Introduction to Python

Eduardo Rodrigues
University of Glasgow

Course given at Glasgow, 12-18 March 2008

Part II

ROOT & RooFit in Python

Part II: ROOT & RooFit in Python

I. Interfacing C++ to Python in HEP

II. Environment set-up

III. ROOT in Python

IV. RooFit in Python

Interfacing C++ to Python in High Energy Physics

Python and dictionaries

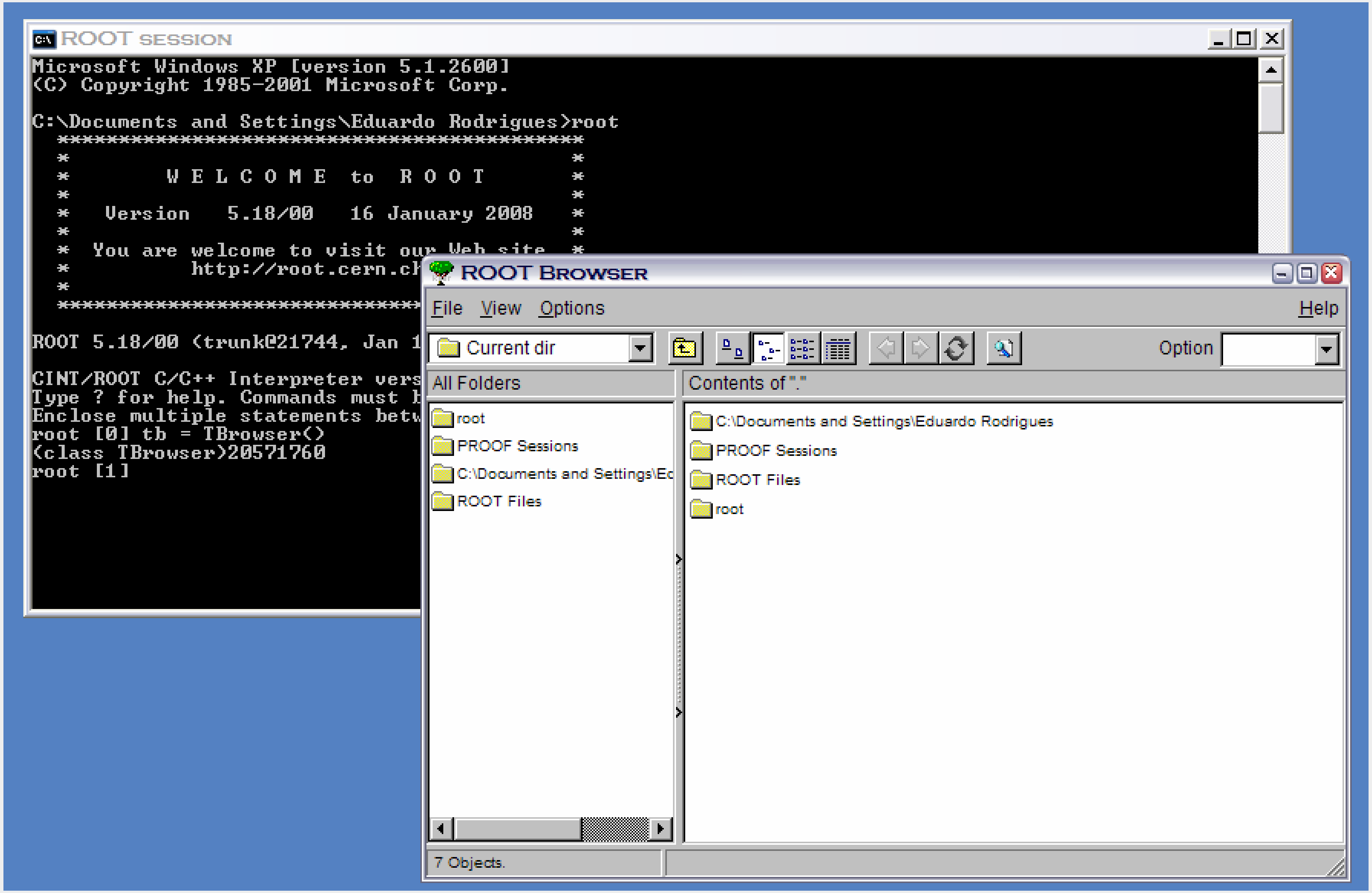
Python and C++ bindings

- Remember Python is a neat “glue” language
- Python knows about C++ objects via dictionaries
- Note: these dictionaries are C++ libraries; nothing to do with Python dictionary class !
- All is nicely done “behind the scenes”
- These dictionaries are DLLs containing “reflection information”,
- i.e. information about the C++ classes needed by Python
