Python Programming in TopSpin

Contents

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
  http://www-128.ibm.com/developerworks/java/edu/j-dw-java-jython1-i.html

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

– The book *Jython for Java Programmers* by Robert W. Bill, New Riders.

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 Pythons *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 decribed the following table, e.g. RE().

Python Function	***Description***
Output and Confirm Dialogs	
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)
 	
Example 1:
`ERRMSG("Test Message Non Modal")`
 	
Example 2:
`ERRMSG("Test Message Modal", modal=1)`
	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="")
 	
Example:
`if CONFIRM("Delete", "Delete this file?")== 0:`
`   EXIT()`
	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"], mnemonics=None)
 	
Example 1:
`value = SELECT("Delete", "Delete these files?", ["Delete Selected", "Delete All", "Close"])`
`if value == 2 or value < 0:`
`   EXIT()`
 	
Example 2:
`value = SELECT("Delete", "Delete these files?",\`
`      ["Delete Selected", "Delete`
	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.

Python Function	*Description*
`All", "Close"], ['s', 'a', 'c'])`	
VIEWTEXT(title="", header="", text="", modal=1) *Example:* `mytext="""` `This is` `my multi-line` `example text` `"""` `VIEWTEXT("MyTitle", "MyHeader", mytext)`	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.
SHOW_STATUS(message="") *Example:* `SHOW_STATUS("Script XYZ In Progress.")`	Displays a message in TopSpin's status line.
Input Dialogs	
result = INPUT_DIALOG(title=None, header=None, items=None, values=None, comments=None, types=None, buttons=None, shortcuts=None, columns=30) *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])`	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.
result = DATASET_DIALOG(title=None, values=None) *Example 1:* `result = DATASET_DIALOG("MyTitle", CURDATA())` `if result <> 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,

Python Function	*Description*
`  MSG(result[0] + "\n" + result[1] + "\n" +  result[2] + "\n" + result[3] + "\n" + result[4])`	procno, directory, user] entered or selected by the user.

result = FIND_DIALOG()
 	
Example 1:
`result = FIND_DIALOG()`
`datapaths = ""`
`if result == None:`
`  EXIT()`
`for datapath in result:`
`  datapaths += datapath + "\n"`
`MSG(datapaths)`
	A „find“ dialog window for a user search for TopSpin data setes
	
	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.

System Functions	
CMDTHREAD	Not a function, but a variable of type CmdThread representing the thread in which a Python script executes. Provides access to the thread state.

result = EXEC_PYSCRIPT(pyscript, arg)
 	
Example:
`script = """`
`f = open("c:/foo.txt", 'w')`
`f.write("test line\\n")`
`f.close()`
`"""`
`EXEC_PYSCRIPT(script)`
	Executes a Python script in its own thread (i.e. in background):
	pyscript = the script text
	arg = an optional argument (any Object)
	result = the thread of type CmdThread

result = EXEC_PYFILE(pyfile, arg) *Example:* `EXEC_PYSCRIPT(script)`	Executes a Python program in its own thread (i.e. in background): *pyfile* = the absolute path of the file containing the script text, or the name of the Python program in the data base *arg* = an optional argument (any Object) *result* = the thread of type CmdThread

EXIT()
 	
Example:
`if CONFIRM("Delete", "Delete this file?")== 0:`
`   EXIT()`
	Terminates the current Python script. Not required at the end of the Python program.

host = GETHOSTNAME()
 	
Example:
`host = GETHOSTNAME()`
`MSG("result="+host)`
	Returns the host name of the PC where TopSpin is currently running.

sys.argv *Example:* *Prints the number of arguments, and the arguments when starting the script.*	The arguments passed to a Python script. 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

Python Function	*Description*

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

SLEEP(seconds)
 	
Example:
`SLEEP(2)`
	Pauses execution of the current Python program by the specified number of seconds. Shortcut for Python's time.sleep() function.

ct = XCMD(cmd, wait = WAIT_TILL_DONE, arg = None)
 	
Example 1:
Prints result as a String: ft return 0 if succeeded, -1 if failed.
 	
`ct = XCMD("ft")`
`MSG("result="+str(ct.getResult())`
 	
Example 2:
Terminates script if Cancle pressed in SI Dialog.
 	
