NASA uses Python...

... so does Rackspace, Industrial Light and Magic, AstraZeneca, Honeywell, and many others.

(from www.python.org)

Python Tutorial: Using Python In TopSpin

B. Guigas, June 2007

Bruker BioSpin

Why Another Scripting Language in TopSpin?

- **AU programs** are C programs: Require compilation/linking after each change → long turn-around time during development
- **AU programs** do not provide access to the Graphical user interface
- **Bruker Macros** are just command sequences, no possibility for looping, branching, computing, ….

In contrast, Python allows for:

- *Rapid development:* Type something in, try it out immediately
- *True scripting:* No declaration of variables such as char*, int [] etc. required → shorter source code, also easier to read/write
- *Make it graphical:* Display dialog windows or graphics of any kind

First Examples

These examples do NOT contain C or Python statements, but just TopSpin functions

AU-Prog

```
GETCURDATA
EM
FT  // fourier
APK
QUIT
```

Macro

```
em
ft  # fourier
apk
```

Python-Prog

```
EM()
FT() # fourier
APK()
```

How To Write A Program

TopSpin commands

edau ↓ AU-Prog

edmac ↓ Macro

edpy ↓ Python-Prog

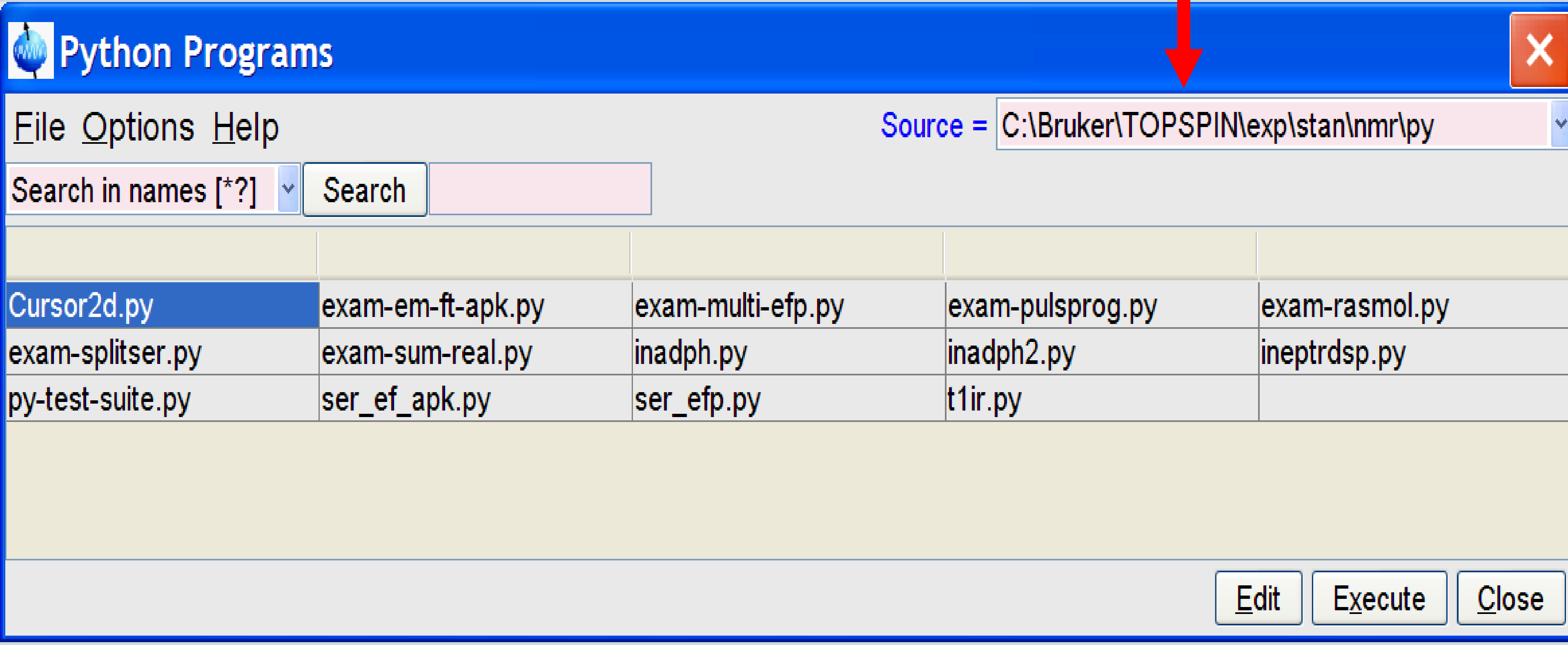
```
GETCURDATA
EM
FT  // fourier
APK
QUIT
```

```
em
ft  # fourier
apk
```

```
EM()
FT() # fourier
APK()
```

Edpy: Opens Python Program Browser

Directory where py-programs are stored

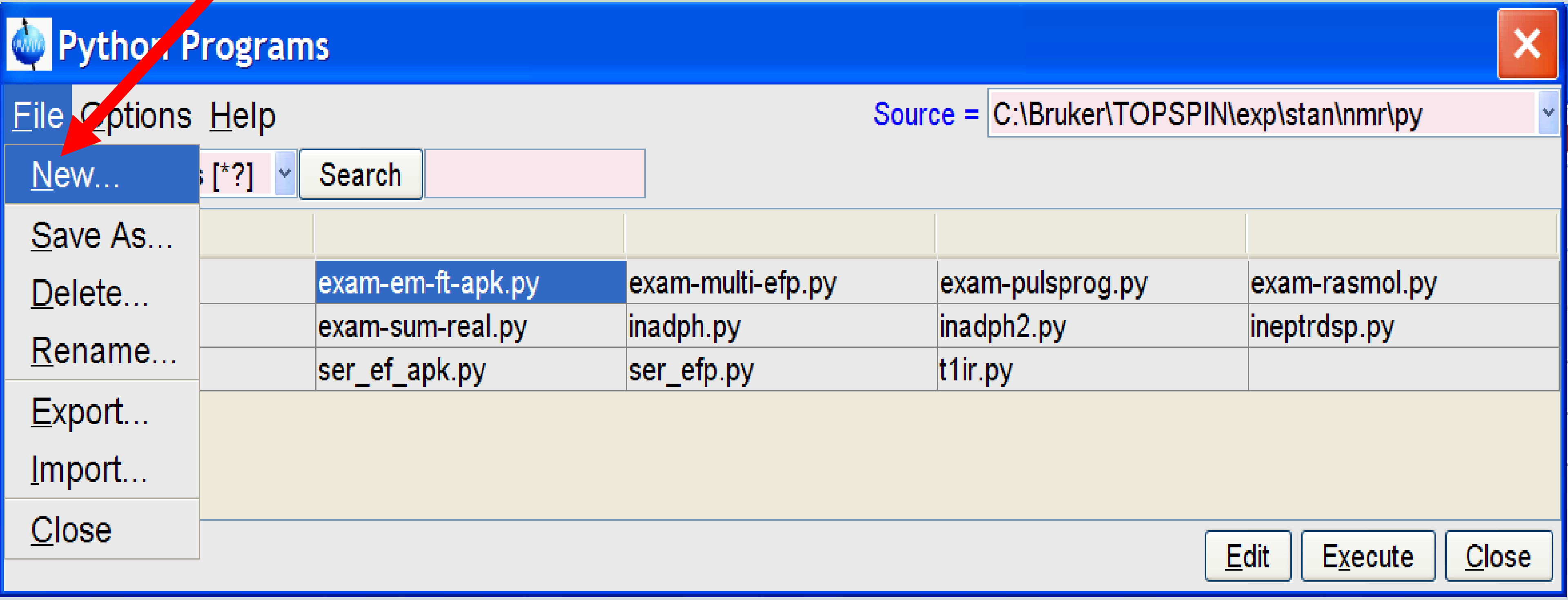


Source = C:\Bruker\TOPSPIN\exp\stan\nmr\py

Cursor2d.py	exam-em-ft-apk.py	exam-multi-efp.py	exam-pulsprog.py	exam-rasmol.py
exam-splitser.py	exam-sum-real.py	inadph.py	inadph2.py	ineptrdsp.py
py-test-suite.py	ser_ef_apk.py	ser_efp.py	t1ir.py	