- Reflection is what HEP does, in particular ROOT …

- Reflection = ability of a language to introspect its own structures at runtime and interact with them in a generic way

Environment set-up

ROOT from the command prompt / shell



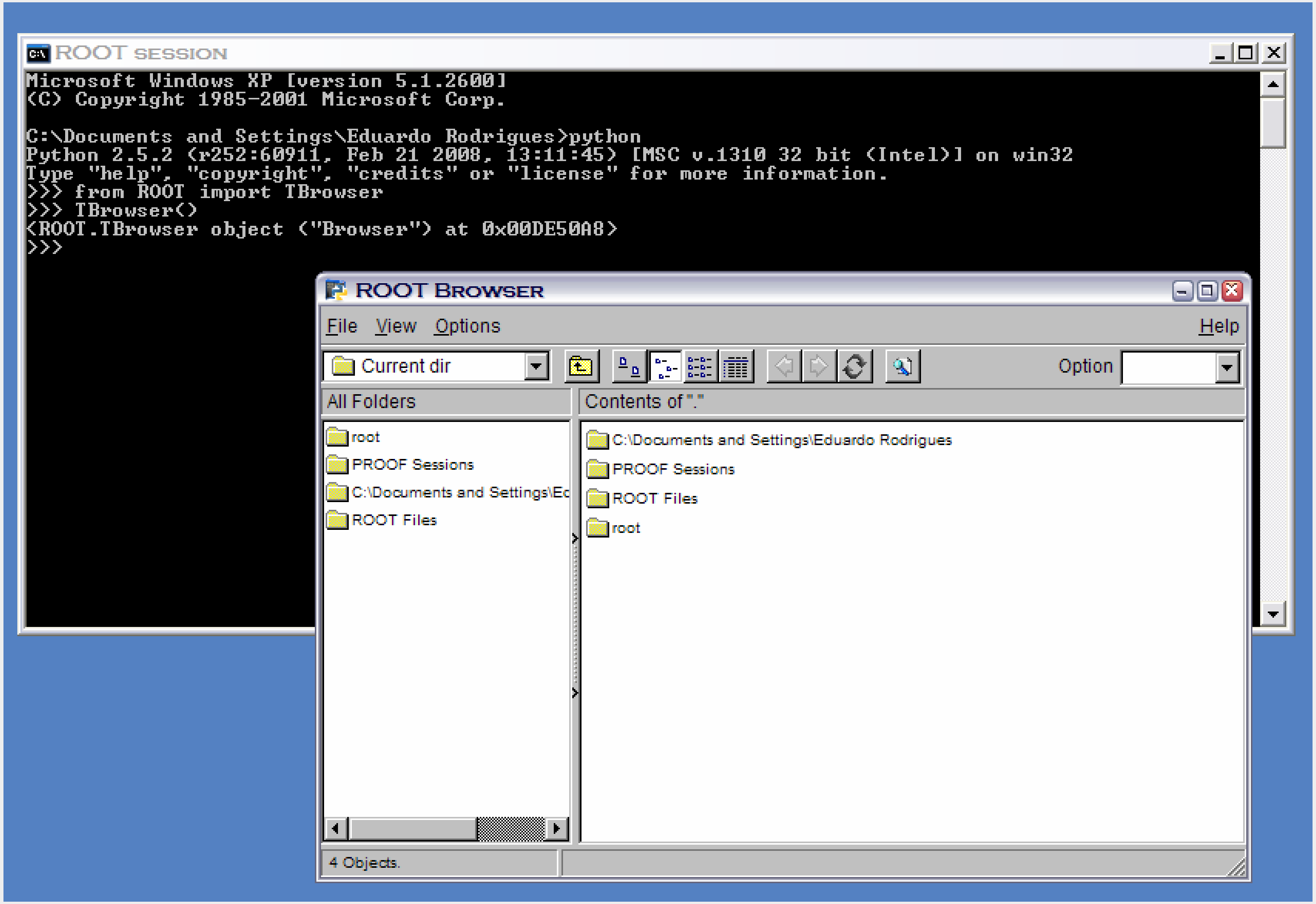
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Eduardo Rodrigues>root
  *******************************************
  *                                         *
  *        W E L C O M E  to  R O O T       *
  *                                         *
  *   Version   5.18/00   16 January 2008   *
  *                                         *
  *  You are welcome to visit our Web site  *
  *          http://root.cern.ch
  *
  ******************************

