ROOTaaS
ROOT as a Service

D. Piparo, E. Tejedor, P. Mato for the ROOT Team

PH-SFT

ROOT Users’ Workshop 2015

Introduction:
The “Notebook”

The Notebook

A web-based interactive computing platform that combines code, equations, text and visualisations.

Many supported languages: Python, Haskell, Julia... One generally speaks about a "kernel" for a specific language

In a nutshell: an "interactive shell opened within the browser"

Also called:
"Jupyter Notebook" or
"iPython Notebook"

http://www.jupyter.org

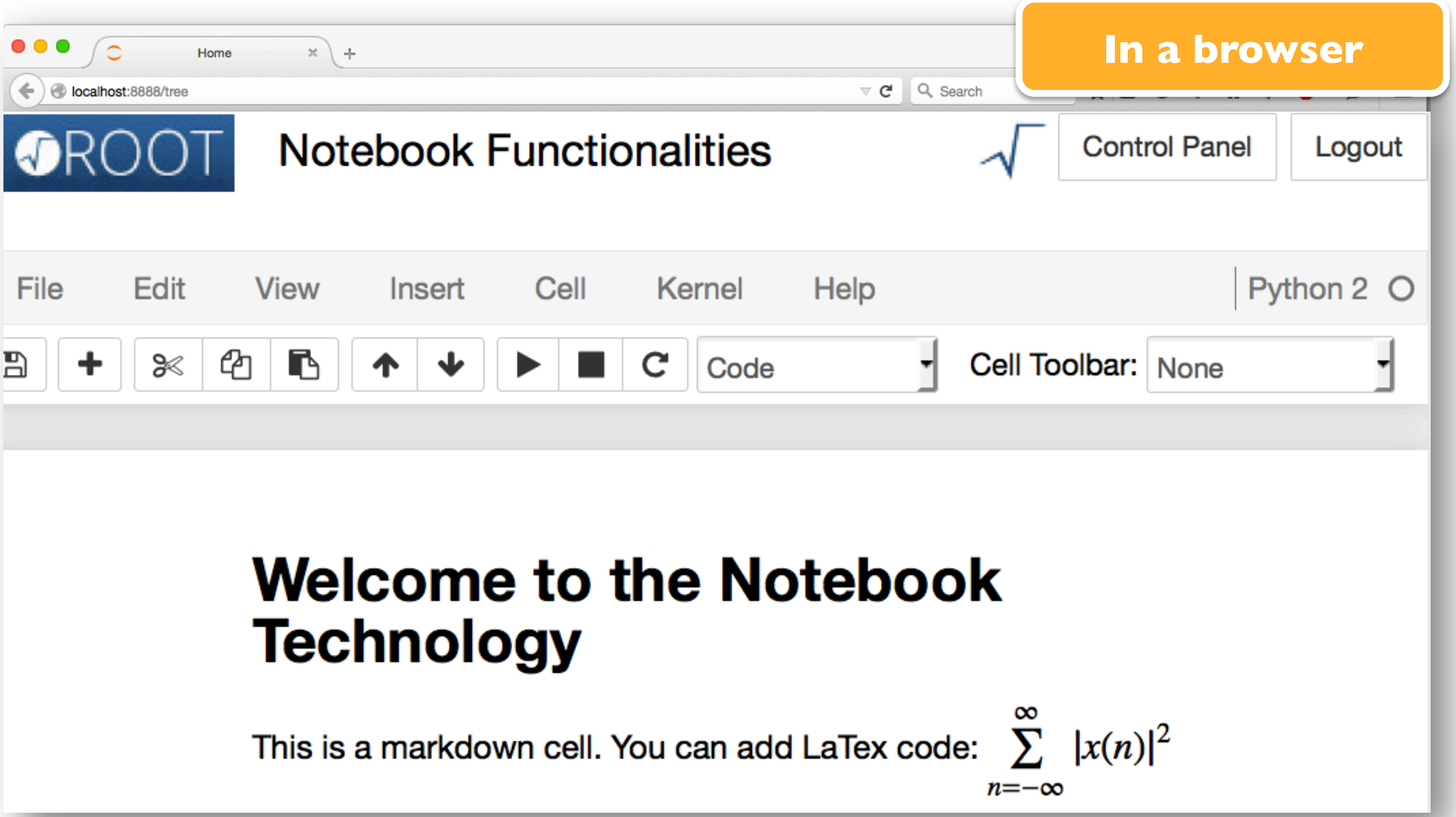
Start a Notebook in a Laptop

```
$ ipython notebook
```

That command:

1. Starts a notebook
2. Opens it in the browser

See backup for more details



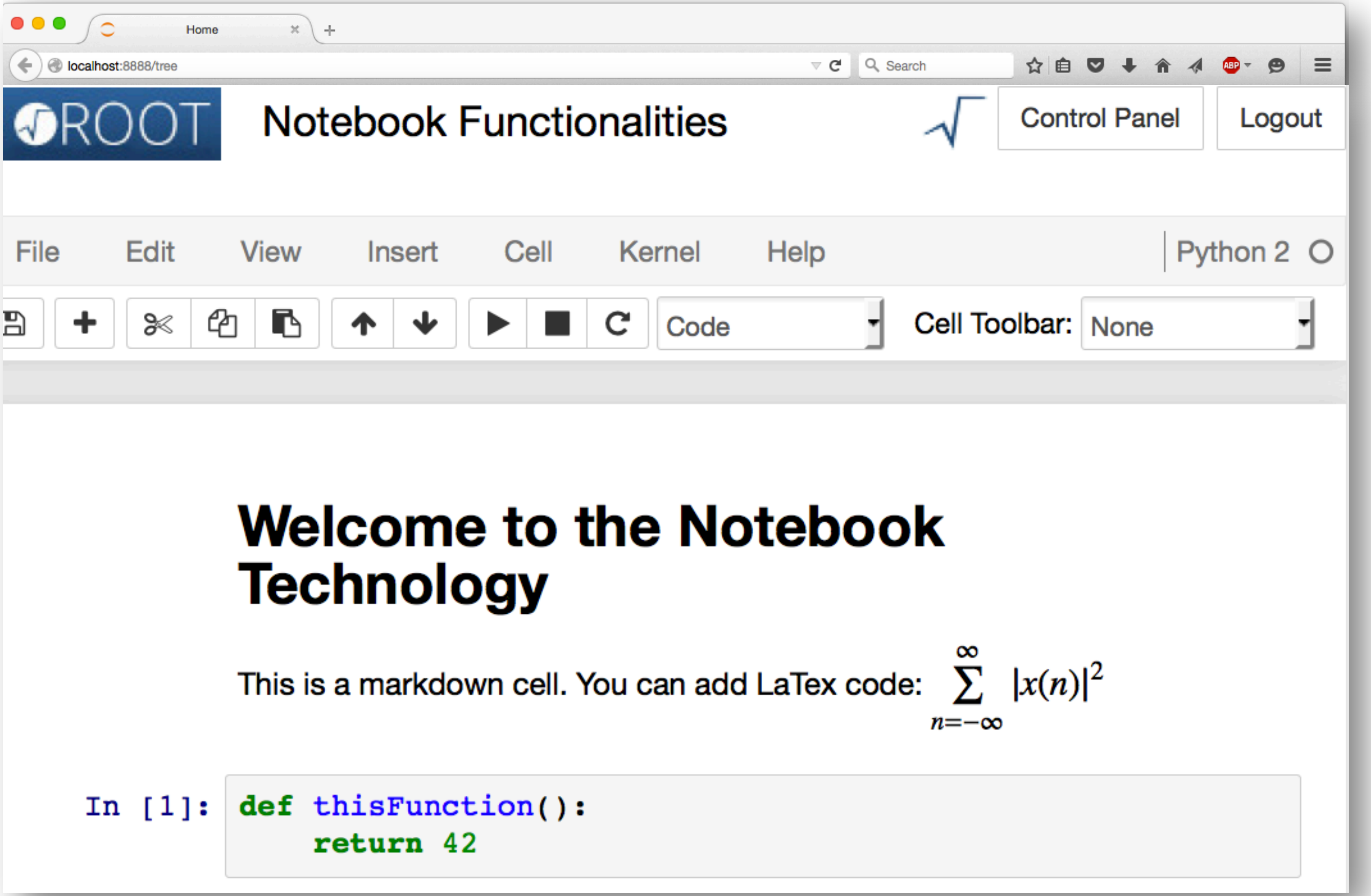
Cell Toolbar: None

Welcome to the Notebook Technology

This is a markdown cell. You can add LaTex code: $\sum_{n=-\infty}^{\infty} |x(n)|^2$

In a browser

Text and Formulas



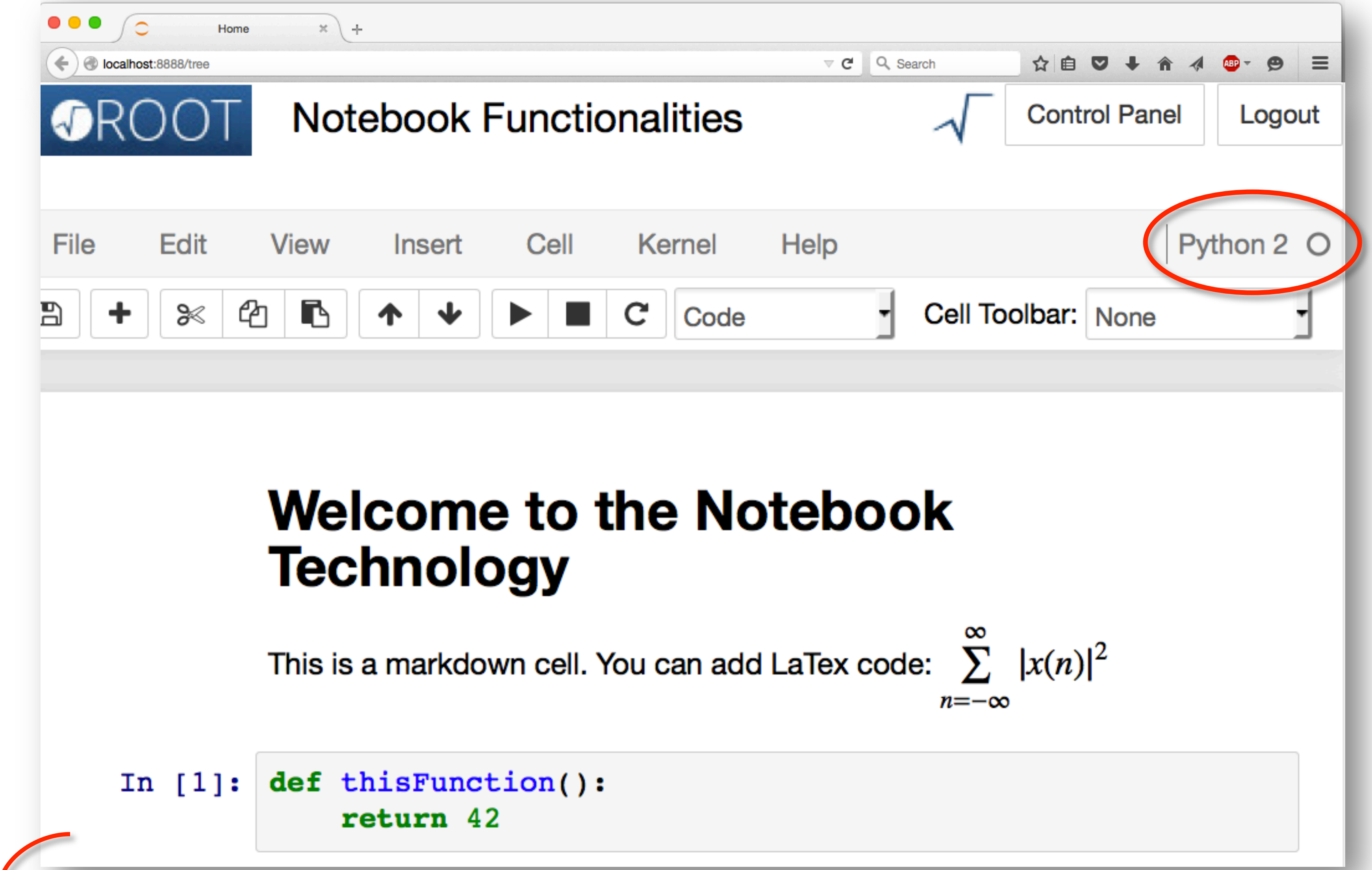
Cell Toolbar: None

Welcome to the Notebook Technology

This is a markdown cell. You can add LaTex code: $\sum_{n=-\infty}^{\infty} |x(n)|^2$

```
In [1]: def thisFunction():
            return 42
```

Code



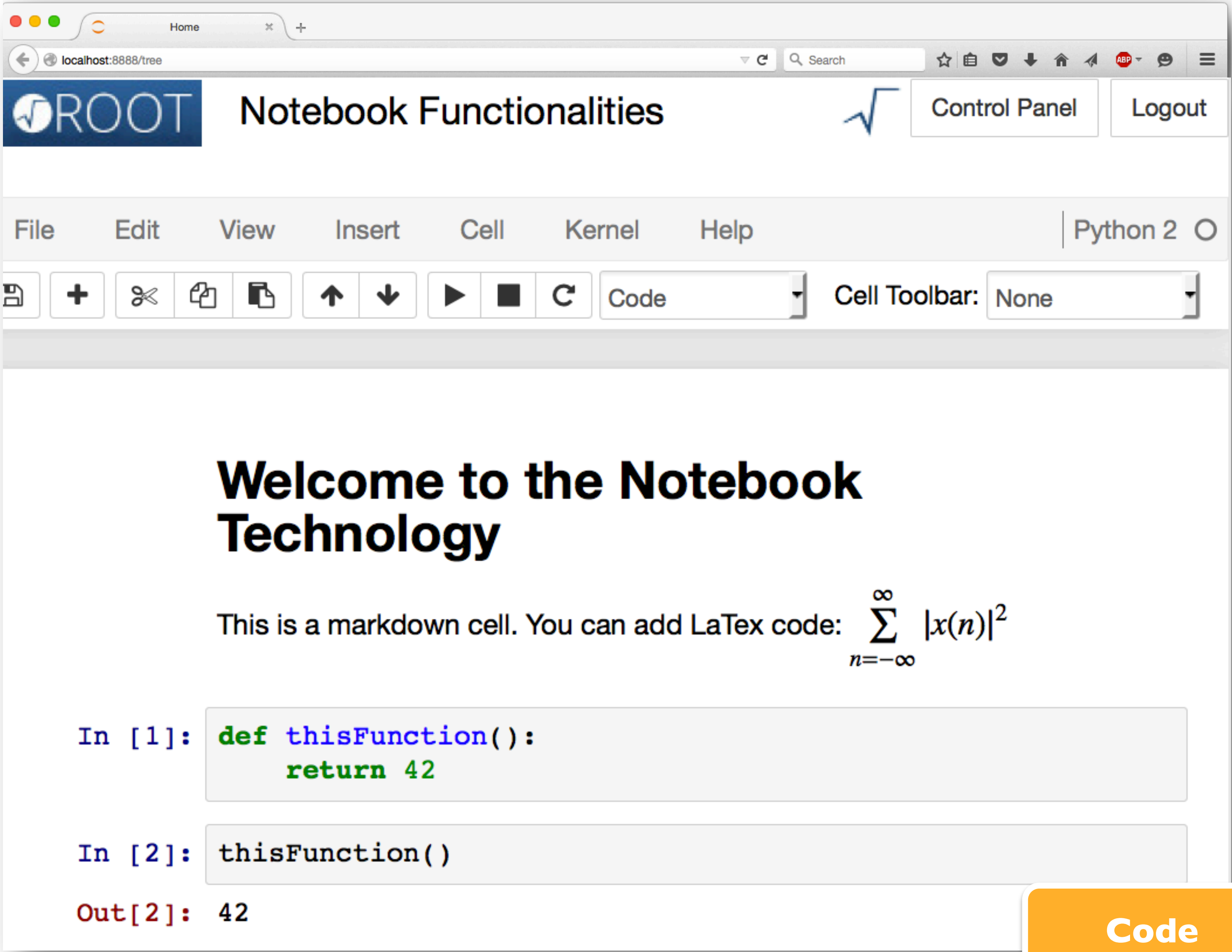
Welcome to the Notebook Technology

This is a markdown cell. You can add LaTex code: $\sum_{n=-\infty}^{\infty} |x(n)|^2$

```
In [1]: def thisFunction():
            return 42
```