Creating A New Python Program

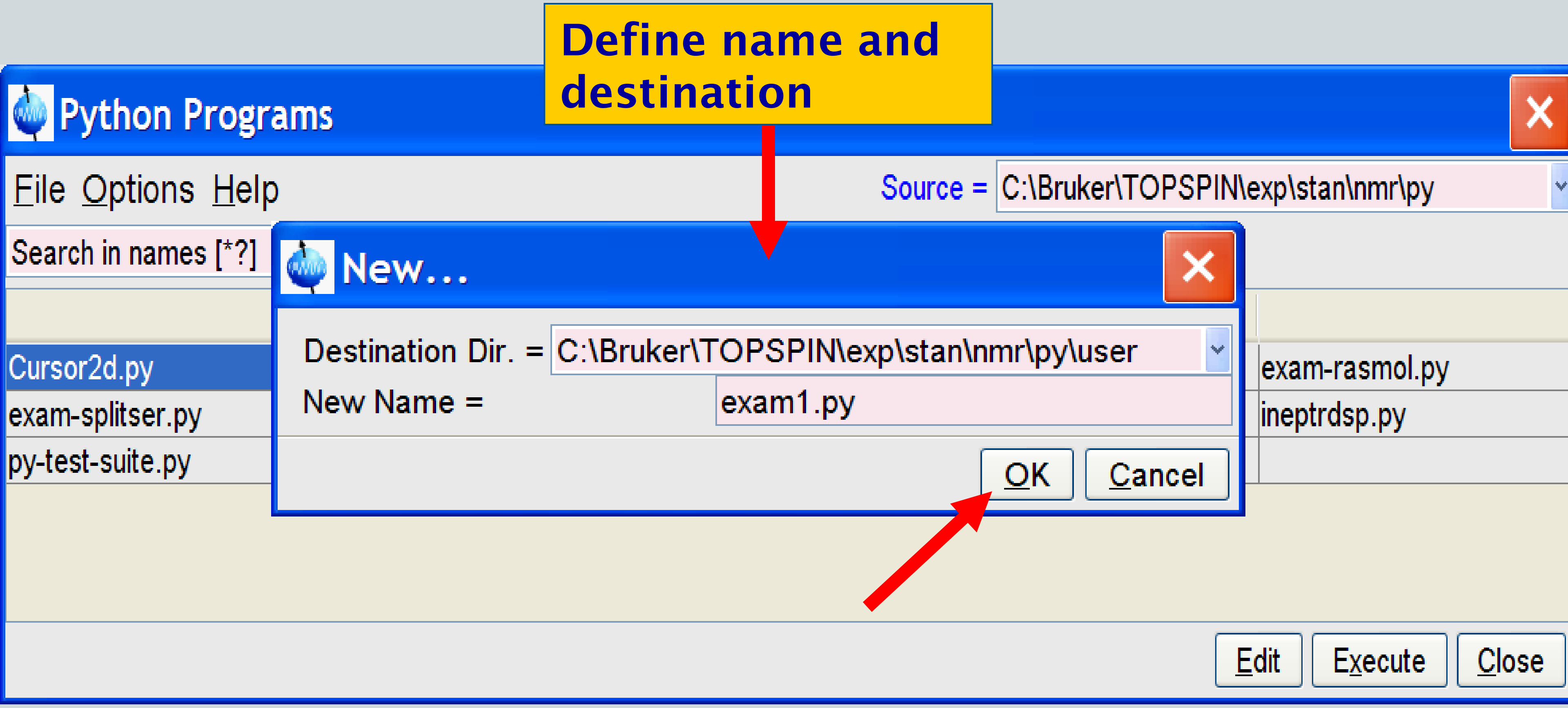
Edpy → File → New…



exam-em-ft-apk.py	exam-multi-efp.py	exam-pulsprog.py	exam-rasmol.py
exam-sum-real.py	inadph.py	inadph2.py	ineptrdsp.py
ser_ef_apk.py	ser_efp.py	t1ir.py	

Creating A New Python Program

Edpy → File → New…

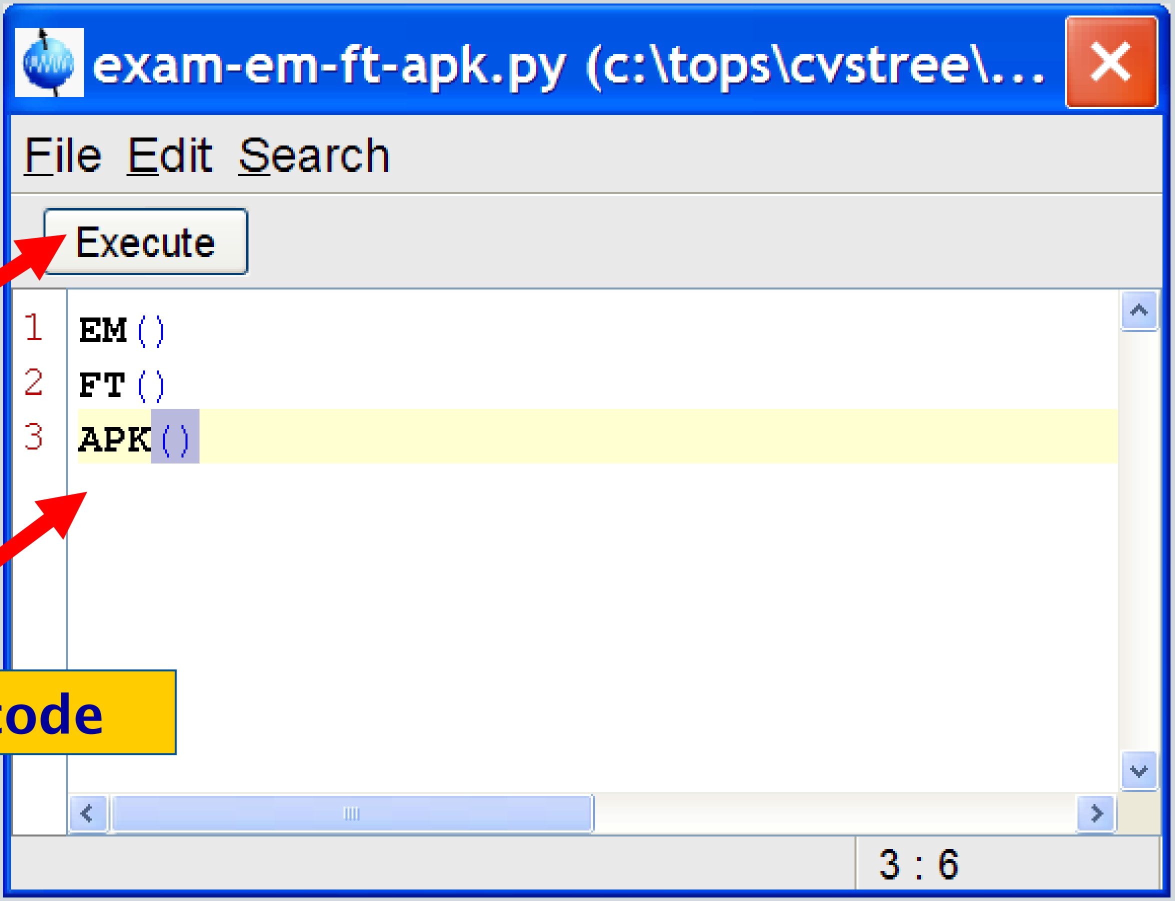


Source = C:\Bruker\TOPSPIN\exp\stan\nmr\py

Destination Dir. = C:\Bruker\TOPSPIN\exp\stan\nmr\py\user

New Name = exam1.py

Edpy → File → New… → ok
Opens Text Editor



```
EM()
FT()
APK()
```

Execute program

Enter program code

Commands Not Using The Browser

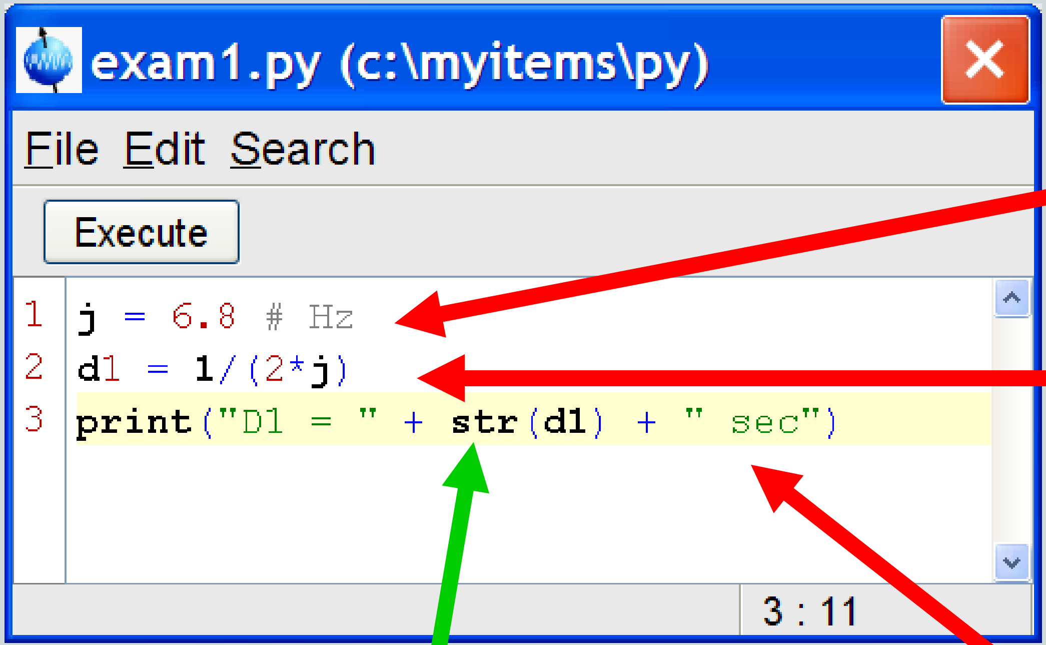
```
edpy  myprog
```

Opens text editor for „myprog.py“

```
xpy  myprog
    or
xpy myprog.py
    or
myprog.py
```

Execute „myprog.py“

A First True Python Example



```
j = 6.8 # Hz
d1 = 1/(2*j)
print("D1 = " + str(d1) + " sec")
```