`if XCMD("SI").getResult() == None:`
`  EXIT()`
`FT()`
	Executes an arbitrary TopSpin command:
	
	cmd = Command name
	wait = Optional. `WAIT_TILL_DONE` or `NO_WAIT_TILL_DONE.`
	arg = optional argument for the command (depends on the command whether such an argument is evaluated)
	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`
	
	If *cmd* is a processing command (e.g. `"ft"`), `XCMD("<command>").getResult()` is `-1` when execution failed.
	
	If *cmd* is a parameter name (e.g. `"SI"`), then usually a dialog window is opened to enter the parameter. `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")
 	
Example 1:
`RE(["exam1d_13C", "2", "1",`
`"c:/bruker/topspin", "guest"])`
	Makes the specified data set the current dataset:
	dataset = list of TopSpin dataset specifiers [name, expno, procno, directory, user]
	show = Optional: y = dataset is displayed (default), n = dataset is not displayed.

Python Function	***Description***

Example 2:
`RE(["exam1d_13C", "2", "1",`
`"c:/bruker/topspin", "guest"], "n")`
 	
Example 3:
Reads 7 data sets (expno = 1...7) into a new window.
`for i in range(7):`
`      data = ["exam1d_13C", str(i+1),`
`"1", "c:/x y", "guest"]`
`      NEWWIN()`
`      RE(data)`	

RE_PATH(dataset = None, show = "y")
 	
Example:
`RE("c:/bruker/topspin/data/guest/nmr/e`
`xam1d_13C/2/pdata/1")`
	Makes the specified data set the current dataset:
	dataset = the complete disk path name of a TopSpin dataset
	show = Optional: y = dataset is displayed (default), n = dataset is not displayed.

value = CURDATA(cmdthread = None)
 	
Example:
`curdat = CURDATA()`
`MSG("name="+curdat[0] + ", procno="`
`+curdat[2])`
	Returns the current dataset of a Python script:
	cmdthread = Optional: The thread in which the script is running. Default is the current script.
	value = the list of dataset specifiers [name, expno, procno, directory, user]

RE_IEXNO(dataset = None, show = "y")
 	
Example:
`MSG("expno-before ="+CURDATA()[1])`
`RE_IEXPNO()`
`MSG("expno-after ="+CURDATA()[1])`
	Increments the EXPNO of the specified dataset and makes it the current one:
	dataset = Optional: list of TopSpin dataset specifiers [name, expno, procno, directory, user]
	Default is the current dataset.
	show = Optional: y = dataset is displayed (default), n = dataset is not displayed.
	
	Does not create a new EXPNO if the new one doesn't exist!

RE_IPROCNO(dataset = None, show = "y")
 	
Example:
`MSG("procno-before ="+CURDATA()[2])`
`RE_IPROCNO()`
`MSG("procno-after ="+CURDATA()[2])`
	Increments the PROCNO of the specified dataset and makes it the current one:
	dataset = Optional: list of TopSpin dataset specifiers [name, expno, procno, directory, user]
	Default is the current dataset.
	show = Optional: y = dataset is displayed (default), n = dataset is not displayed.
	Does not create a new EXPNO if the new one doesn't exist!

RSER(fidnum = None, expno = None, show = "y")
 	
Example:
`RSER("27", "777")`
`FT()`
	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.
	fidnum = the fid number as a String (1, 2, ...)
	expno = the destination expno as a String
	show = Optional: y = expno is displayed (default), n = expno is not displayed.

Python Function	*Description*

RSR(row = None, procno = None, show = "y")
 	
Example:
`RSR("27", "999")`
	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.

RSC(col = None, procno = None, show = "y")
 	
Example:
`RSC("27", "999")`
	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.

Saving The Current NMR Dataset	

WR(dataset = None, override = "y")
 	
Example 1:
`WR(["exam1d_13C", "2", "1", "c:/", "guest"])`
 	
Example 2:
`WR(["exam1d_13C", "2", "1", "c:/", "guest"], "n")`
	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.

WR_PATH(dataset = None, override = "y")
 	
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")`
	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.