This is a notebook in Python

Code



Welcome to the Notebook Technology

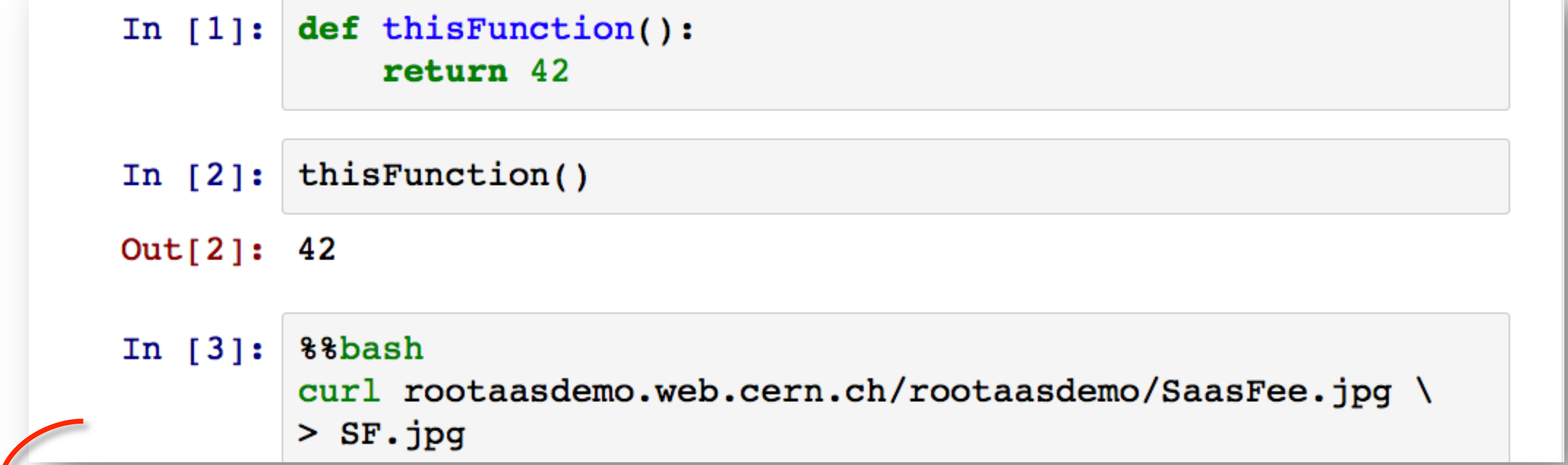
This is a markdown cell. You can add LaTex code: $\sum_{n=-\infty}^{\infty} |x(n)|^2$

In [1]:
```
def thisFunction():
    return 42
```

In [2]:
```
thisFunction()
```

Out[2]:
```
42
```

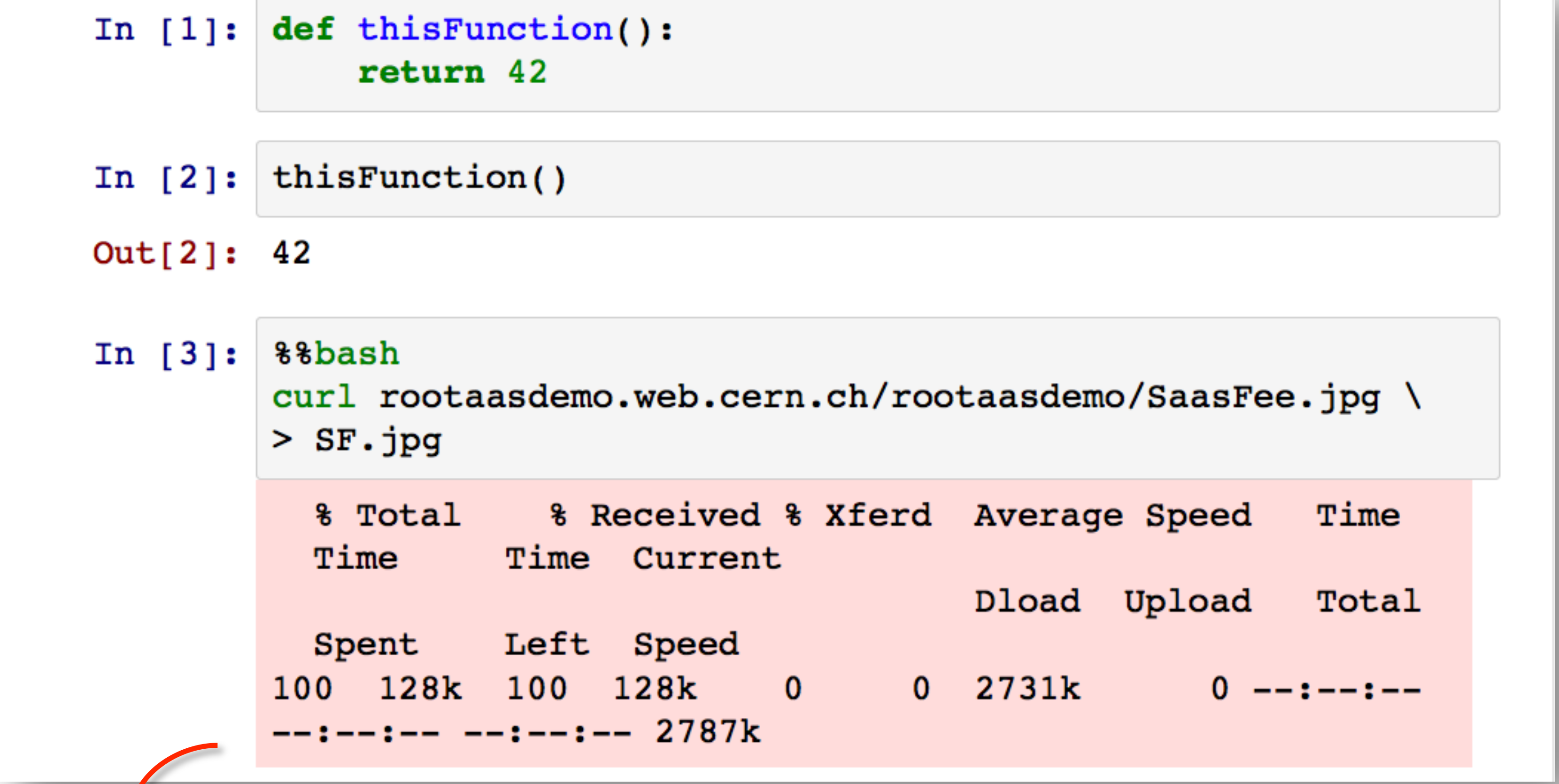
Code



```
In [1]: def thisFunction():
            return 42

In [2]: thisFunction()

Out[2]: 42

In [3]: %%bash
        curl rootaasdemo.web.cern.ch/rootaasdemo/SaasFee.jpg \
        > SF.jpg
```

We can invoke commands in the shell...

Shell Commands

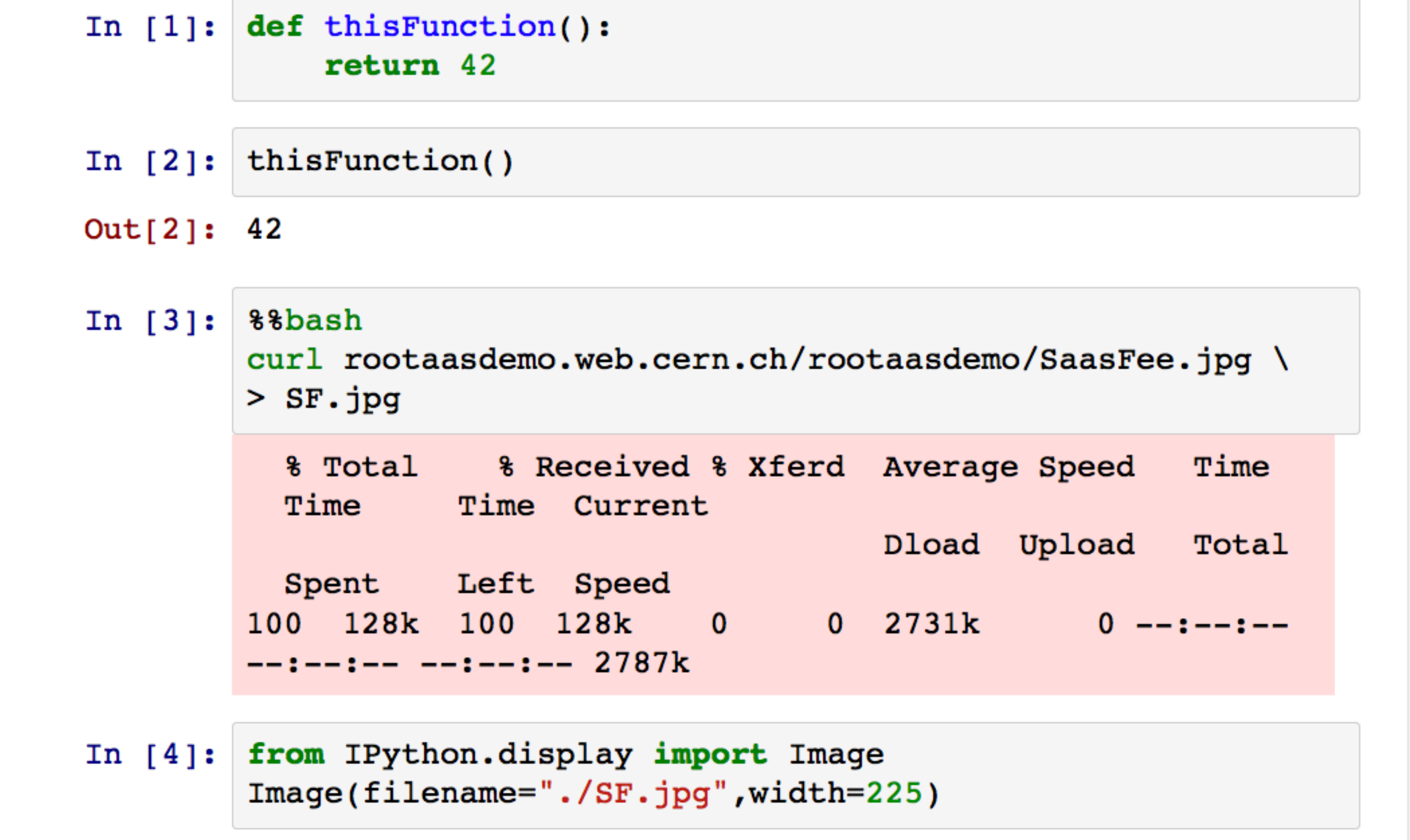


```
In [1]: def thisFunction():
            return 42

In [2]: thisFunction()

Out[2]: 42

In [3]: %%bash
        curl rootaasdemo.web.cern.ch/rootaasdemo/SaasFee.jpg \
        > SF.jpg
```

```
  % Total    % Received % Xferd  Average Speed   Time
  Time     Time  Current
                                 Dload  Upload   Total

  Spent    Left  Speed
100  128k  100  128k    0     0  2731k      0 --:--:--
--:--:-- --:--:-- 2787k
```

… and get their output

Shell Commands

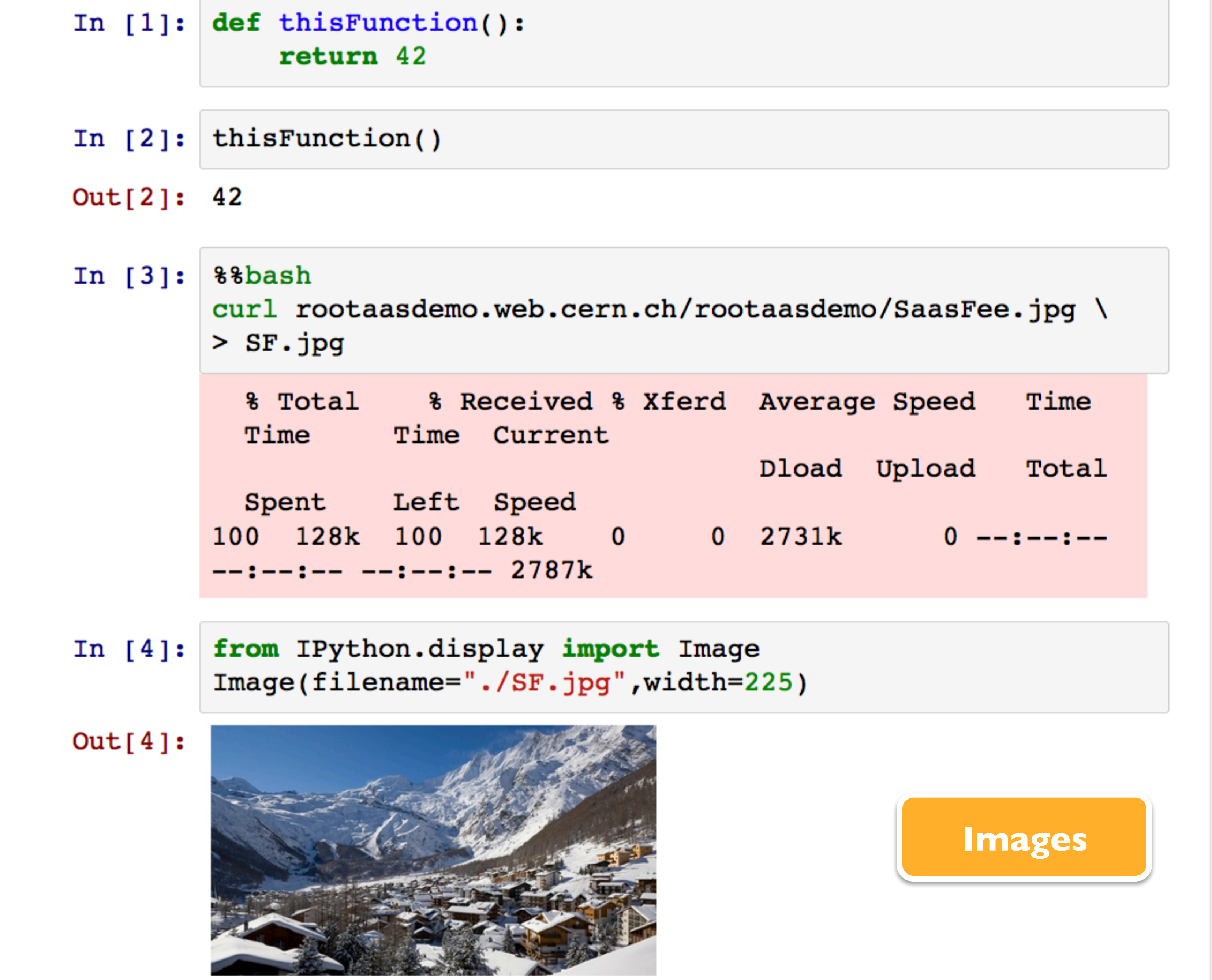


```
In [1]: def thisFunction():
            return 42
```

```
In [2]: thisFunction()

Out[2]: 42
```