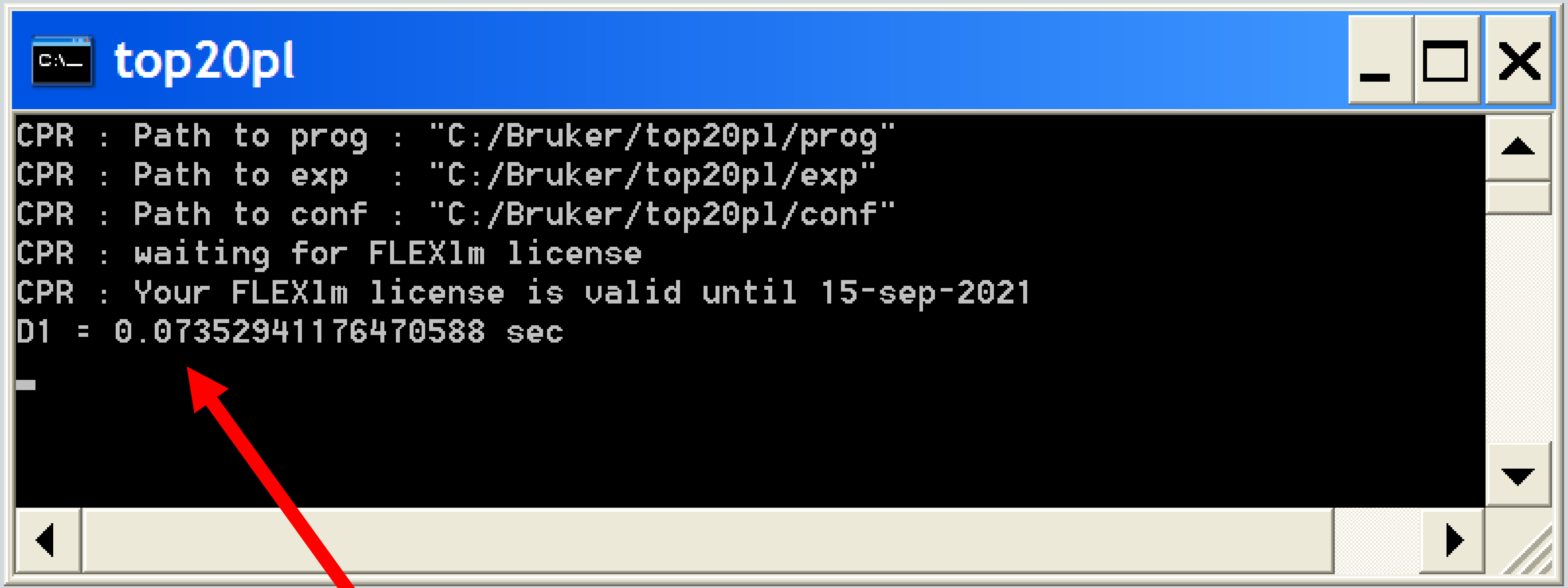
Assign a value to a variable j, add a comment (#)

Perform a calculation, assign result to a variable d1

Print the result on the console

Numbers must be converted to a String before printing using the str() function!

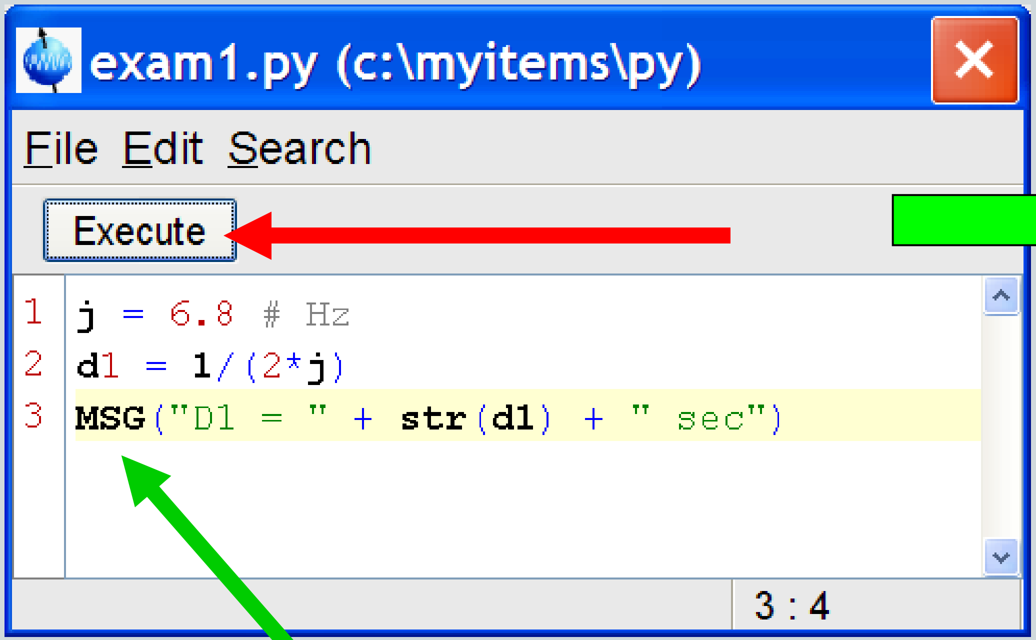
TopSpin Console With Result



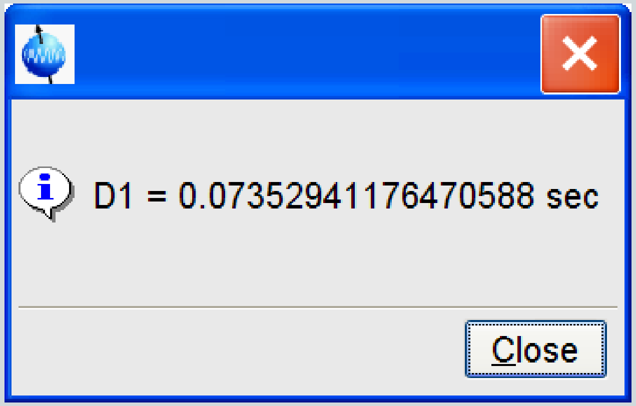
```
CPR : Path to prog : "C:/Bruker/top20pl/prog"
CPR : Path to exp  : "C:/Bruker/top20pl/exp"
CPR : Path to conf : "C:/Bruker/top20pl/conf"
CPR : waiting for FLEXlm license
CPR : Your FLEXlm license is valid until 15-sep-2021
D1 = 0.0735294117647O588 sec
```

Result of „exam1.py“

Better Show Result In A Window



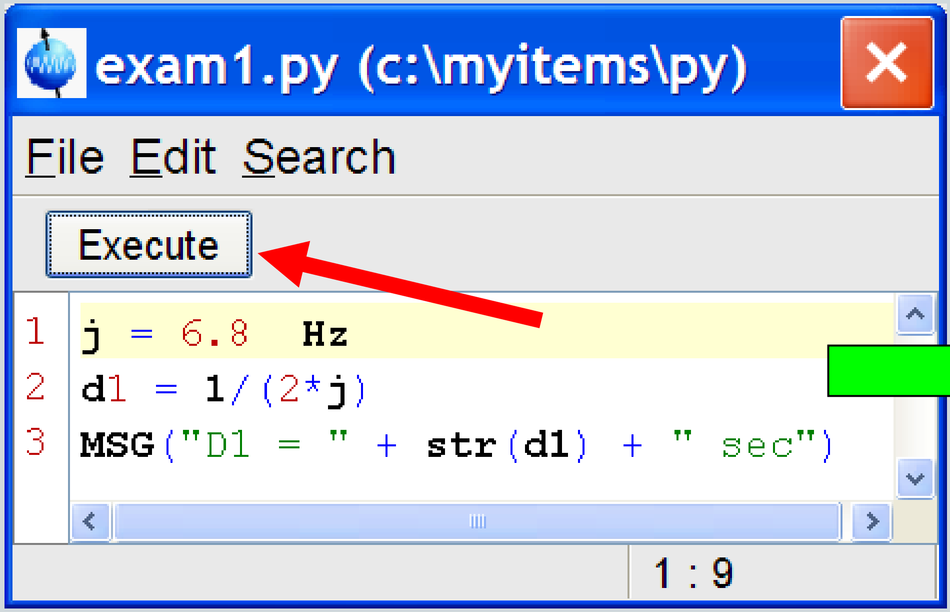
```
j = 6.8 # Hz
d1 = 1/(2*j)
MSG("D1 = " + str(d1) + " sec")
```



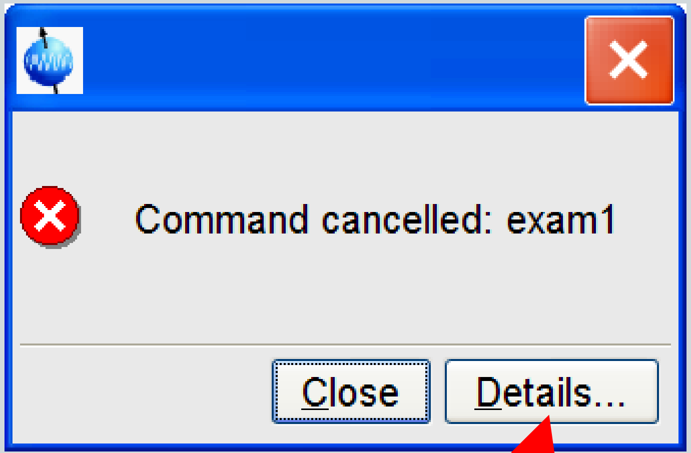
D1 = 0.07352941176470588 sec

TopSpin function MSG()