ROOT 5.18/00 (trunk@21744, Jan 1

CINT/ROOT C/C++ Interpreter vers
Type ? for help. Commands must b
Enclose multiple statements betw
root [0] tb = TBrowser()
(class TBrowser)20571760
root [1]
```

ROOT from the Python prompt



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Eduardo Rodrigues>python
Python 2.5.2 (r252:60911, Feb 21 2008, 13:11:45) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from ROOT import TBrowser
>>> TBrowser()
<ROOT.TBrowser object ("Browser") at 0x00DE50A8>
>>>
```

Start-up file

- ❑ Handy to set the environment variable PYTHONSTARTUP to a script where some commands are executed everytime Python starts … (not the case when running as a script!)
- ❑ Particularly handy to “automatically” make ROOT available to Python, which requires ROOT being in the Python search path for modules:

```
# my start-up script
# makes Python aware of ROOT
import os, sys

pyrootpath = os.environ[ ‘ROOTSYS’ ]

if os.path.exists( pyrootpath ) :
    sys.path.append( pyrootpath + os.sep + ‘bin’ )
    sys.path.append( pyrootpath + os.sep + ‘lib’ )
```

ROOT in Python

Importing ROOT classes

```
# default import
>>> import ROOT

>>> dir(ROOT)

# import “everything” from ROOT. Actually import the basics, only
>>> from ROOT import *

# import specific classes
>>> from ROOT import TBrowser, TH1
```

ROOT looks gets personal

```
from ROOT import gROOT, TStyle

# user-defined function
def rootSettings():
    global myStyle
    myStyle = TStyle( 'myStyle', 'My personal ROOT style' )

    myStyle.SetCanvasColor( 0 )
    myStyle.SetPadColor( 0 )
    myStyle.SetOptStat( 111111 )
    myStyle.SetOptFit( 1111 )

    gROOT.SetStyle( 'myStyle' )
    gROOT.ForceStyle()
```

Fit with a user-defined function

```
from ROOT import TF1

# define a fitting function
def fit_histo( h ):
    max = h.GetMaximum()
    doubleGaussian = TF1( "Double Gaussian", "gaus(0) + gaus(3)" )
    doubleGaussian.SetParameters( max, h.GetMean(), h.GetRMS(),
                                  max/100., h.GetMean(), h.GetRMS()*10. )
    h.Fit( doubleGaussian )
    return doubleGaussian.GetParameter(2) # RMS of core Gaussian
```

```
# “histo” holds a histogram …
fit_histo( histo )

# draw the histogram together with the fitted function
histo.Draw()
```

Some examples to test …

```
# import the necessary classes
from ROOT import gROOT, gRandom
from ROOT import TCanvas, TF1, TH1F

gROOT.Reset()

c1 = TCanvas( 'c1', 'Example with Formula', 200, 10, 700, 500 )

# Create a one dimensional function and draw it
fun1 = TF1( 'fun1', 'abs(sin(x)/x)', 0, 10. )
c1.SetGridx()
c1.SetGridy()
fun1.Draw()
c1.Update()

# Gaussian histogram
c2 = TCanvas( 'c2', 'Example', 200, 10, 700, 500 )
hpx = TH1F( 'hpx', 'px', 100, -4, 4 )
for i in xrange( 25000 ):
    px = gRandom.Gaus()
    hpx.Fill( px )

hpx.Draw()
c2.Update()
```