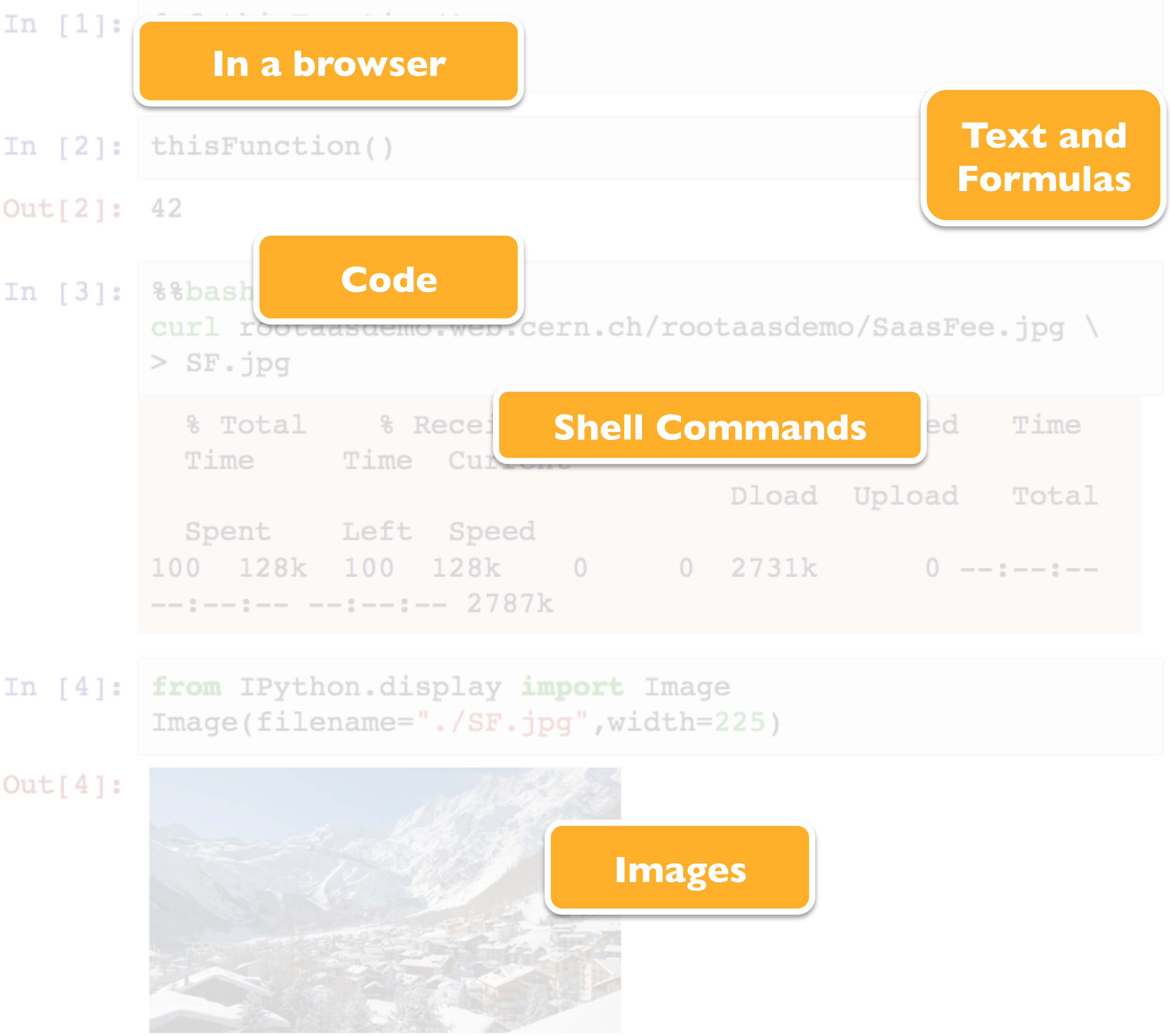
```
In [3]: %%bash
        curl rootaasdemo.web.cern.ch/rootaasdemo/SaasFee.jpg \
        > SF.jpg

          % Total    % Received % Xferd  Average Speed   Time
          Time     Time  Current
                                         Dload  Upload   Total
          Spent    Left  Speed
        100  128k  100  128k    0     0  2731k      0 --:--:--
        --:--:-- --:--:-- 2787k
```

```
In [4]: from IPython.display import Image
        Image(filename="./SF.jpg",width=225)
```



```
In [1]: def thisFunction():
            return 42

In [2]: thisFunction()

Out[2]: 42

In [3]: %%bash
        curl rootaasdemo.web.cern.ch/rootaasdemo/SaasFee.jpg \
        > SF.jpg

          % Total    % Received % Xferd  Average Speed   Time
          Time     Time  Current
                                         Dload  Upload   Total
          Spent    Left  Speed
        100  128k  100  128k    0     0  2731k      0 --:--:--
        --:--:-- --:--:-- 2787k

In [4]: from IPython.display import Image
        Image(filename="./SF.jpg",width=225)

Out[4]:
```

Images



In a browser

Text and Formulas

Code

Shell Commands

Images

```
In [1]:

In [2]: thisFunction()

Out[2]: 42

In [3]: %%bash
        curl r                  cern.ch/rootaasdemo/SaasFee.jpg \
        > SF.jpg

          % Total    % Recei                   ed   Time
          Time      Time   Cu
                                     Dload  Upload   Total
          Spent     Left   Speed
        100  128k  100  128k    0     0  2731k      0 --:--:--
        --:--:-- --:--:-- 2787k

In [4]: from IPython.display import Image
        Image(filename="./SF.jpg",width=225)

Out[4]:
```

This Talk

- The ROOTaaS project and why it is needed
- Integration of ROOT with the notebook technology
  - Programming model and usability for data analysis
- ROOTaaS within the CERN IT services' portfolio
  - Spotlight on storage
- A ROOTaaS demo

The ROOTaaS Project

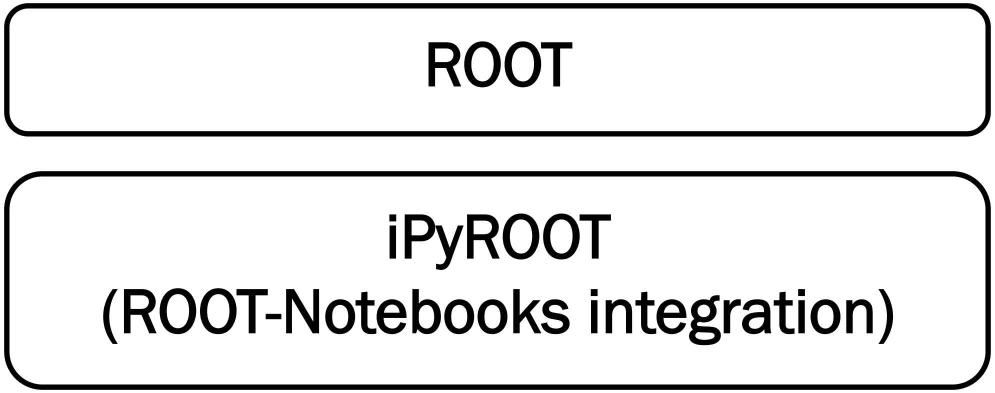
Data mining with ROOT "as a service"

Interface: Notebooks

Goals:

- Use ROOT only with a web browser
  - Platform independent ROOT based data analysis
  - Calculations, input and results "in the cloud"
- Allow easy sharing of scientific results: plots, data, code
  - Storage is crucial
- Simplify teaching of data processing and programming
- C++, Python and other languages interfaced to ROOT

Integration of ROOT with Notebooks



ROOT

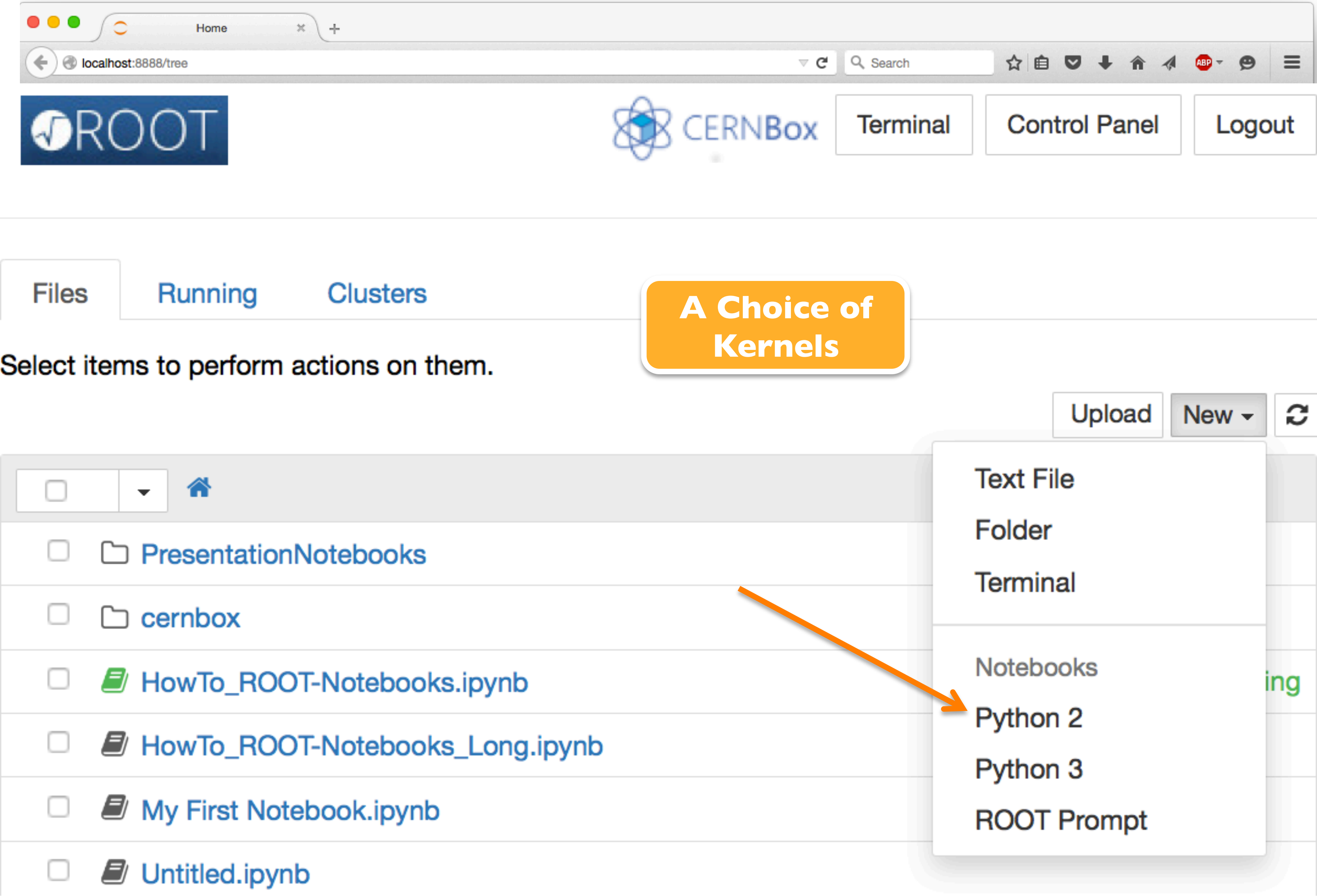
iPyROOT
(ROOT-Notebooks integration)

Integration: Main Requirements

- Code in macros/programs usable in notebooks (and vice versa)
- Provide a novel ROOT Prompt (C++) kernel
  - A notebook which is a web based ROOT prompt
- Easy access to well known ROOT and notebooks features
- Provide clear, useful examples and documentation

Requirements satisfied
Delivered in release 6.05/02

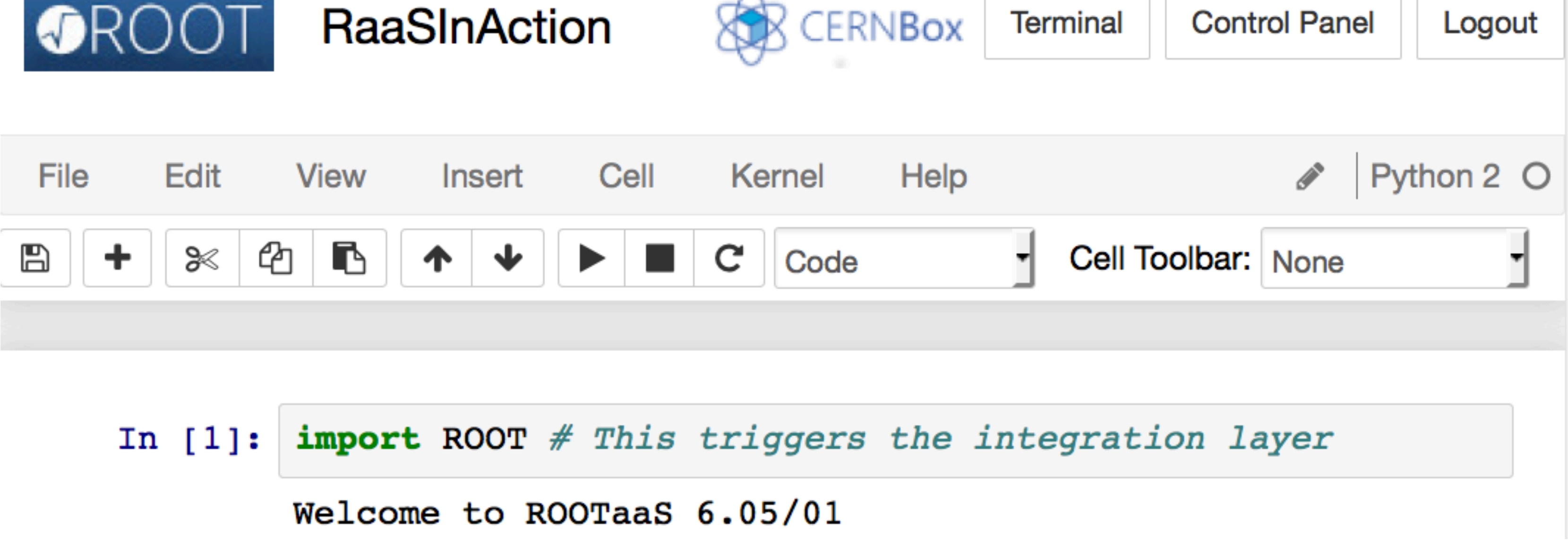
Now it's time to take a tour of the new provided functionalities!



Select items to perform actions on them.