Don’t Give Up On Errors… Case 1

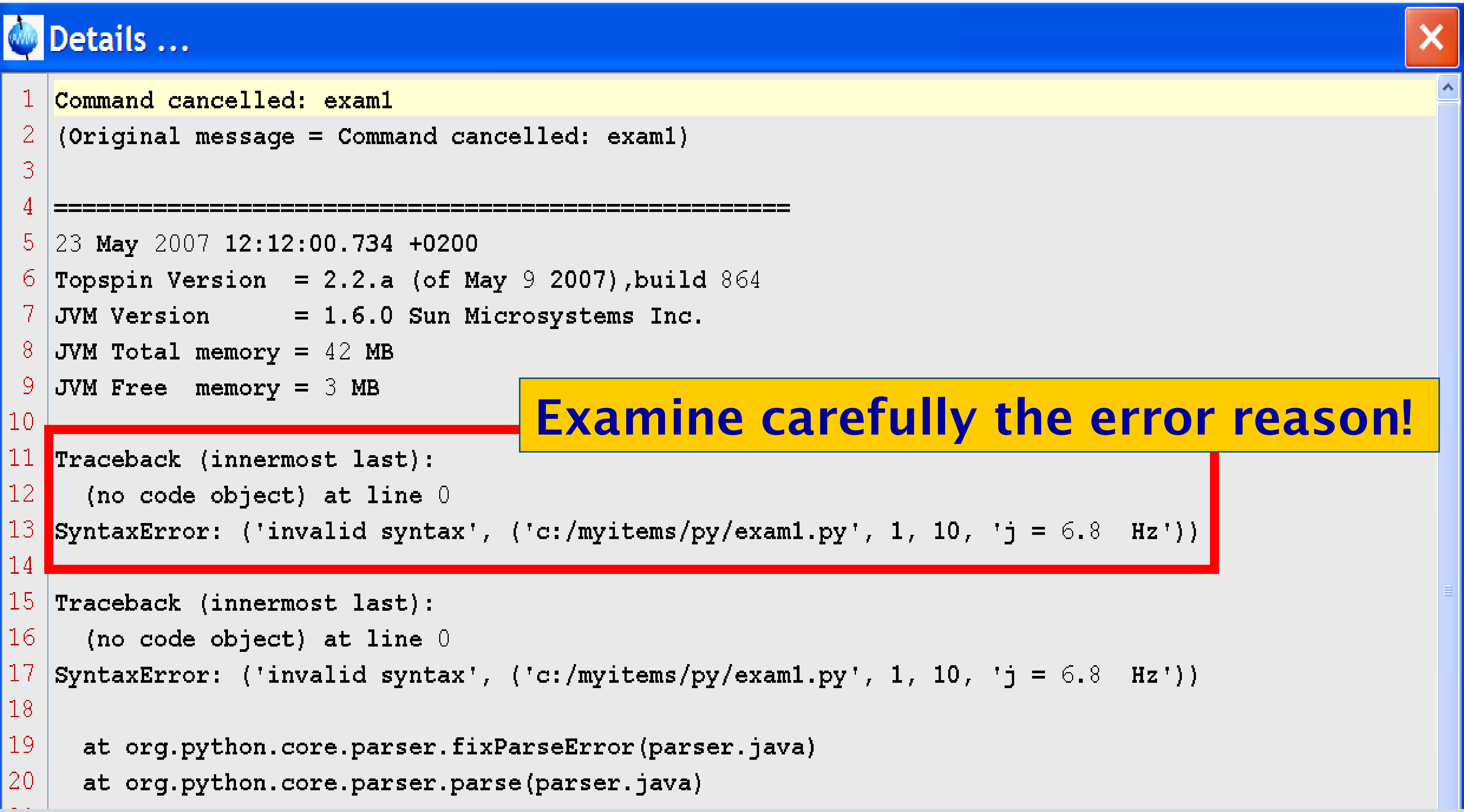


```
j = 6.8  Hz
d1 = 1/(2*j)
MSG("D1 = " + str(d1) + " sec")
```


Command cancelled: exam1

Typical error message when developing Python programs!
Click on *Details* to learn more…

Details-Window When Program Cancelled



Details ...

```
Command cancelled: exam1
(Original message = Command cancelled: exam1)

====================================================
23 May 2007 12:12:00.734 +0200
Topspin Version  = 2.2.a (of May 9 2007),build 864
JVM Version      = 1.6.0 Sun Microsystems Inc.
JVM Total memory = 42 MB
JVM Free  memory = 3 MB

Traceback (innermost last):
  (no code object) at line 0
SyntaxError: ('invalid syntax', ('c:/myitems/py/exam1.py', 1, 10, 'j = 6.8  Hz'))

Traceback (innermost last):
  (no code object) at line 0
SyntaxError: ('invalid syntax', ('c:/myitems/py/exam1.py', 1, 10, 'j = 6.8  Hz'))

  at org.python.core.parser.fixParseError(parser.java)
  at org.python.core.parser.parse(parser.java)
```

Examine carefully the error reason!

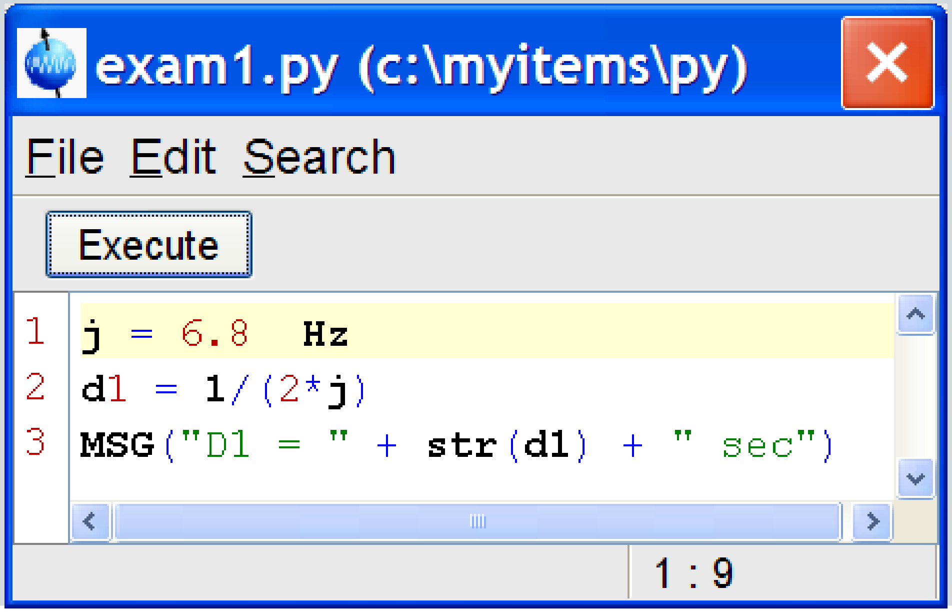
Details-Window, Error Reason

```
Traceback (innermost last):
  (no code object) at line 0
SyntaxError: ('invalid syntax', ('c:/myitems/py/exam1.py', 1, 10, 'j = 6.8  Hz'))
```

error reason

file

line and column number



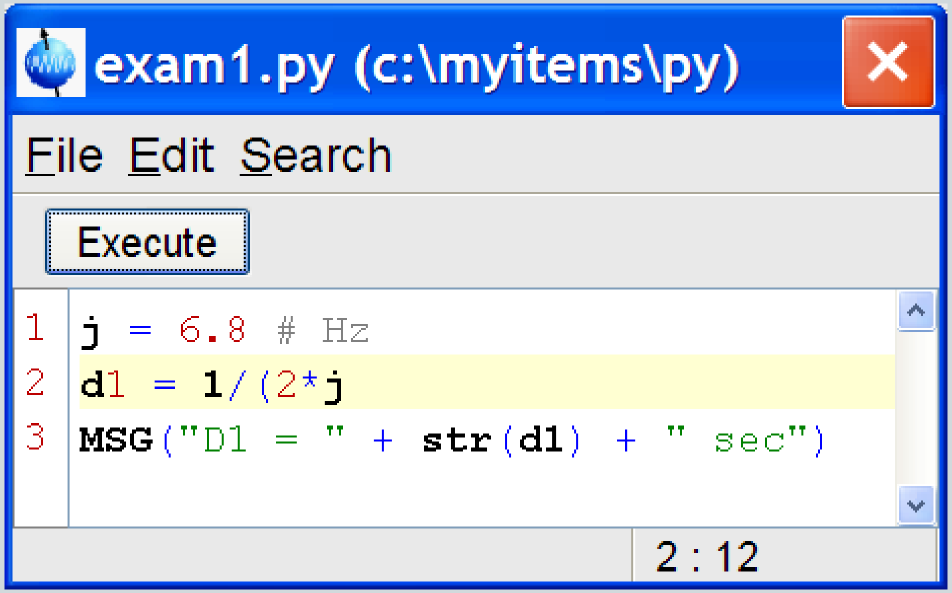
```
j = 6.8  Hz
d1 = 1/(2*j)
MSG("D1 = " + str(d1) + " sec")
```

We forgot the comment sign # in line 1 before Hz

Don’t Give Up On Errors… Case 2

```
Traceback (innermost last):
  (no code object) at line 0
SyntaxError: ('invalid syntax', ('c:/myitems/py/exam1.py', 3, 1, 'MSG("D1 = " + str(d1) + " sec")'))
```

line number

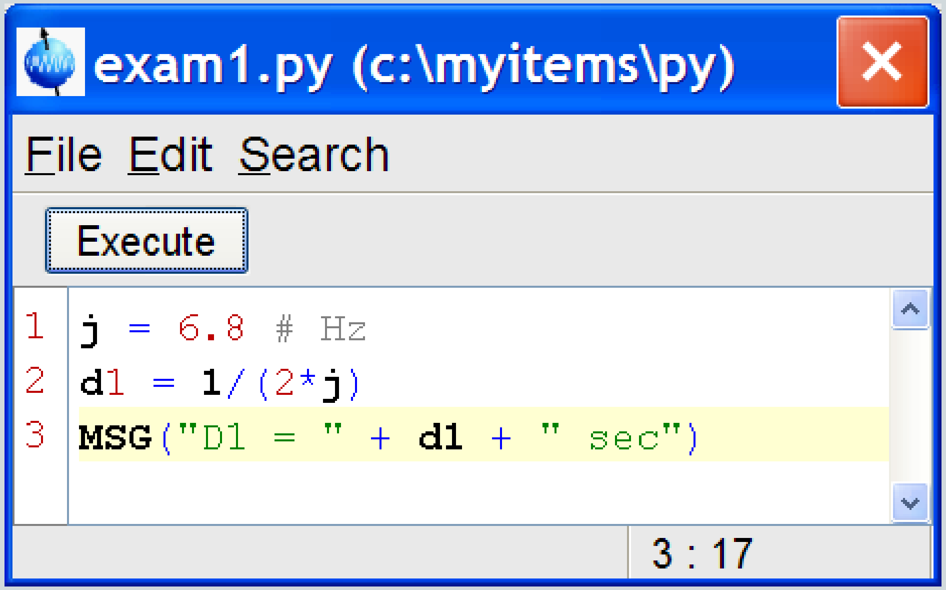


```
1 j = 6.8 # Hz
2 d1 = 1/(2*j
3 MSG("D1 = " + str(d1) + " sec")
```

Do not only examine the printed line, but also the surrounding: Here, the error occurs in line 2 (missing ')'), but the Python interpreter detected it not before line 3.