Fetching And Storing NMR Parameters	

value = GETPAR(name, axis = 0)
 	
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")`
	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().

Python Function	***Description***

`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))`
 	
Example 4:
*computes 2*SI (from F1 direction of nD data) and prints result:*
 	
`SIstring = GETPAR("SI", 1)`
`twoTimesSI = 2*int(SIstring)`
`MSG("result="+str(twoTimesSI))`
	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 = GETACQUDIM() *Example:* `MSG(str(GETACQUDIM()))`	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)
result = GETPROCDIM() *Example:* `MSG(str(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)

PUTPAR(name, value)
 	
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")`
	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

TopSpin Internal Windows	

Python Function	*Description*

result = NEWWIN(width = -1, height = -1)
 	
Example 1:
`NEWWIN()`
`RE(["exam1d_13C", "2", "1", "c:/bruker/topspin", "guest"])`
 	
Example 2:
`NEWWIN(300, 400)`
	Opens an empty new internal TopSpin window.
	
	width = Optiona: window width (pixels)
	height = Optiona: window height (pixels)
	result = a JPanel object, a container for e.g. NMR data or other components.
	
	RE() type functions will load their datasets into the last opened window, regardless what other windows are open.

result = GETWINID(userPanel = None)
 	
Example 1:
Prints the id of a new window
`MSG(GETWINID(NEWWIN()))`
 	
Example 2:
Prints the id of the default window.
`MSG(GETWINID())`
	Gets the unique ID String of an internal TopSpin window.
	
	userPanel = Optional: The user panel of the window, as e.g. returned by NEWWIN(). If not specified, the id of the window is returned which was the current one at the time the script was started (default window).
	result = the unique ID String, or None if no window existing for this script.

CLOSEWIN(dataset = None)
 	
Example 1:
`CLOSEWIN(["exam1d_13C", "2", "1", "c:/bruker/topspin", "guest"])`
 	
Example 2:
Closes the window containg the current dataset.
`CLOSEWIN(CURDATA())`
	Closes the internal TopSpin window containing the specified dataset. Has no effect if no such window exists.
	
	dataset = a list of dataset specifiers [name, expno, procno, dir, user]
	
	RE() type functions will load their datasets into the last opened window.

ARRANGE(mode = "$v")
 	
Example:
`ARRANGE()`
 	
Example 2:
`ARRANGE("$h")`
	Arranges all open internal TopSpin windows.
	
	mode = $v (default, arrange vertically = in stack), $h (arrange horizontally = in side-by-side), $m ((arrange as a grid)

ARRANGE_WIN(winid, x, y, width, height)
 	
Example 1:
`id = GETWINID(NEWWIN())`
`ARRANGE_WIN(id, 20, 30, 300, 400)`
 	
Example 2:
Loads the data set into the current window (creates a new one if none exists) and arranges the window
`RE(["exam1d_13C", "2", "1", "c:/bruker/topspin", "guest"])`
`id = GETWINID()`
`ARRANGE_WIN(id, 20, 30, 300, 400)`
	Arranges the nternal window with the specified ID.
	
	winid = window identifier as returned by GETWINID()
	x = x coordinate
	y = y coordinate
	width = window width (pixels)
	height = window height (pixels)
	
	x/y/width/height must be specified as integers. The point (0, 0) is the upper left corner of a window, the point (*width-1, height-1)* is the lower right corner.

result = ALL_WINDOWS()	Returns the all currently open internal windows (in their creation order)

Python Function	*Description*

Example:
Prints the window Ids of all open windows.
`for win in ALL_WINDOWS():`
`  MSG(GETWINID(win.getUserPanelData())`
`)`
	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.

result = SELECTED_WINDOW()
 	
Example:
Prints the window Id of the selected window.
`win = SELECTED_WINDOW()`
`MSG(GETWINID(win.getUserPanelData()))`
	Returns the currently selected window, which is the window on which the user recently clicked.
	
	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.

SET_SELECTED_WIN(winid)
 	
Example 1:
`SET_SELECTED_WIN("2")`
`FT()`
	Makes the window with the specified ID the current window of this script. All further actions will happen in this window.
	
	winid = the unique window ID. The script is cancelled if the id does not exist.