- PresentationNotebooks
- cernbox
- HowTo_ROOT-Notebooks.ipynb
- HowTo_ROOT-Notebooks_Long.ipynb
- My First Notebook.ipynb
- Untitled.ipynb

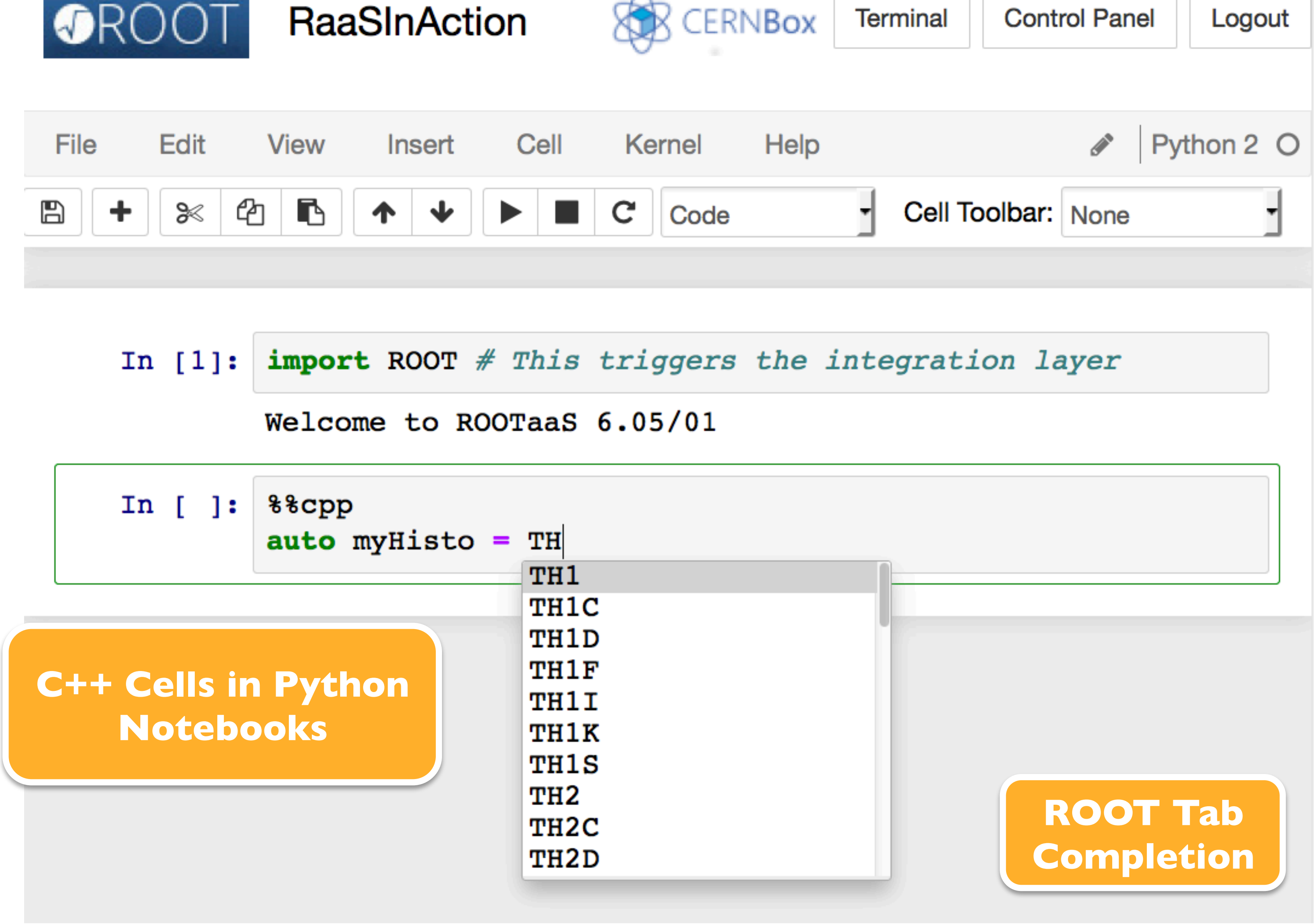
A Choice of Kernels



In [1]:

```
import ROOT # This triggers the integration layer
```

```
Welcome to ROOTaaS 6.05/01
```

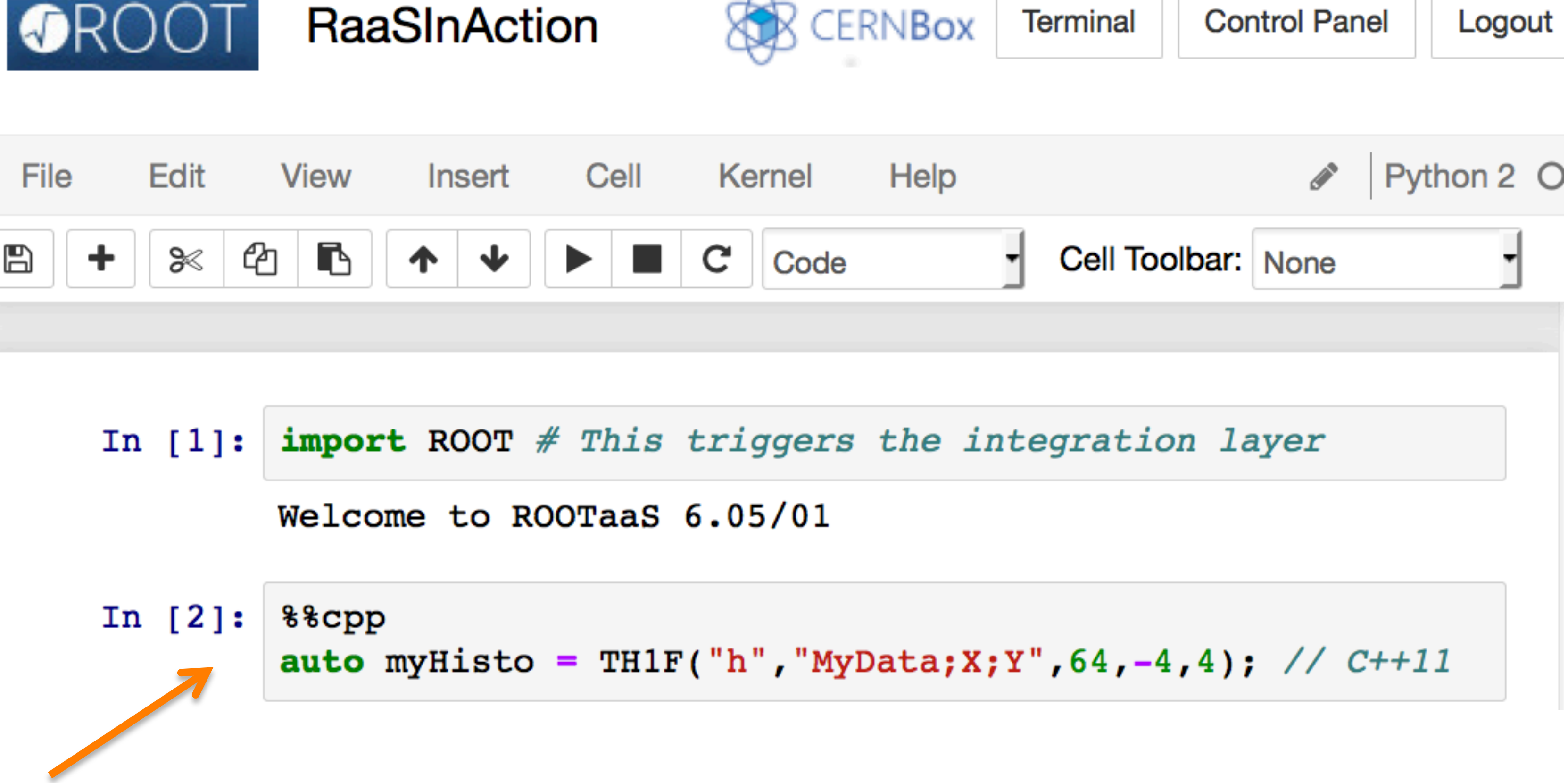


```
In [1]: import ROOT # This triggers the integration layer

        Welcome to ROOTaaS 6.05/01

In [ ]: %%cpp
        auto myHisto = TH
```

```
TH1
TH1C
TH1D
TH1F
TH1I
TH1K
TH1S
TH2
TH2C
TH2D
```

C++ Cells in Python Notebooks

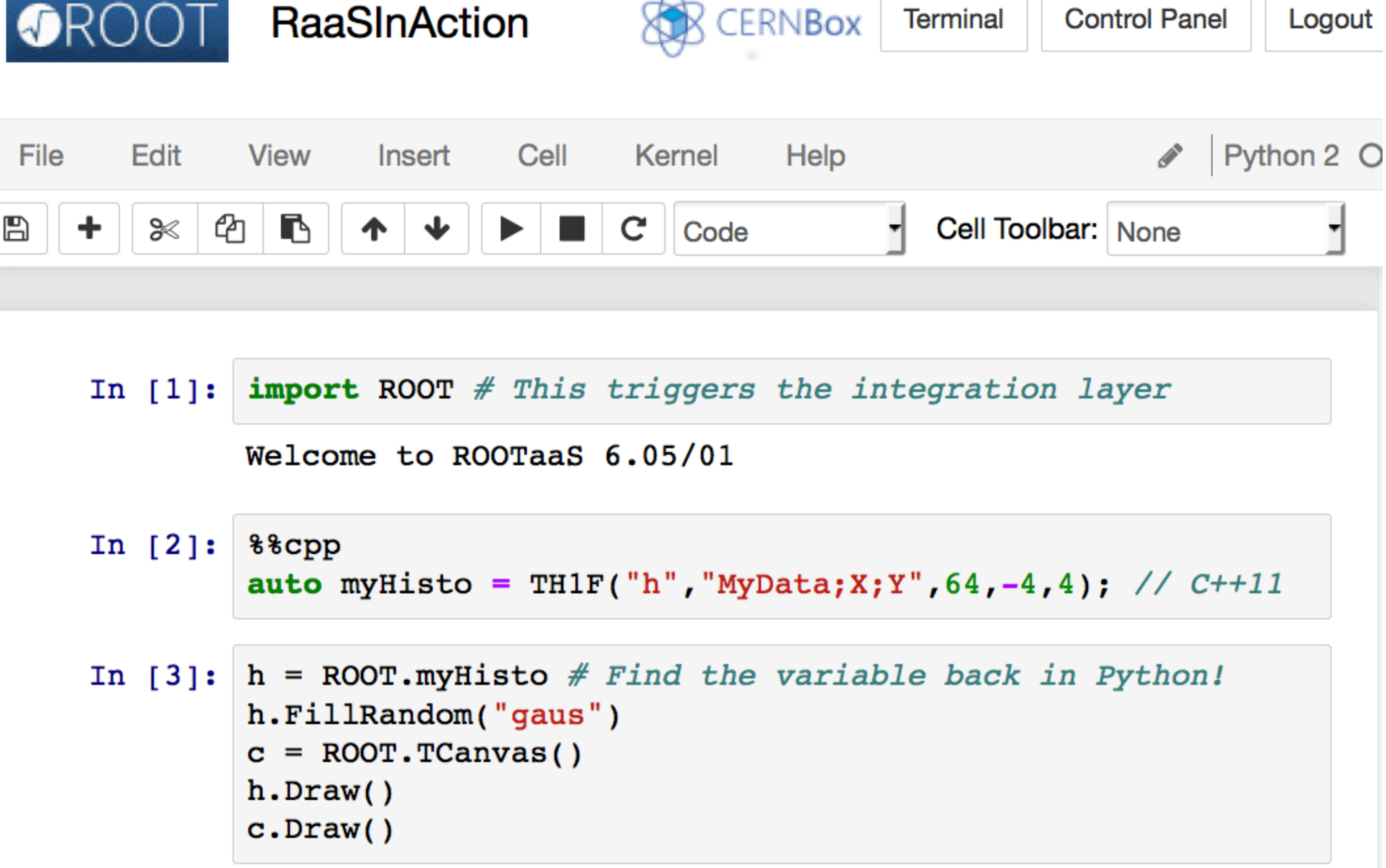
ROOT Tab Completion



```
In [1]: import ROOT # This triggers the integration layer

        Welcome to ROOTaaS 6.05/01

In [2]: %%cpp
        auto myHisto = TH1F("h","MyData;X;Y",64,-4,4); // C++11
```

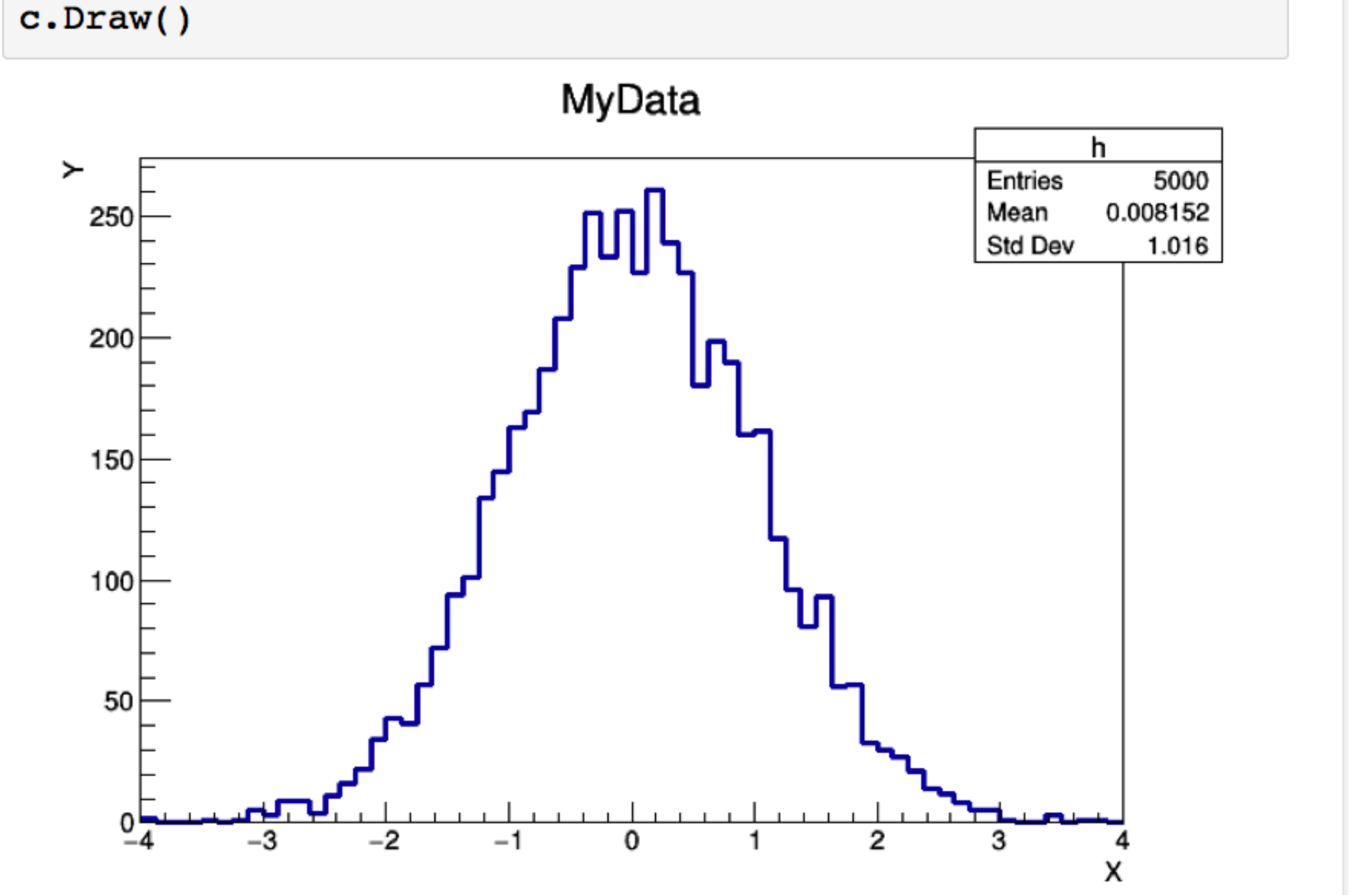
Prefer initialisation!