Don’t Give Up On Errors… Case 3

```
Traceback (innermost last):
  File "c:/myitems/py/exam1.py", line 3, in ?
TypeError: __add__ nor __radd__ defined for these operands
```



```
j = 6.8 # Hz
d1 = 1/(2*j)
MSG("D1 = " + d1 + " sec")
```

We forgot to use the str() function to print d1. Python tried to add a String to a Number, which caused the error message above

Lesson 1: Compute A TopSpin Pulse Shape

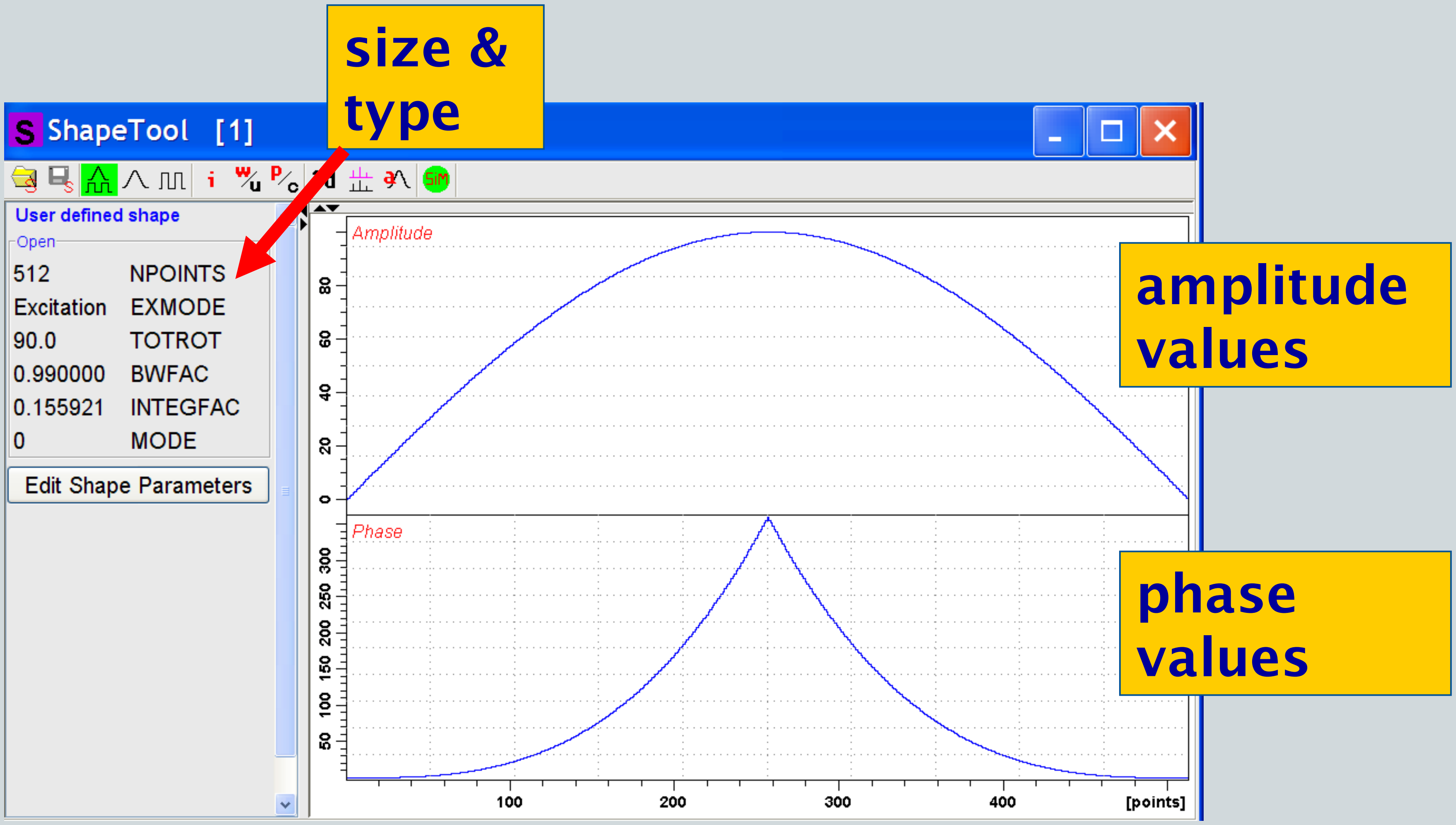
Tasks of Lesson 1

- Compute amplitudes and phases
- Save in a TopSpin shape file

What we will learn

- Multi-line comments, imports, math
- Python lists
- for-loops, if-branches, indenting
- SAVE_SHAPE TopSpin function