Array Functions	

SAVE_ARRAY_AS_1R1I(reals, imags)
 	
Example:
Fills 1r with a ramp, leaves 1i as it is.
`curdat = ["exam1d_13C", "1", "1",`
`"c:/bruker/topspin", "guest"]`
`RE(curdat)`
`reals = []`
`imags = None`
`for i in range(int(GETPAR("status`
`SI"))):`
`      reals.append(i)`
`SAVE_ARRAY_AS_1R1I(reals, imags)`
`RE(curdat)`
	Replaces the real or imaginary part of a 1D spectrum (= the contents of the 1r and/or 1i files) by new values.
	
	reals = list of float numbers to replace 1r
	imags = list of float numbers to replace 1i
	
	Note:
	The array may have a size different from the existing file sizes. The status SI parameter is automatically updated.

result = GETPROCDATA(fromppm, toppm, type = None)
 	
Example 1:
`result = GETPROCDATA(-0.5, 0.5)`
`text = ""`
`for i in range(len(result)):`
`   text += str(i) + " " +`
`       str(result[i])  + "\n"`
`VIEWTEXT("GETPROCDATA Test", "Read`
`Real Data", text)`
 	
Example 2:
`result = GETPROCDATA(-0.5, 0.5,`
`dataconst.PROCDATA_IMAG)`
	Reads a region of the current processed data set into a Python list (for 1D data only):
	fromppm = region start in ppm
	toppm = region end in ppm
	type = Optional: `dataconst.PROCDATA_IMAG` to read imaginary data. Default is real.
	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 .
	
	To read the entire spectrum, specifiy large limits, e.g. From -500 to 500 ppm.

result = GETPROCDATA2D(from1, to1, from2, to2, type = None)	Reads a region of the current processed data set into a Python list (for 2D data only):

Python Function	*Description*

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)`
	from1, from2 = region start in ppm
	to1, to2 = region end in ppm
	type = Optional: `dataconst.PROCDATA_IMAG` to read imaginary data. Default is real.
	result = list of list of float numbers or None if data file not found (files 2rr or 2ii doesn't exist). A list is always ordered from left to right, where "left" is the file start, regardless in which order the region limits are specified .
	
	To read the entire spectrum, specifiy large limits, e.g. From -500 to 500 ppm.

DISPLAY_DATALIST(ydata, props = None, title = "", subwindows = 0)
 	
Example 1:
Displays 2 ramps without subwindows.
`array1 = []`
`array2 = []`
`for i in range(100):`
`  array1.append(float(i))`
`  array2.append(100-float(i))`
`DISPLAY_DATALIST([array1, array2])`
Example 2:
Displays 2 ramps with a window title and subwindows.
`array1 = []`
`array2 = []`
`for i in range(100):`
`  array1.append(float(i))`
`  array2.append(100-float(i))`
`DISPLAY_DATALIST([array1, array2], None, "Example", 1)`
 	
Example 3:
Displays 2 ramps using display properties
`array1 = []`
`array2 = []`
`for i in range(100):`
`  array1.append(float(i))`
`  array2.append(float(2*i))`
 	
`props1 = GET_DISPLAY_PROPS(8.0, 1.0, "rad","x", "y", "Example", "line")`
`props2 = GET_DISPLAY_PROPS(8.0, 1.0, "rad","x", "y", "Example", "dots")`
`DISPLAY_DATALIST([array1, array2], [props1, props2])`
	Displays a list of data arrays. Note that TopSpin supports printing and exporting the graphs generated with this function.
	
	ydata = A list of arrays. Each array must contain float numbers.
	props = Optional. A list of display properties, one for each array. See GET_DISPLAY_PROPS() function.
	title = the window title
	subwindows = 1: display each array in a separate subwindow. 0: display the arrays without using subwindows.

DISPLAY_DATALIST_XY(ydata, xdata, props = None, title = "", subwindows = 0)
 	
Example 1:
Displays 2 ramps without subwindows.
`array1 = []`
	Displays a list of data arrays where y and x values are given.
	
	ydata = A list of arrays. Each array must contain float numbers.
	xdata = The x axis values for *ydata*.

Python Function	***Description***

`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])`
	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.