… I *really* wanted to show you the “auto” keyword ☺



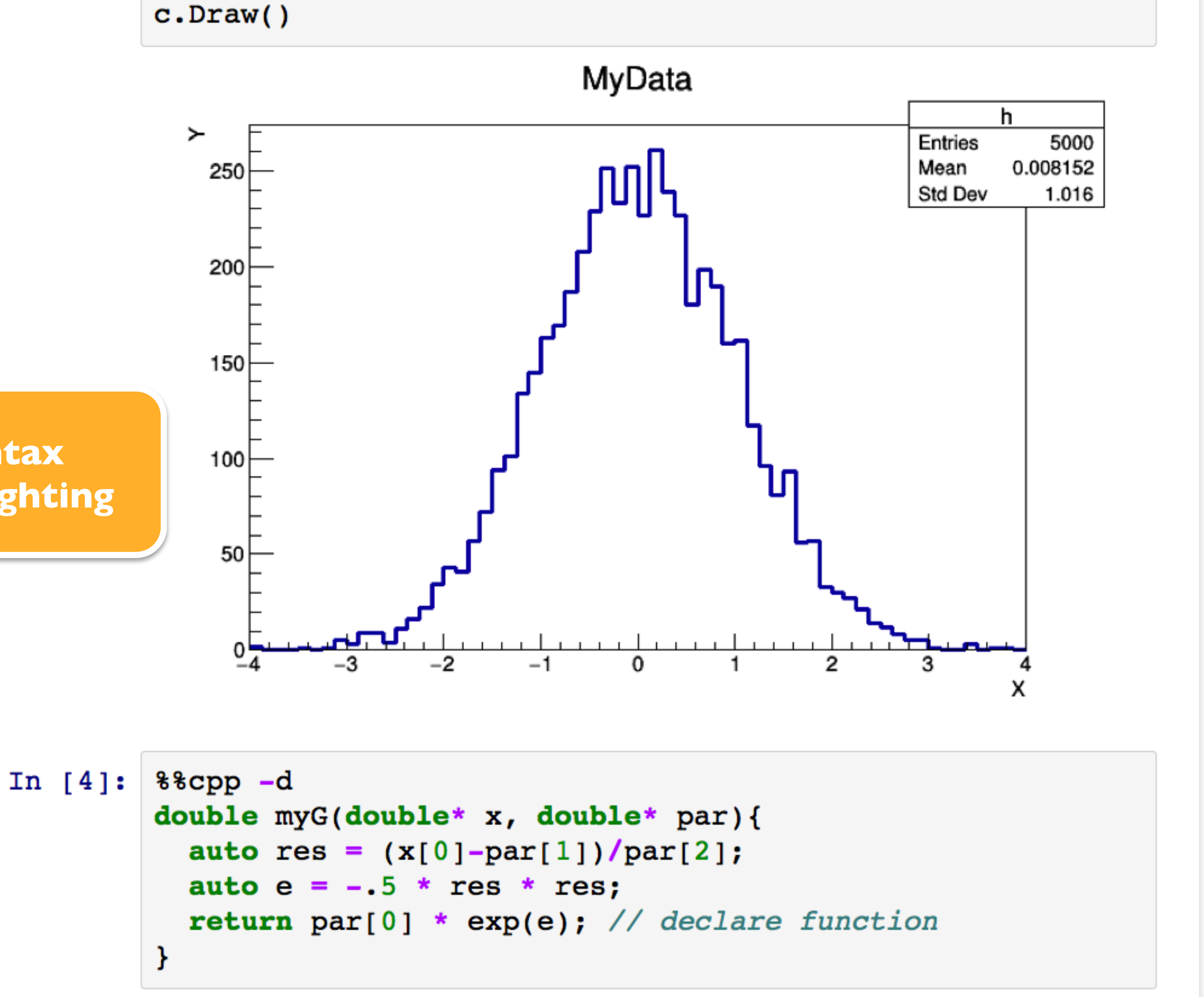
```
In [1]: import ROOT # This triggers the integration layer

Welcome to ROOTaaS 6.05/01

In [2]: %%cpp
auto myHisto = TH1F("h","MyData;X;Y",64,-4,4); // C++11

In [3]: h = ROOT.myHisto # Find the variable back in Python!
h.FillRandom("gaus")
c = ROOT.TCanvas()
h.Draw()
c.Draw()
```

C++-Python Interoperability



```
c.Draw()
```

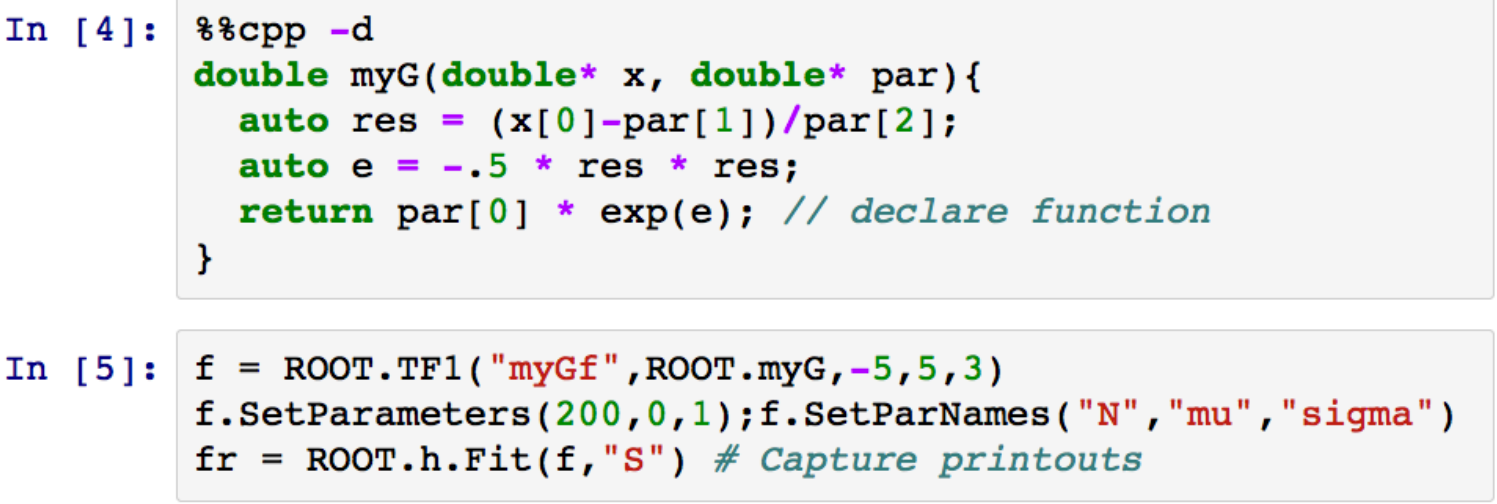
Seamless display of graphics



```
c.Draw()
```

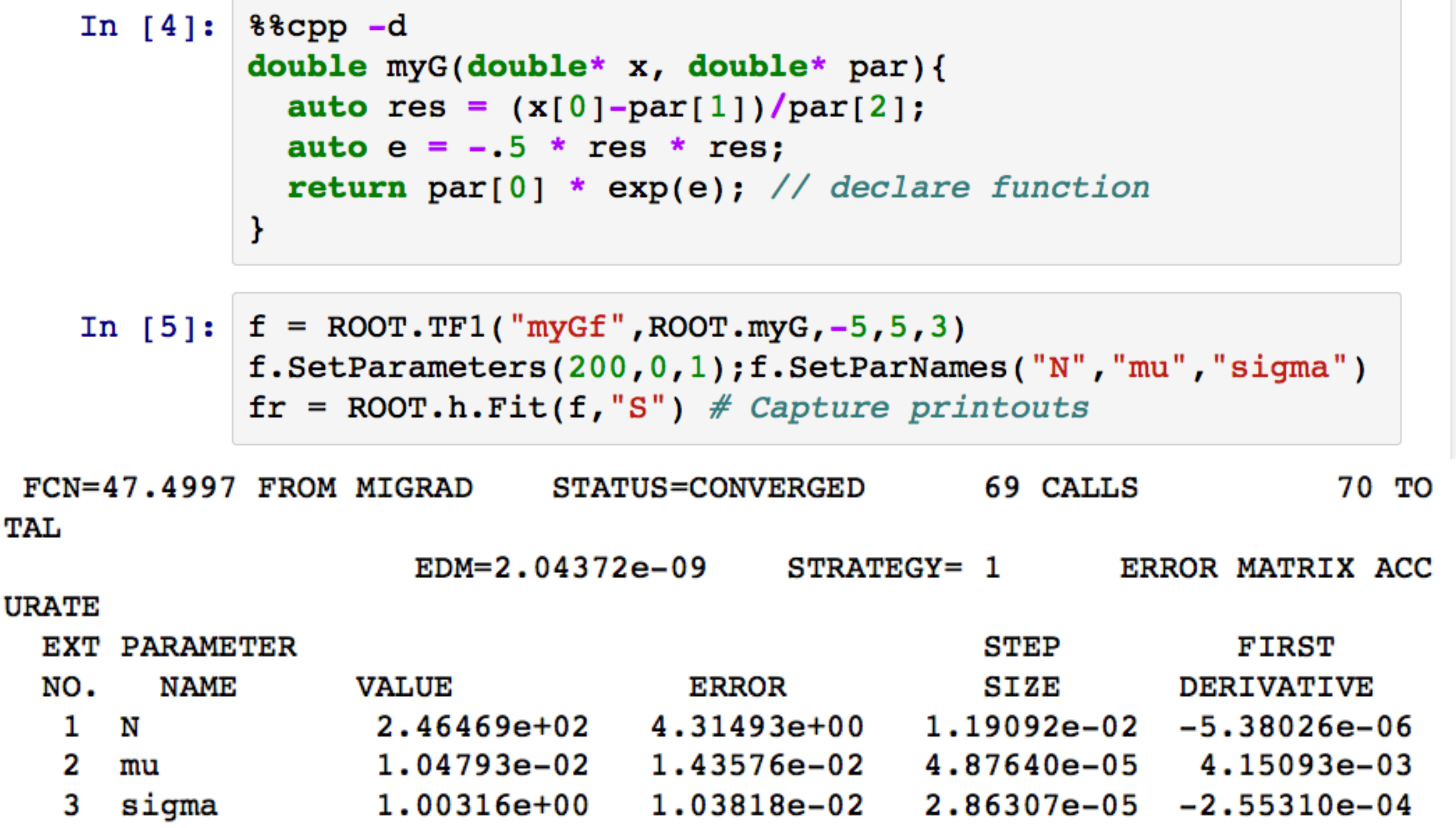
```
%%cpp -d
double myG(double* x, double* par){
  auto res = (x[0]-par[1])/par[2];
  auto e = -.5 * res * res;
  return par[0] * exp(e); // declare function
}
```

Syntax Highlighting



```
In [4]: %%cpp -d
        double myG(double* x, double* par){
          auto res = (x[0]-par[1])/par[2];
          auto e = -.5 * res * res;
          return par[0] * exp(e); // declare function
        }
```

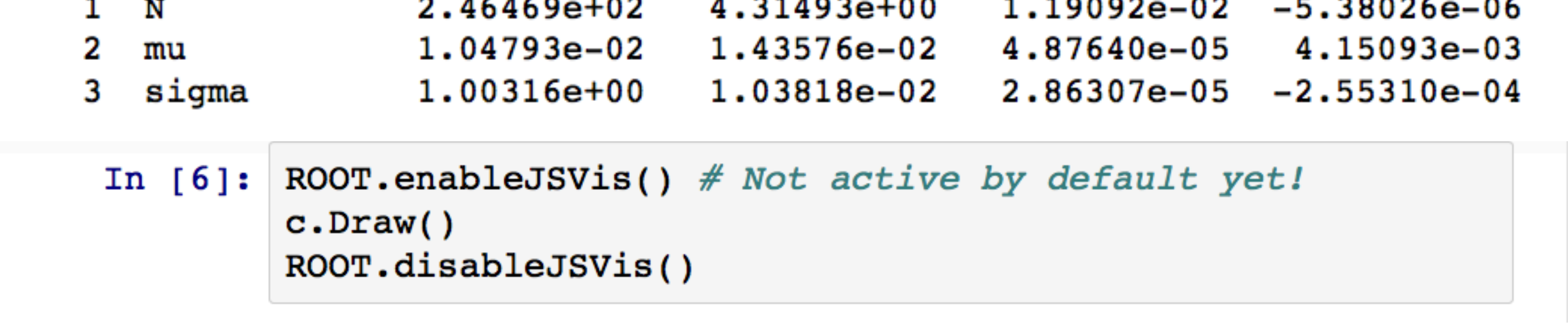
```
In [5]: f = ROOT.TF1("myGf",ROOT.myG,-5,5,3)
        f.SetParameters(200,0,1);f.SetParNames("N","mu","sigma")
        fr = ROOT.h.Fit(f,"S") # Capture printouts
```



In [4]:
```
%%cpp -d
double myG(double* x, double* par){
  auto res = (x[0]-par[1])/par[2];
  auto e = -.5 * res * res;
  return par[0] * exp(e); // declare function
}
```

In [5]:
```
f = ROOT.TF1("myGf",ROOT.myG,-5,5,3)
f.SetParameters(200,0,1);f.SetParNames("N","mu","sigma")
fr = ROOT.h.Fit(f,"S") # Capture printouts
```

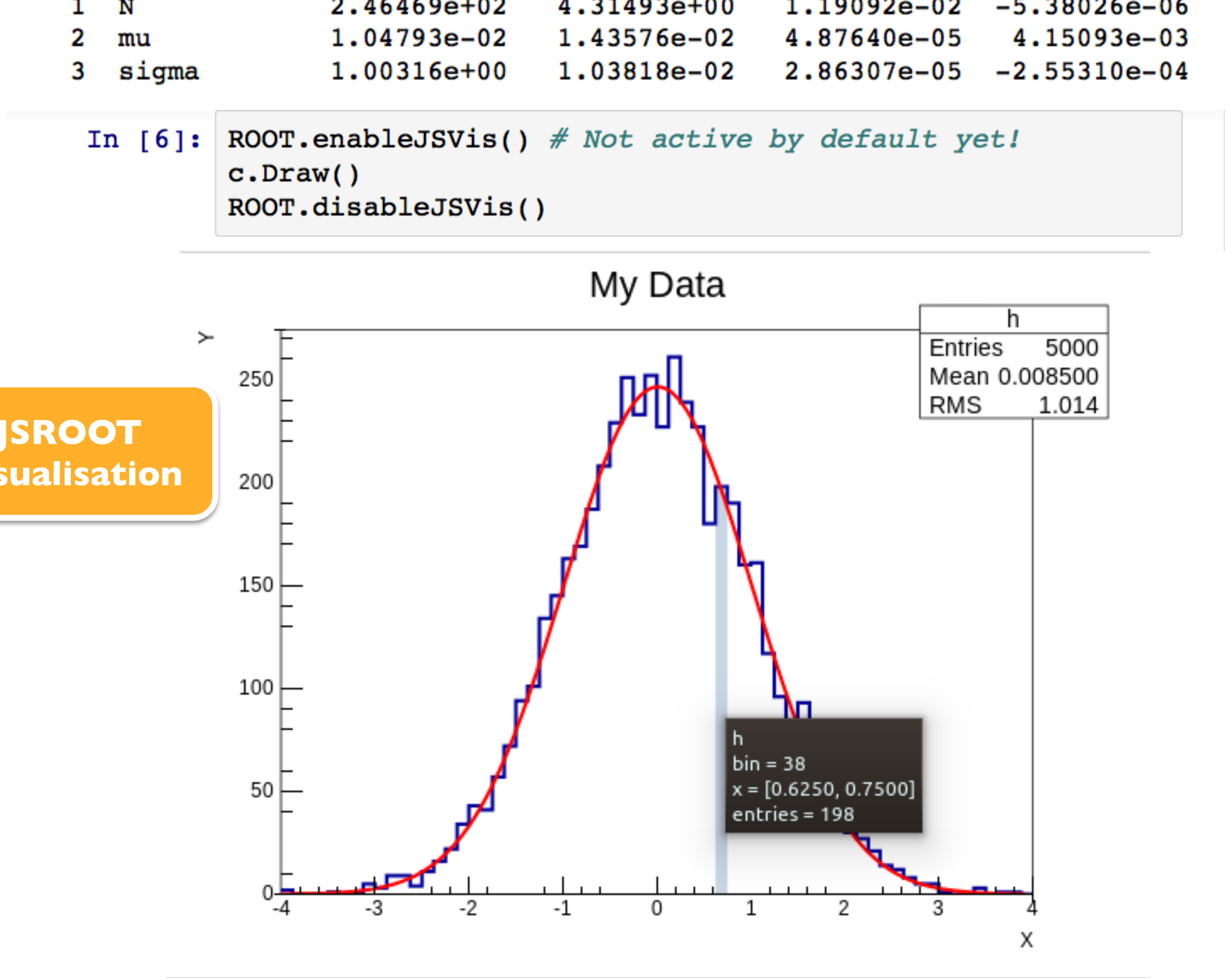
```
 FCN=47.4997 FROM MIGRAD    STATUS=CONVERGED      69 CALLS          70 TO
TAL
                     EDM=2.04372e-09    STRATEGY= 1      ERROR MATRIX ACC
URATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  N            2.46469e+02   4.31493e+00   1.19092e-02  -5.38026e-06
   2  mu           1.04793e-02   1.43576e-02   4.87640e-05   4.15093e-03
   3  sigma        1.00316e+00   1.03818e-02   2.86307e-05  -2.55310e-04
```



```
  1  N              2.46469e+02   4.31493e+00   1.19092e-02  -5.38026e-06
  2  mu             1.04793e-02   1.43576e-02   4.87640e-05   4.15093e-03
  3  sigma          1.00316e+00   1.03818e-02   2.86307e-05  -2.55310e-04
```