Lesson 1: The Shape To Be Computed

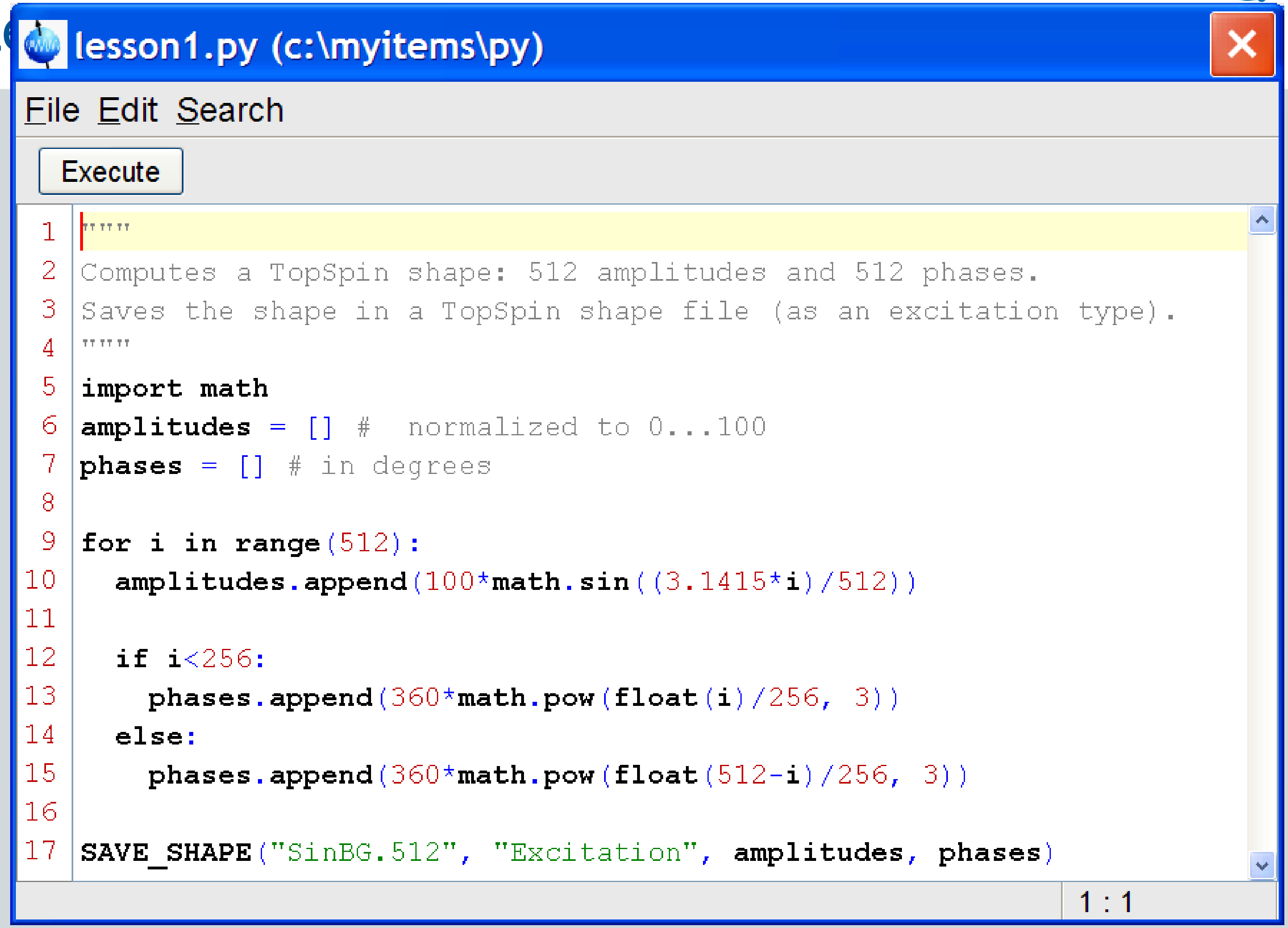


User defined shape

Open

512	NPOINTS
Excitation	EXMODE
90.0	TOTROT
0.990000	BWFAC
0.155921	INTEGFAC
0	MODE

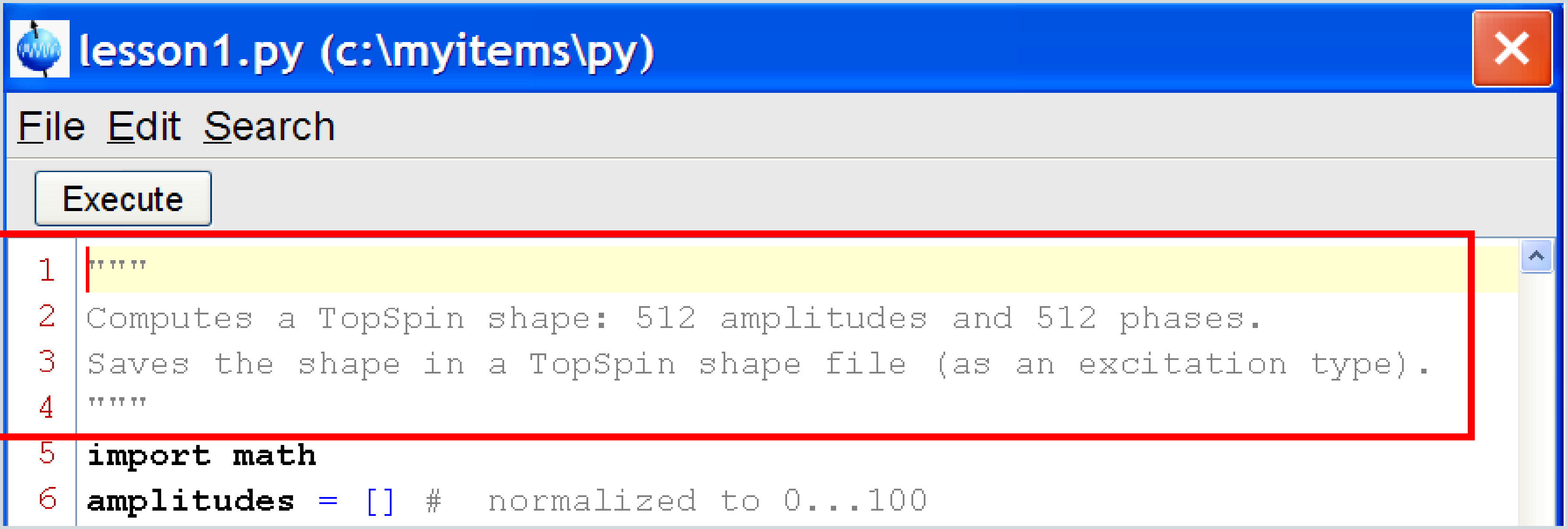
Le



```python
"""
Computes a TopSpin shape: 512 amplitudes and 512 phases.
Saves the shape in a TopSpin shape file (as an excitation type).
"""
import math
amplitudes = [] #  normalized to 0...100
phases = [] # in degrees

for i in range(512):
  amplitudes.append(100*math.sin((3.1415*i)/512))

  if i<256:
    phases.append(360*math.pow(float(i)/256, 3))
  else:
    phases.append(360*math.pow(float(512-i)/256, 3))

SAVE_SHAPE("SinBG.512", "Excitation", amplitudes, phases)
```

Bruker BioSpin

Lesson 1: Multi-Line Comments

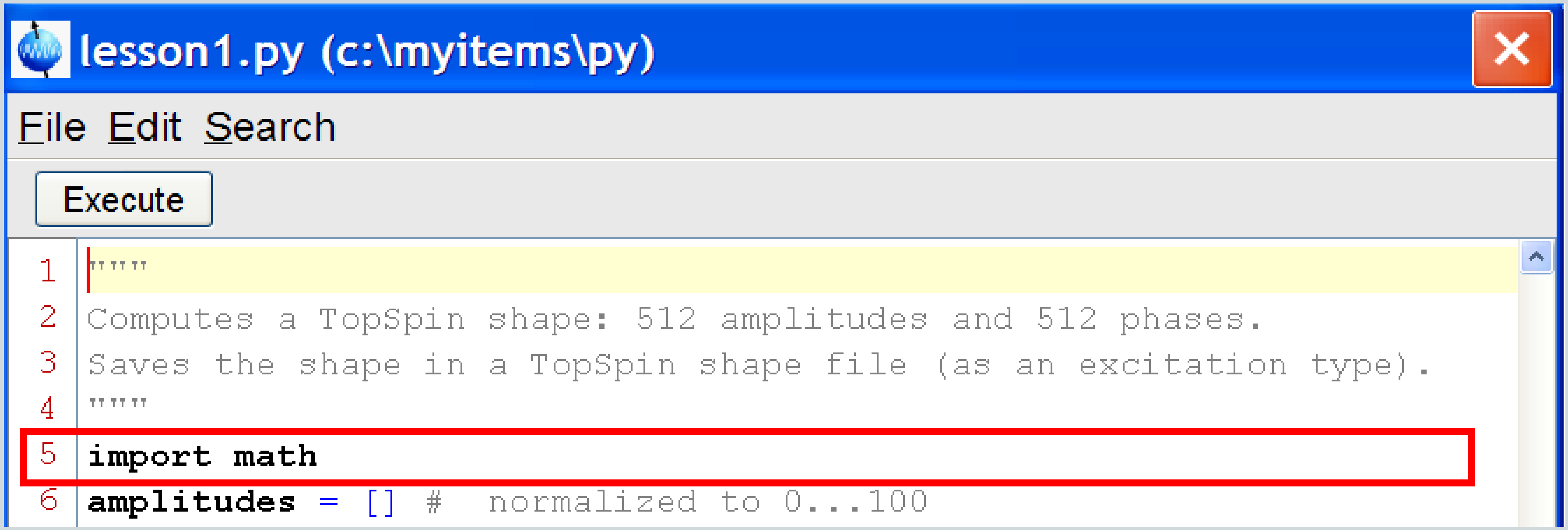


```
"""
Computes a TopSpin shape: 512 amplitudes and 512 phases.
Saves the shape in a TopSpin shape file (as an excitation type).
"""
import math
amplitudes = [] #  normalized to 0...100
```

Single-line comment: # starts the comment, the line end terminates it

Multi-line comment: """ starts and terminates the comment

Lesson 1: Importing Libraries

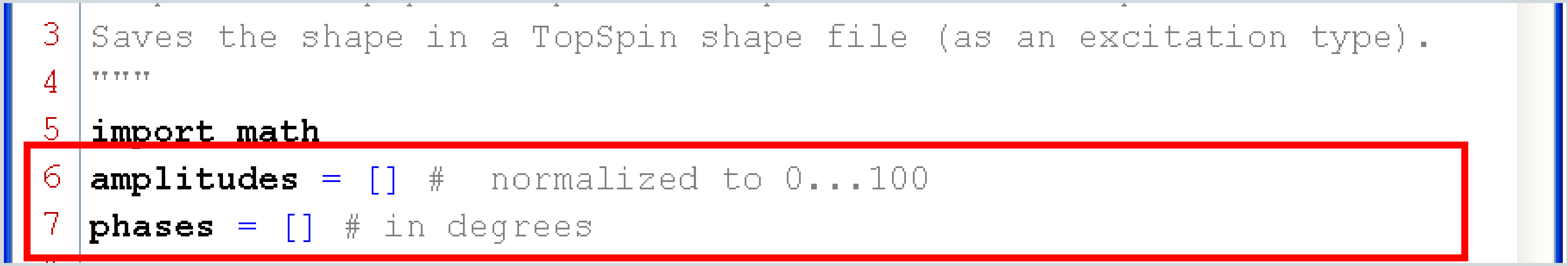


```
"""
Computes a TopSpin shape: 512 amplitudes and 512 phases.
Saves the shape in a TopSpin shape file (as an excitation type).
"""
import math
amplitudes = [] #  normalized to 0...100
```

This program uses the *sine()* function, which is not part of the standard Python library, but of the *math* library.

***import math* makes all functions of this library available in this Python program**

Lesson 1: Lists



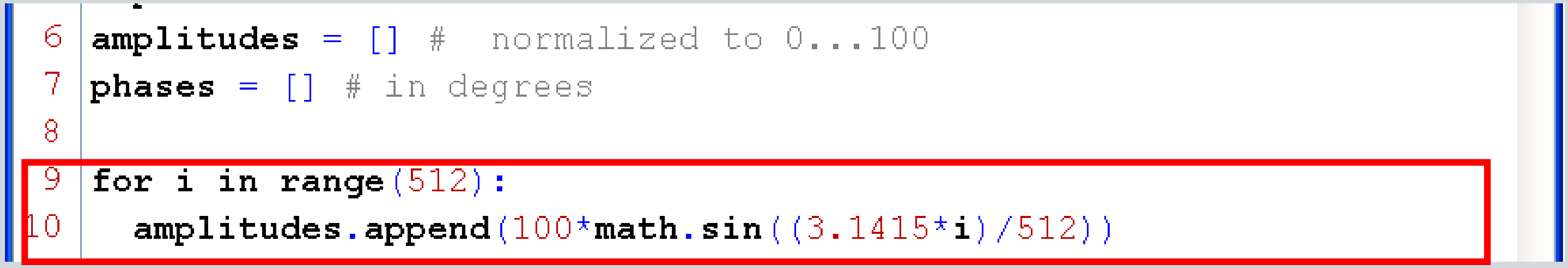
```
Saves the shape in a TopSpin shape file (as an excitation type).
"""
import math
amplitudes = [] #  normalized to 0...100
phases = [] # in degrees
```

Lists are the Array equivalent of Python, however

(1) no list size needs to be specified (in C: double phases[512])
(2) a list can grow by appending elements (arbitrary objects)
phases = [] defines an empty list
phases = [0, 90, 180, 270] defines a list with 4 elements
phases = [0, 90, 180, 270]*2 defines a list with 8 elements, repeating the first 4.

phases = [0, “a text“, 180, 270] is possible, text/numbers mixed
(3) indexing: phases[0] equals 0, phases[1] equals “a text“, etc.

