the x axis units are points.

title = the window title
subwindows = 1: display each array in a separate subwindow. 0: display the arrays without using subwindows.

```
GET_DISPLAY_PROPS(xStart = None, xEnd = None, xUnit = None, xLegend=None, yLegend=None, info=None, drawMode="line")
```

Example:
```
GET_DISPLAY_PROPS(2.0, 8.0, "rad","x", "y","Example", "line")
```

Returns display properties as optionally used by the DISPLAY_DATALIST functions.

xStart – left x axis limit, e.g. 0.0
xEnd – right x axis limit, e.g. 10.5. If `xEnd < xStart`, axis labelling is reversed.
xUnit – x axis unit, e.g. "ppm"
xLegend – extra text displayed below x axis
yLegend – extra text displayed left of y axis
info – extra text displayed in the upper left of the data window
drawMode – "dots" or "line". In the first case the data points are displayed in form of small circles, without connecting lines. In the second case the data points are drawn in form of a polyline.

### Acquisition, Processing And Analysis Functions

**Examples:**

```
FT(), EFP(), ZG(), APK(), XFB() etc.
```

TopSpin provides functions for most of the TopSpin commands for acquisition, processing and analysis. The name of a function is derived from the respective command by using capital letters.

In case a desired function is missing please use:

```
XCMD("TopSpin Command")
```

e.g.
```
XCMD("abs n")
```

4.