In [6]:

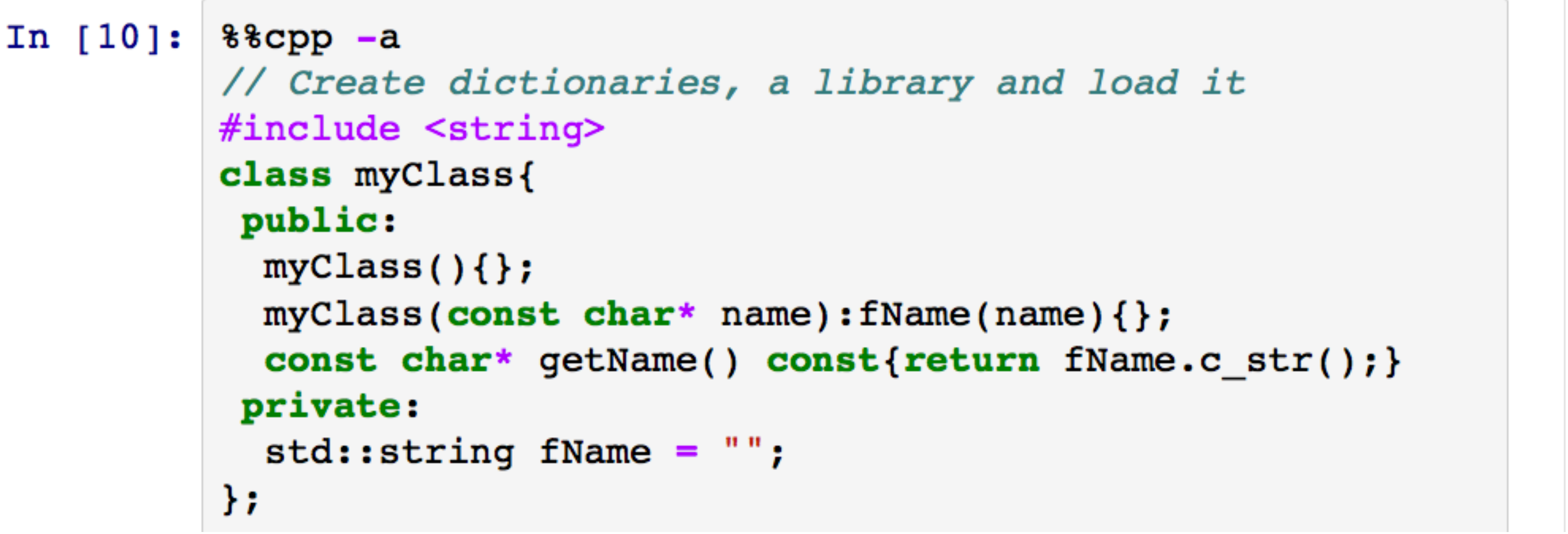
```
ROOT.enableJSVis() # Not active by default yet!
c.Draw()
ROOT.disableJSVis()
```



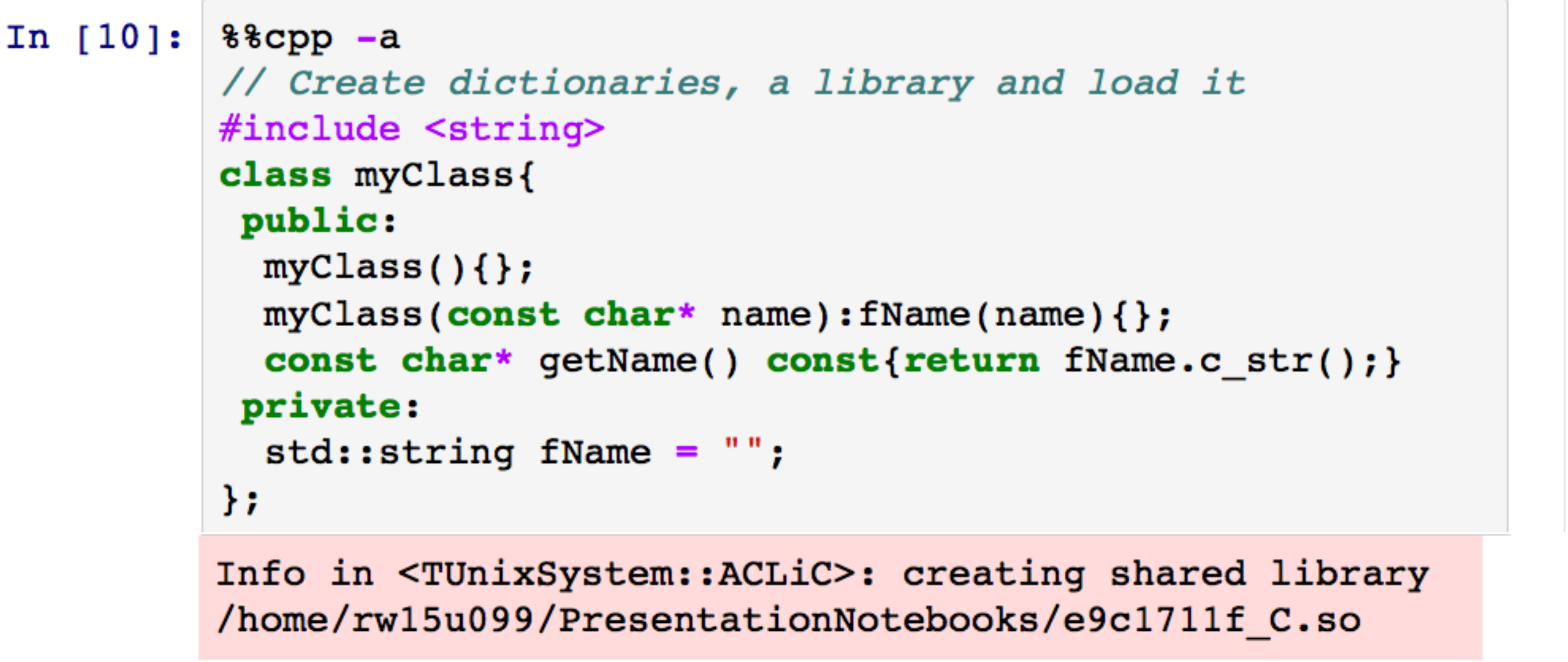
```
  1  N            2.46469e+02   4.31493e+00   1.19092e-02  -5.38026e-06
  2  mu           1.04793e-02   1.43576e-02   4.87640e-05   4.15093e-03
  3  sigma        1.00316e+00   1.03818e-02   2.86307e-05  -2.55310e-04
```

In [6]:
```
ROOT.enableJSVis() # Not active by default yet!
c.Draw()
ROOT.disableJSVis()
```

JSROOT Visualisation



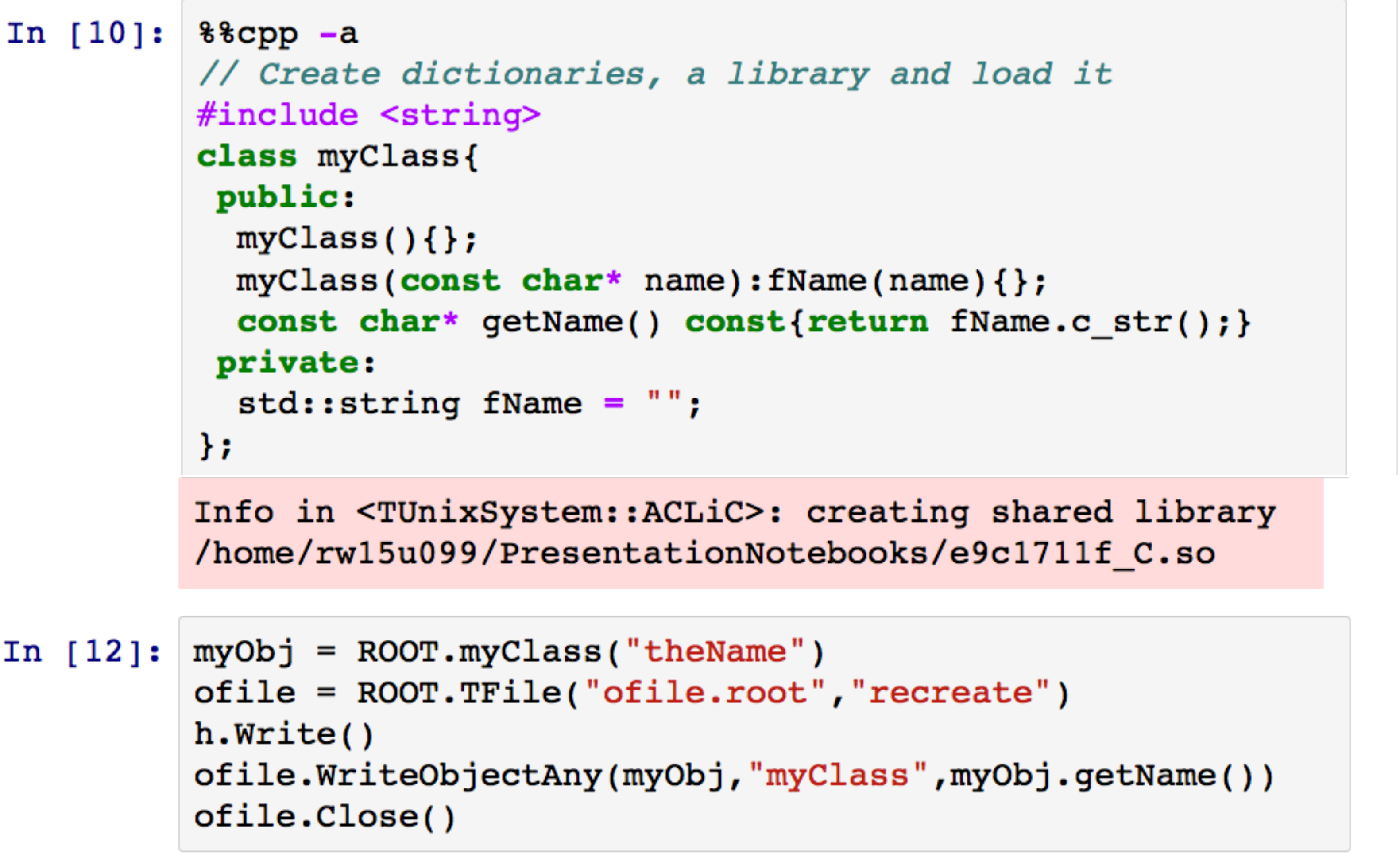
```
%%cpp -a
// Create dictionaries, a library and load it
#include <string>
class myClass{
 public:
  myClass(){};
  myClass(const char* name):fName(name){};
  const char* getName() const{return fName.c_str();}
 private:
  std::string fName = "";
};
```



In [10]:

```
%%cpp -a
// Create dictionaries, a library and load it
#include <string>
class myClass{
 public:
  myClass(){};
  myClass(const char* name):fName(name){};
  const char* getName() const{return fName.c_str();}
 private:
  std::string fName = "";
};
```

```
Info in <TUnixSystem::ACLiC>: creating shared library
/home/rw15u099/PresentationNotebooks/e9c1711f_C.so
```



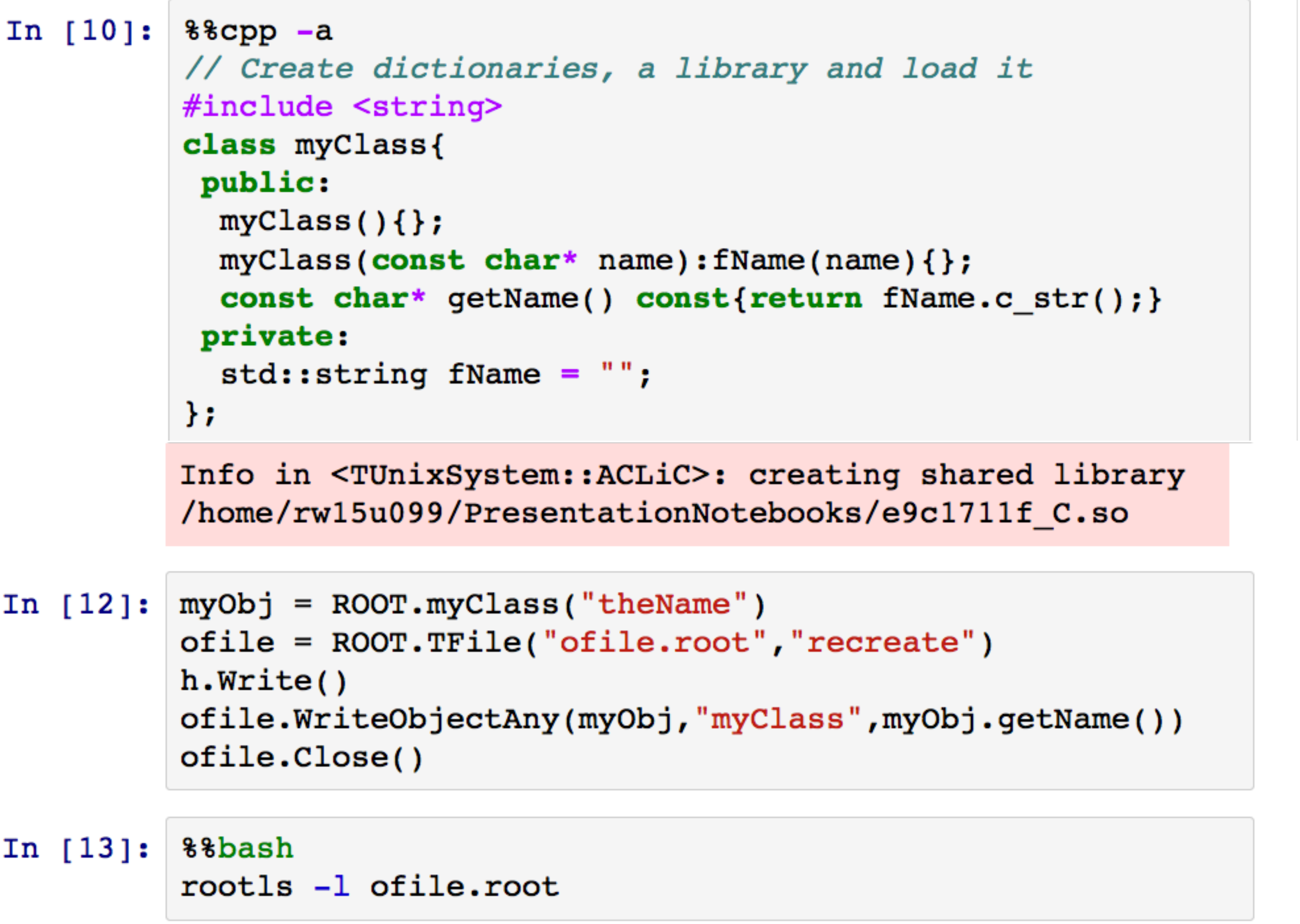
In [10]:

```
%%cpp -a
// Create dictionaries, a library and load it
#include <string>
class myClass{
 public:
  myClass(){};
  myClass(const char* name):fName(name){};
  const char* getName() const{return fName.c_str();}
 private:
  std::string fName = "";
};
```

```
Info in <TUnixSystem::ACLiC>: creating shared library
/home/rw15u099/PresentationNotebooks/e9c1711f_C.so
```