Using a user ROOT C++ class in Python

Ability to use someone else’s C++ class directly in your Python scripts

```
# import the necessary classes
from ROOT import gROOT

gROOT.LoadMacro( MyCplusplusClass.cxx+’ )


# import MyCplusplusClass as if it were a standard Python module
from ROOT import MyCplusplusClass
```

- **This is particularly powerful ;-) !**

RooFit in Python

Importing RooFit classes (1/2)

- ❑ Many things to try …

```
>>> import ROOT
>>> ROOT.RooRealVar
>>> ROOT.RooFit

>>> from ROOT import *
>>> RooRealVar
>>> RooFit

>>> from ROOT import RooRealVar
>>> RooRealVar
>>> from ROOT import RooFit
>>> RooFit

>>> from ROOT import
```

- ❑ … and the results might be surprising depending on which order one imports RooFit and anything else …
- ❑ Check the namespaces …

Importing RooFit classes (2/2)

```
 # this import only makes available some basic ROOT classes
 >>> import ROOT

 # explicitly load the RooFit library
 >>> from ROOT import gSystem
 >>> gSystem.Load( ‘libRooFit’ )
 ←[1mRooFit v2.31 -- Developed by Wouter Verkerke and David Kirkby←[0m
 Copyright (C) 2000-2008 NIKHEF, University of California & Stanford University
  All rights reserved, please read http://roofit.sourceforge.net/license.txt
 0
 >>> gSystem.Load( ‘libRooFit’ )
 Note: (file "", line 0) File "C:\Users\Software\root\bin\libRooFit.dll"
already loaded
 1

 # the RooFit classes are now available ... in the ROOT namespace
 >>> ROOT.RooFit
 <class ‘ROOT.RooFit’>
 >>> ROOT.RooRealVar
 <class ‘ROOT.RooRealVar’>
```

Simple C++ - Python RooFit comparisons

Elementary operations on a Gaussian PDF

```
{
 // Build Gaussian PDF
 RooRealVar x( "x", "x", -10, 10 );
 RooRealVar mean( "mean", "mean of gaussian", -1 );
 RooRealVar sigma( "sigma", "width of gaussian", 3 );
 RooGaussian gauss( "gauss", "gaussian PDF",
                    x, mean, sigma );

 // Plot PDF
 RooPlot* xframe = x.frame();
 gauss.plotOn( xframe );
 xframe->Draw();

 // Generate a toy MC set
 RooDataSet* data = gauss.generate( x, 10000 );

 // Plot PDF and toy data overlaid
 RooPlot* xframe2 = x.frame();
 data->plotOn( xframe2 );
 gauss.plotOn( xframe2 );
 xframe2->Draw();

 // Fit PDF to toy
 mean.setConstant( kFALSE );
 sigma.setConstant( kFALSE );
 gauss.fitTo( *data, "mh" );

 // Print final value of parameters
 mean.Print();
 sigma.Print();
}
```

```
from ROOT import RooRealVar, RooArgSet, RooLinkedList
From ROOT import RooGaussian
from ROOT import kFALSE

# Build Gaussian PDF
x     = RooRealVar(  'x',      'x',                -10, 10 )
mean  = RooRealVar(  'mean',  'mean of gaussian', -1 )
sigma = RooRealVar(  'sigma', 'width of gaussian', 3 )

gauss = RooGaussian( 'gauss', 'gaussian PDF', \
                     x, mean, sigma )

# Plot PDF
xframe = x.frame()
gauss.plotOn( xframe )
xframe.Draw()

# Generate a toy MC set
data = gauss.generate( RooArgSet(x), 10000 )

# Plot PDF and toy data overlaid
xframe2 = x.frame()
data.plotOn( xframe2, RooLinkedList() )
gauss.plotOn( xframe2 )
xframe2.Draw()

# Fit PDF to toy
mean.setConstant( kFALSE )
sigma.setConstant( kFALSE )
gauss.fitTo( data, 'mh' )

# Print final value of parameters
mean.Print()
sigma.Print()
```