Lesson 1: for-Loops, Indenting



```
amplitudes = [] #  normalized to 0...100
phases = [] # in degrees

for i in range(512):
  amplitudes.append(100*math.sin((3.1415*i)/512))
```

- All lines after for which are indented with respect to for (by space or tab) belong to the loop.
- i = 0, 1, …., 512-1

math.sin(…) calls the sine function of the math library

amplitudes.append(...) appends the computed value to the amplitude list, thereby growing the list which contained no elements initially

Lesson 1: if-Branches

```python
for i in range(512):
  amplitudes.append(100*math.sin((3.1415*i)/512))

  if i<256:
    phases.append(360*math.pow(float(i)/256, 3))
  else:
    phases.append(360*math.pow(float(512-i)/256, 3))
```

All lines after if/else which are indented with respect to if/else (by space or tab) belong to the respective branch.

Lesson 1: SAVE_SHAPE TopSpin Function

```
for i in range(512):
  amplitudes.append(100*math.sin((3.1415*i)/512))

  if i<256:
    phases.append(360*math.pow(float(i)/256, 3))
  else:
    phases.append(360*math.pow(float(512-i)/256, 3))

SAVE_SHAPE("SinBG.512", "Excitation", amplitudes, phases)
```

Saves the shape defined by the computed lists amplitudes and phases on disk:
Filename = "SinBG.512"
Shape type = Excitation, could also be Refocussing, Inversion
The result can be viewed using Shapetool (stdisp).

TopSpin Functions: Processing/Acquisition

Processing & Acquisition

- For most TopSpin commands a corresponding Python function is available: EM(), EF(), ZG(), XFB(), …
- If not, use XCMD(…), e.g.
  XCMD(".int") enters integration mode
  XCMD("lb") opens the LB (line broadening) dialog
- XCPR(…) sends the specified command directly to the TopSpin cpr module: It is equivalent to CPR_exec(…) in AU programs.

TopSpin Functions: GETPAR, PUTPAR

Parameter Handling

Functions: GETPAR(…) and PUTPAR(…)

Example Program Using GETPAR

```
si = GETPAR("SI") # get SI as a String! ("32768")
doubledSize = 2*int(si) # convert to integer for calc.!
MSG("result="+str(doubledSize)) # print result
```

Remember

- GETPAR delivers a String, not a number
- Use the function int(String) or float(String) to convert the String to a number before performing calculations

TopSpin Functions: GETPAR, PUTPAR

Example Program Using PUTPAR

```
d1 = 1 / (2*6.8) # compute a value
PUTPAR("D 1", str(d1)) # MUST store it as a String!
PUTPAR("P 1", "1.37") # Set P1 to 1.37 microsec
```

Remember

- PUTPAR requires parameters as a String, not a number
- Use the function str(Number) to convert a number to a String before storing a value
- Indexed parameters such as Delays must have a space between the parameter name and the index (see above). This applies to GETPAR and PUTPAR.

TopSpin Functions: GETPAR, PUTPAR

PUTPAR and GETPAR For 2D, 3D, …

```
PUTPAR(“SI", “1024”) # set acquisition dimension
PUTPAR(“1 SI", “256") # Set other dimension, here F1
```

PUTPAR and GETPAR For Status Parameters

```
ssi = GETPAR(“status SI”) # get acquisition dimension
PUTPAR(“1s SI", “256") # Set other dimension, here F1
```

TopSpin Functions: Changing The Dataset

```
EF()
APK()
```

This Python program operates on the currently displayed dataset, because no data are defined explicitely.

TopSpin Functions: Changing The Dataset

```
dataset = \
 ["exam1d_13C", "2", "1", "c:/bruker/topspin", "guest"]
RE(dataset)
EF()
APK()
```

This Python program defines the dataset as a Python list […].

All list elements are Strings, also EXPNO and PROCNO!

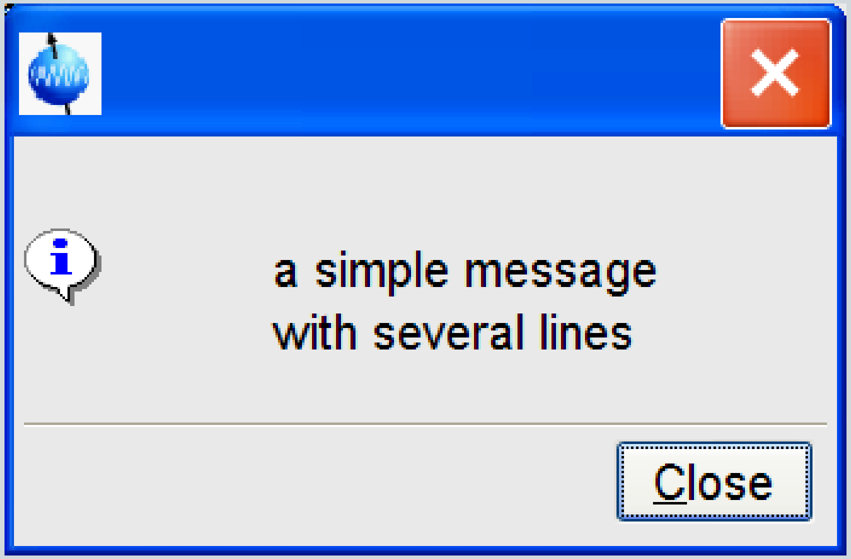
\ (backslash followed by Enter) indicates line continuation.

RE sets the current data.

```
dataset = \
"c:/bruker/topspin/data/guest/nmr/exam1d_13C/2/pdata/1"
RE_PATH(dataset) # Alternative to RE(..)
```

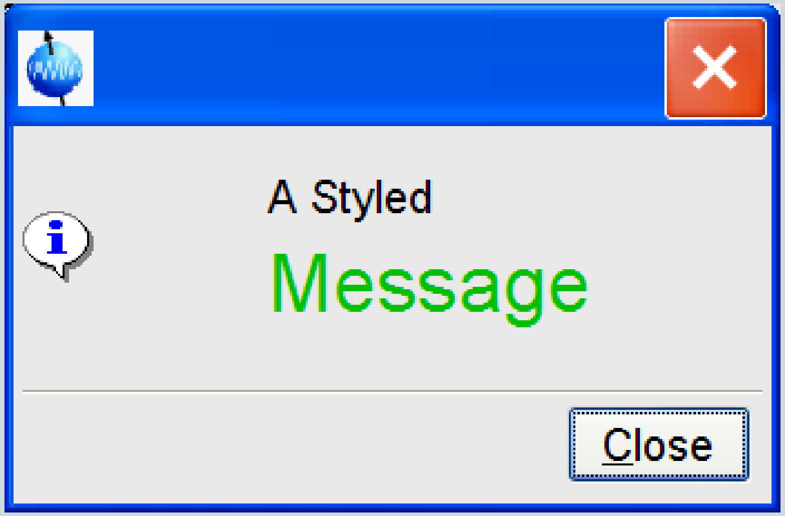
TopSpin Functions: MSG Dialog

```
MSG("a simple message\nwith several lines");
```


a simple message
with several lines

TopSpin Functions: MSG Dialog

```
MSG("<html>A Styled<br>" +\
   "<font size=28><font color=\"00BF00\">"+\
   "Message</font></html>")
```