In [12]:

```
myObj = ROOT.myClass("theName")
ofile = ROOT.TFile("ofile.root","recreate")
h.Write()
ofile.WriteObjectAny(myObj,"myClass",myObj.getName())
ofile.Close()
```



In [10]:

```
%%cpp -a
// Create dictionaries, a library and load it
#include <string>
class myClass{
 public:
  myClass(){};
  myClass(const char* name):fName(name){};
  const char* getName() const{return fName.c_str();}
 private:
  std::string fName = "";
};
```

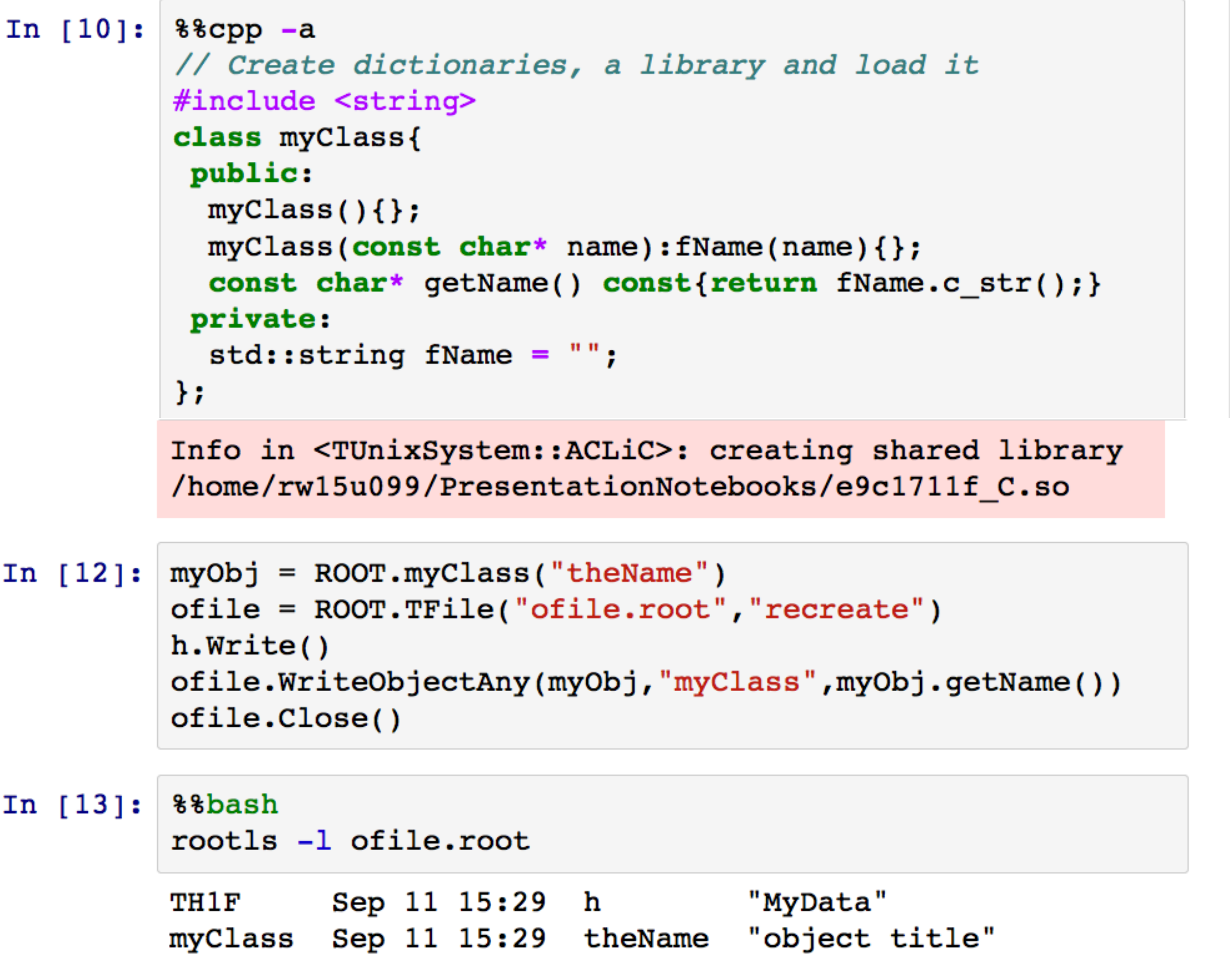
```
Info in <TUnixSystem::ACLiC>: creating shared library
/home/rw15u099/PresentationNotebooks/e9c1711f_C.so
```

In [12]:

```
myObj = ROOT.myClass("theName")
ofile = ROOT.TFile("ofile.root","recreate")
h.Write()
ofile.WriteObjectAny(myObj,"myClass",myObj.getName())
ofile.Close()
```

In [13]:

```
%%bash
rootls -l ofile.root
```



```
In [10]: %%cpp -a
         // Create dictionaries, a library and load it
         #include <string>
         class myClass{
          public:
           myClass(){};
           myClass(const char* name):fName(name){};
           const char* getName() const{return fName.c_str();}
          private:
           std::string fName = "";
         };

         Info in <TUnixSystem::ACLiC>: creating shared library
         /home/rw15u099/PresentationNotebooks/e9c1711f_C.so
```

```
In [12]: myObj = ROOT.myClass("theName")
         ofile = ROOT.TFile("ofile.root","recreate")
         h.Write()
         ofile.WriteObjectAny(myObj,"myClass",myObj.getName())
         ofile.Close()
```

```
In [13]: %%bash
         rootls -l ofile.root

         TH1F     Sep 11 15:29  h        "MyData"
         myClass  Sep 11 15:29  theName  "object title"
```

All the power of ROOT: Dictionaries, I/O, runtime loading of libraries

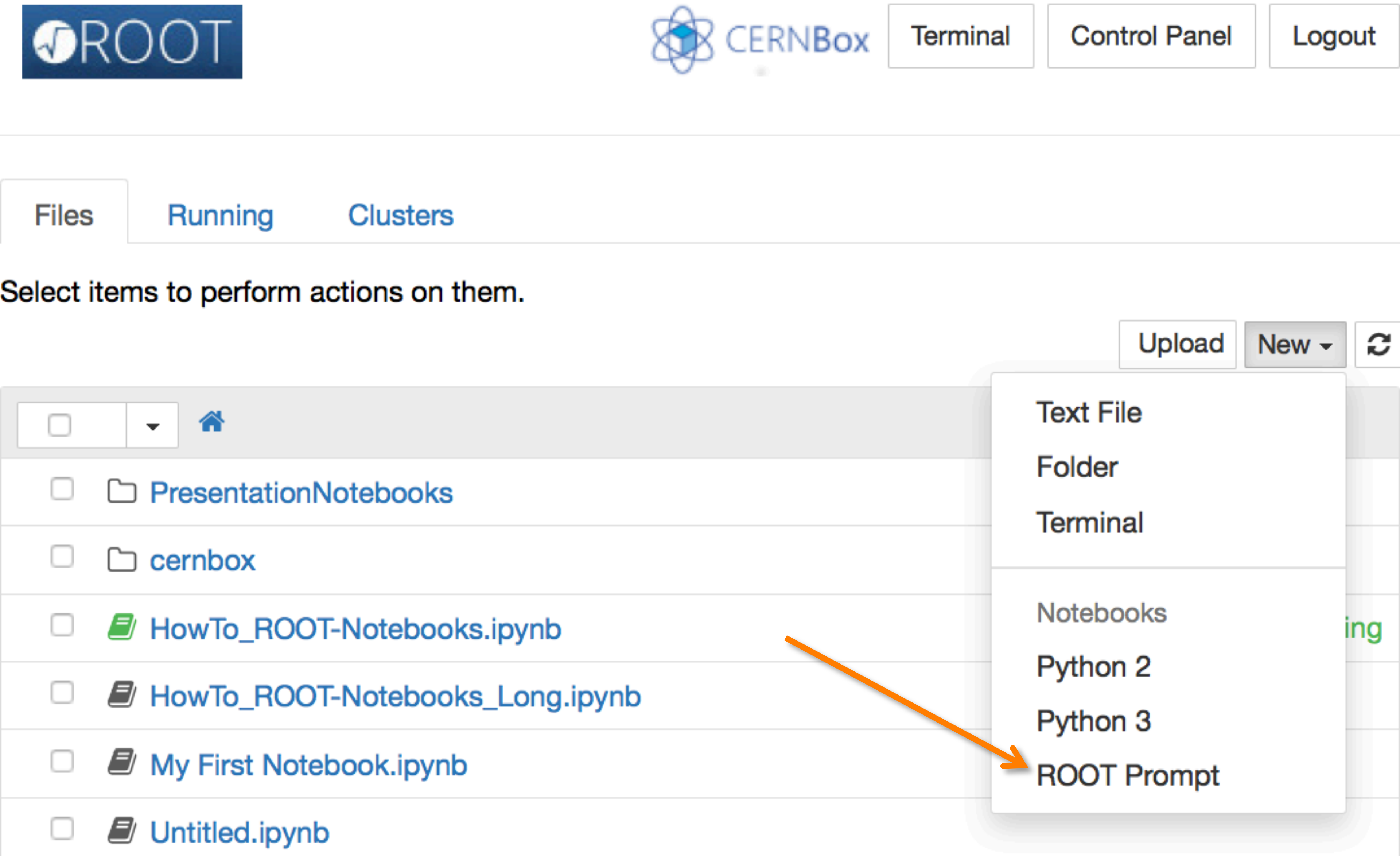
Take-away Message

"import ROOT" turns on all notebook goodies

- Tab-completion
- C++ cells, ACLiC
- Display of graphics
- Syntax highlighting

All the power of ROOT and the ROOT Python bindings, PyROOT, are there

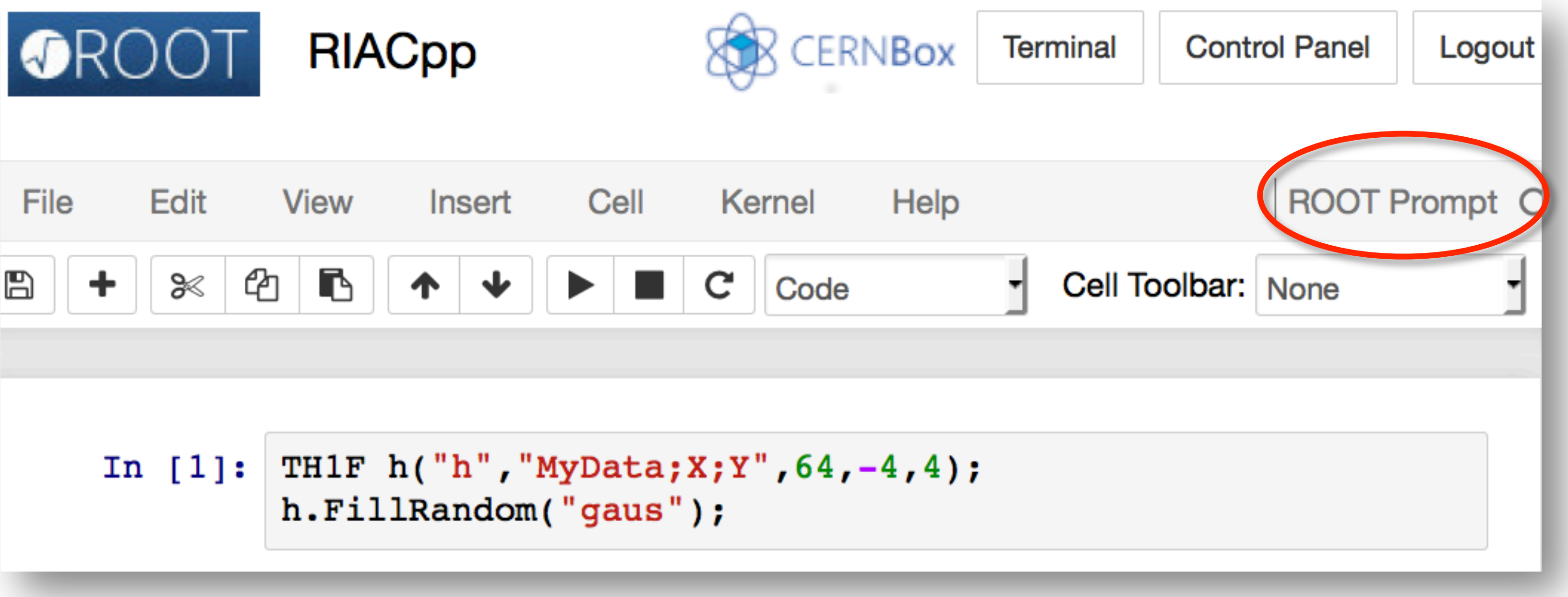
Like Before, but better



Select items to perform actions on them.

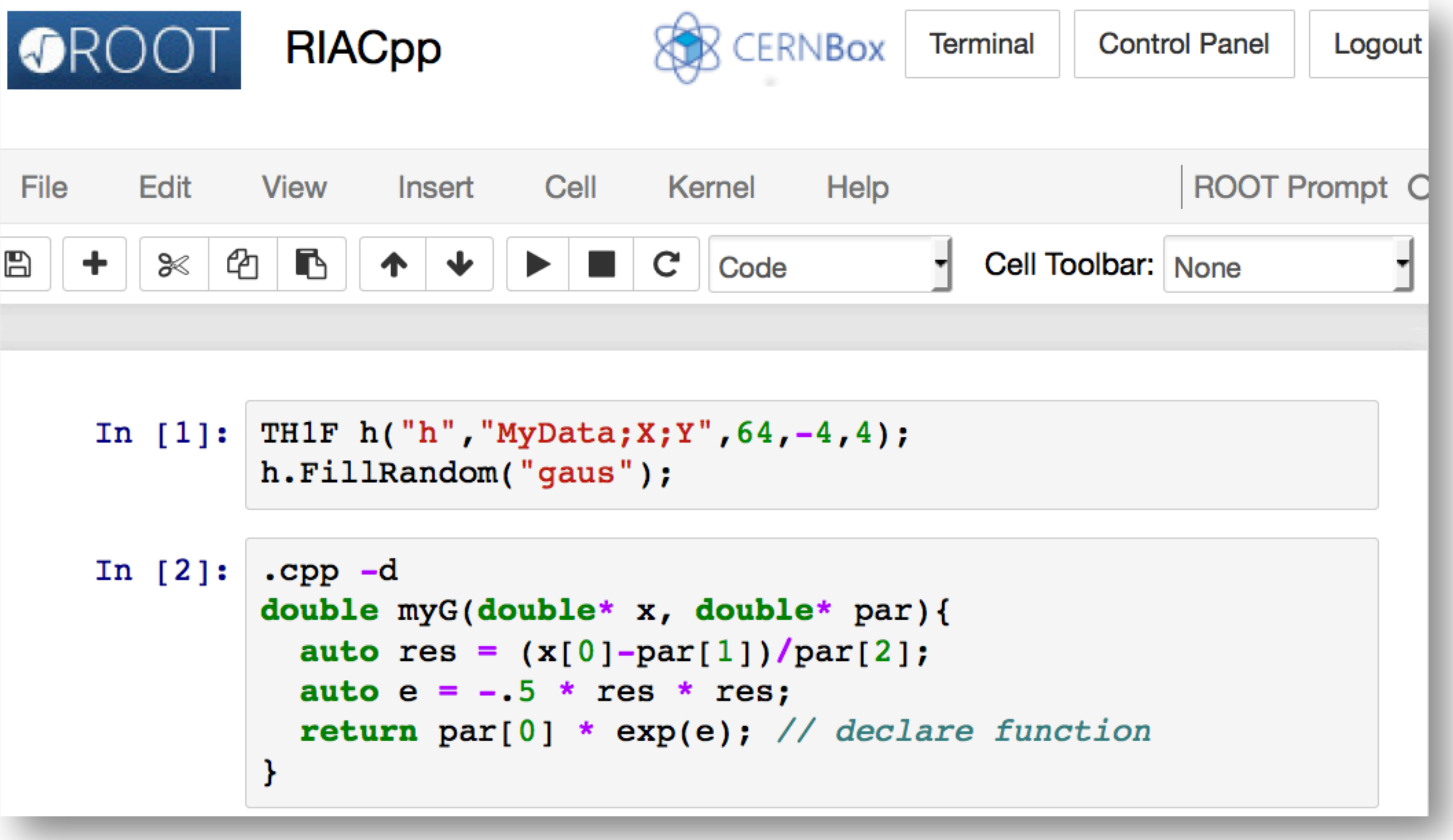
- PresentationNotebooks
- cernbox
- HowTo_ROOT-Notebooks.ipynb
- HowTo_ROOT-Notebooks_Long.ipynb
- My First Notebook.ipynb
- Untitled.ipynb