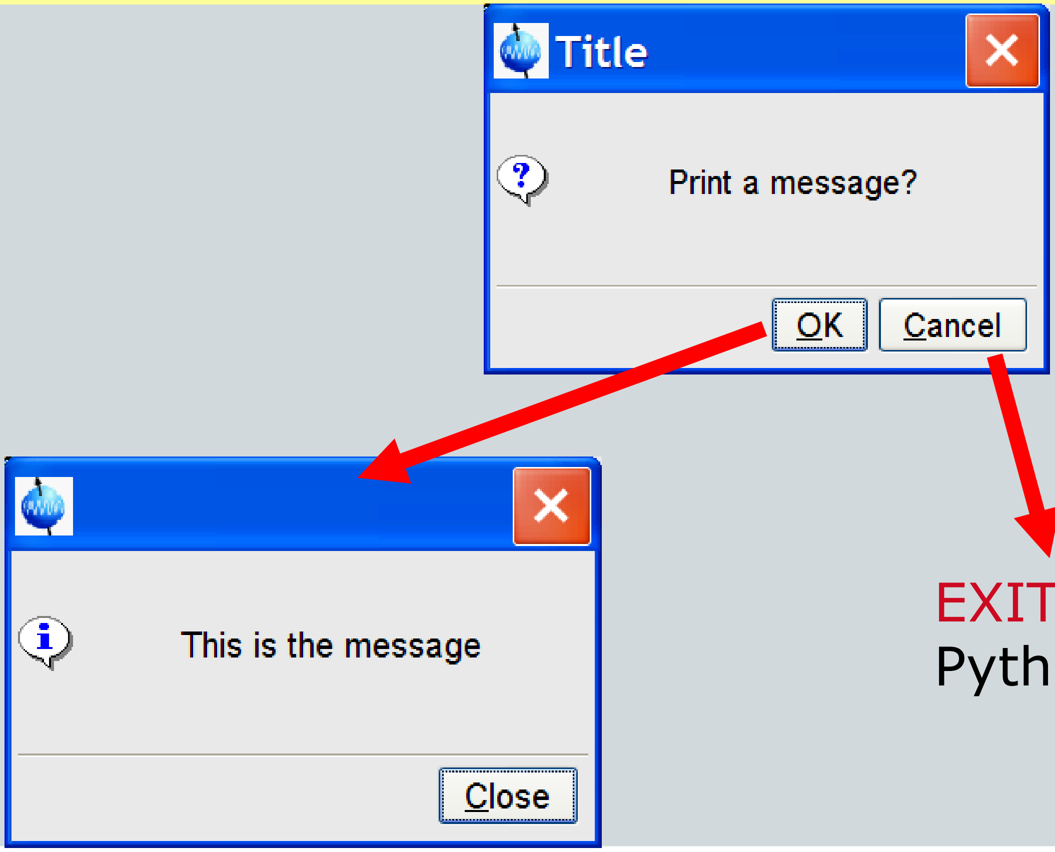
A Styled

Message

A Message text may be written in html format

TopSpin Functions: CONFIRM Dialog

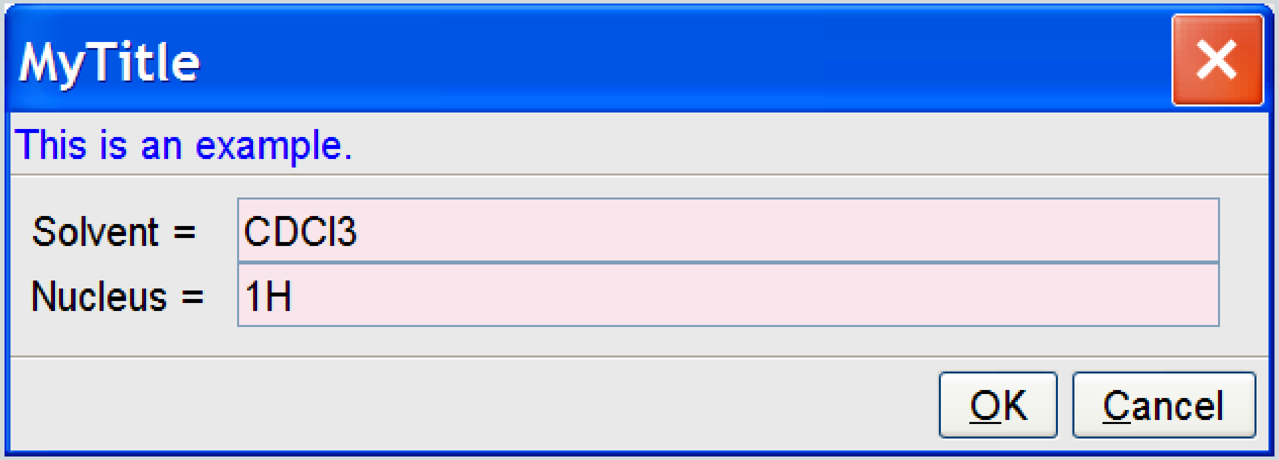
```
if CONFIRM("Title", "Print a message?") == 0:
      EXIT()
MSG("This is the message")
```



EXIT: Terminate Python program

TopSpin Functions: INPUT Dialog

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



This is an example.

Solvent = CDCl3
Nucleus = 1H

Can display arbitray number of input text fields

TopSpin Functions: Reading NMR Data

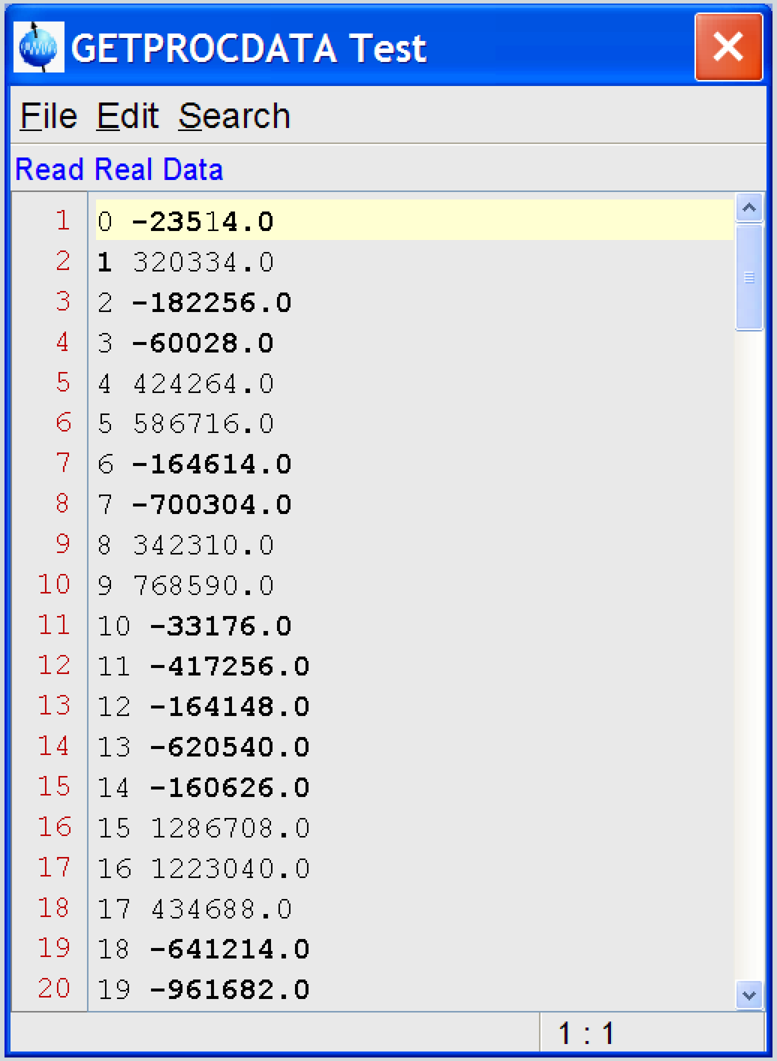
```
result = GETPROCDATA(-0.5, 0.5)
text = ""
for i in range(len(result)):
      text += str(i) + " " + str(result[i]) + "\n"
VIEWTEXT("GETPROCDATA Test", "Read Real Data", text)
```

GETPROCDATA(…) reads the current data from the specified range (in ppm).

The result is a list of float values (Python float = C double)

The rest of this program display the values in a text viewer window using the TopSpin function VIEWTEXT(..), see next page.

TopSpin Functions: Reading NMR Data



Read Real Data

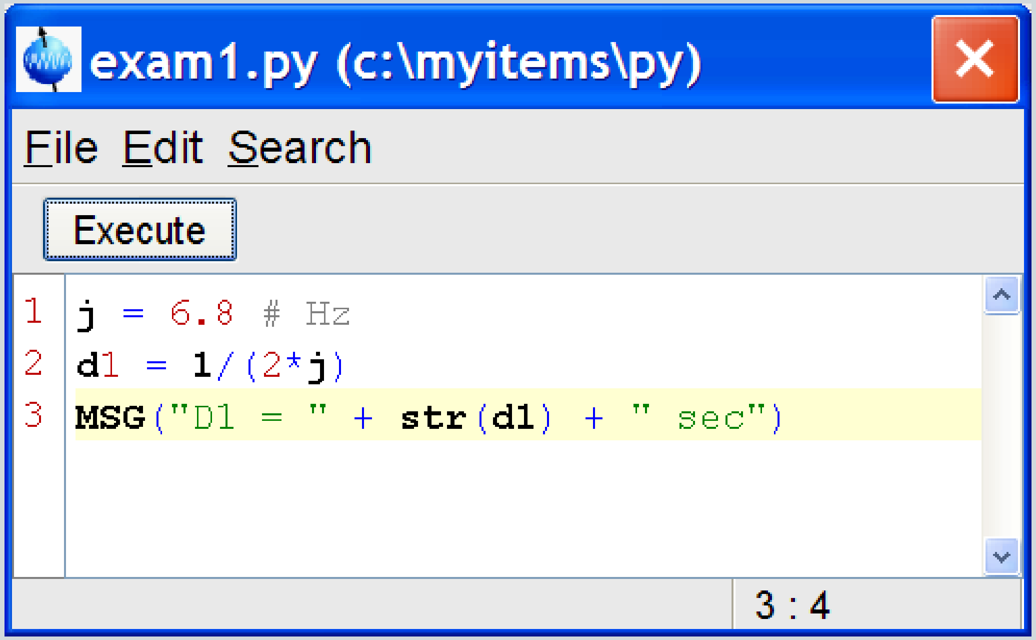
```
0 -23514.0
1 320334.0
2 -182256.0
3 -60028.0
4 424264.0
5 586716.0
6 -164614.0
7 -700304.0
8 342310.0
9 768590.0
10 -33176.0
11 -417256.0
12 -164148.0
13 -620540.0
14 -160626.0
15 1286708.0
16 1223040.0
17 434688.0
18 -641214.0
19 -961682.0
```

Result window of previous page

In additon to GETPROGDATA, there is also a function GETPROCDATA2D

Python Functions

So far, our Python program only had a „main" program without an internal structure (functions or subroutines, classes, …)



```
j = 6.8 # Hz
d1 = 1/(2*j)
MSG("D1 = " + str(d1) + " sec")
```

Python Functions

```
"""
Computes a TopSpin shape: Uses a Python function.
"""
import math
def calcShape(size):
  amplitudes = [] #  normalized to 0...100
  phases = [] # in degrees
  for i in range(size):
    amplitudes.append(100*math.sin((3.1415*i)/size))

    if i<256:
      phases.append(360*math.pow([illegible]/2), 3))
    else:
      phases.append(360*math.pow(float(size-i)/256, 3))
  return amplitudes, phases

amplitudes, phases = calcShape(512)
SAVE_SHAPE("SinBG1.512", "Excitation", amplitudes, phases)
```

Define function with args

Return a list of results

Call function, get results

Python Literature

This tutorial only covered the very basics of Python!

Manuals coming with TopSpin, accessible via TopSpin Help:
- this tutorial
- Python Programming (particularly the TopSpin functions)
- Python Introduction (into the language)
- Pulse Programming with Python

Books:
- Jython Essentials (by S. Pedroni, N. Rappin)

WEB:
- Jython Tutorial (by B. Feigenbaum):
  http://www-128.ibm.com/developerworks/edu/j-dw-java-jython1-i.html

Python and Jython

TopSpin uses the Java variant of Python (= Jython)

(for easy integration into TopSpin)

Traditional Python is the C variant

The language is the same for both variants.
There are a few incompatibilties, decribed in the book.

Jython has its own WEB site:

www.jython.org