A C++ Notebook



```
TH1F h("h","MyData;X;Y",64,-4,4);
h.FillRandom("gaus");
```

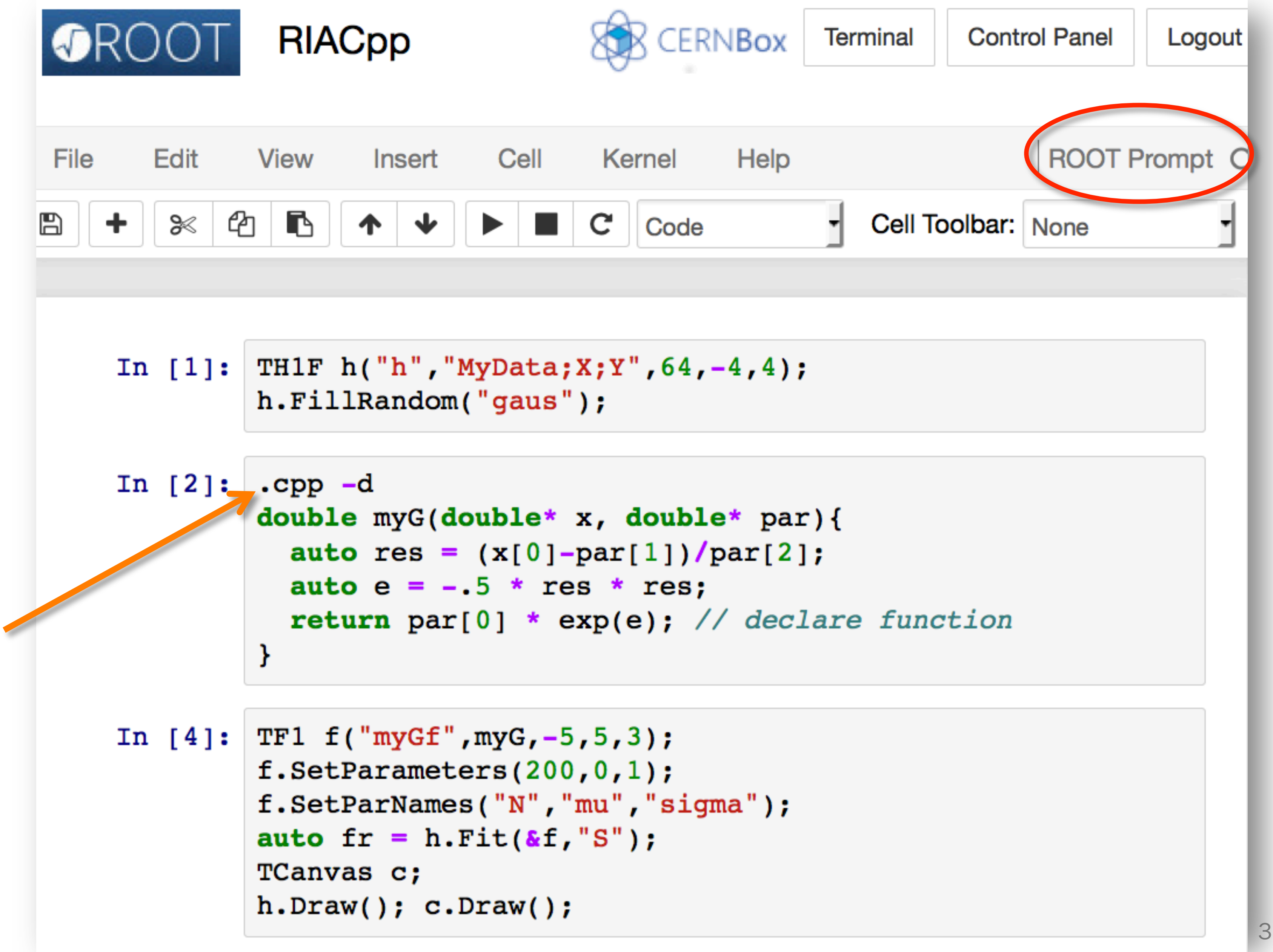
A C++ Notebook



In [1]:
```
TH1F h("h","MyData;X;Y",64,-4,4);
h.FillRandom("gaus");
```

In [2]:
```
.cpp -d
double myG(double* x, double* par){
  auto res = (x[0]-par[1])/par[2];
  auto e = -.5 * res * res;
  return par[0] * exp(e); // declare function
}
```

A C++ Notebook

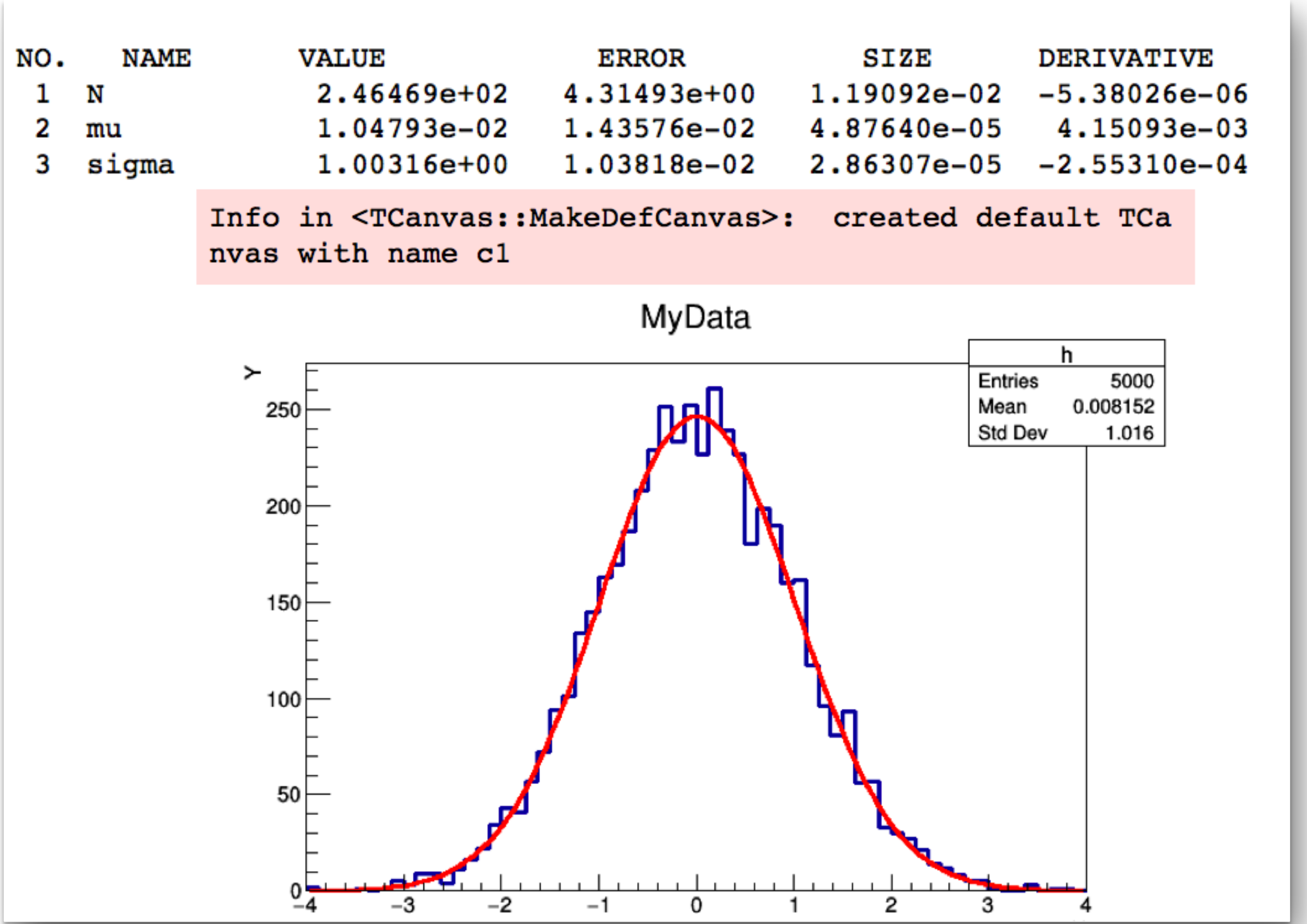


In [1]:
```
TH1F h("h","MyData;X;Y",64,-4,4);
h.FillRandom("gaus");
```

In [2]:
```
.cpp -d
double myG(double* x, double* par){
  auto res = (x[0]-par[1])/par[2];
  auto e = -.5 * res * res;
  return par[0] * exp(e); // declare function
}
```

In [4]:
```
TF1 f("myGf",myG,-5,5,3);
f.SetParameters(200,0,1);
f.SetParNames("N","mu","sigma");
auto fr = h.Fit(&f,"S");
TCanvas c;
h.Draw(); c.Draw();
```

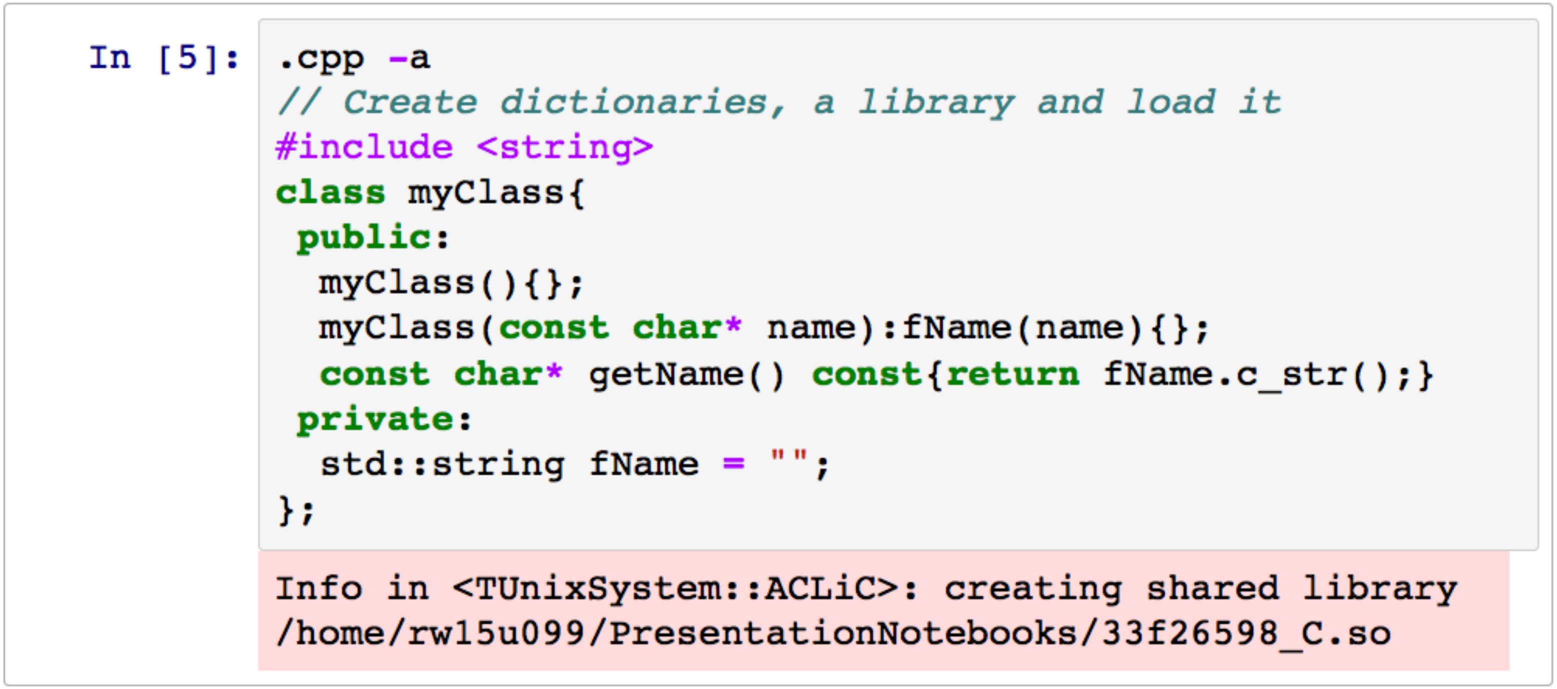
A C++ Notebook



```
NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
  1  N           2.46469e+02   4.31493e+00   1.19092e-02  -5.38026e-06
  2  mu          1.04793e-02   1.43576e-02   4.87640e-05   4.15093e-03
  3  sigma       1.00316e+00   1.03818e-02   2.86307e-05  -2.55310e-04
```

```
Info in <TCanvas::MakeDefCanvas>:  created default TCa
nvas with name c1
```

A C++ Notebook



In [5]:

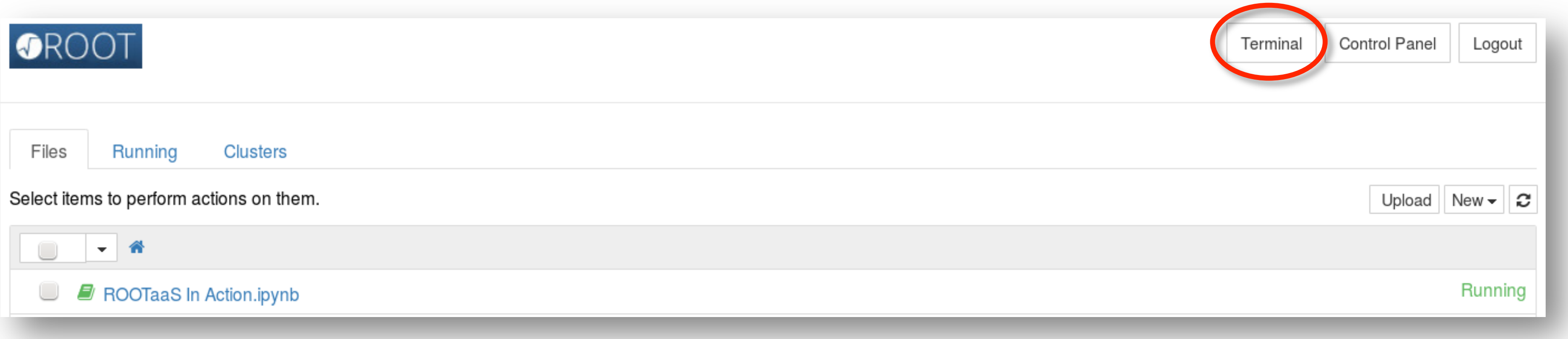
```
.cpp -a
// Create dictionaries, a library and load it
#include <string>
class myClass{
 public:
  myClass(){};
  myClass(const char* name):fName(name){};
  const char* getName() const{return fName.c_str();}
 private:
  std::string fName = "";
};
```

```
Info in <TUnixSystem::ACLiC>: creating shared library
/home/rw15u099/PresentationNotebooks/33f26598_C.so
```

A C++ Notebook there out of the box,
ROOT libraries available

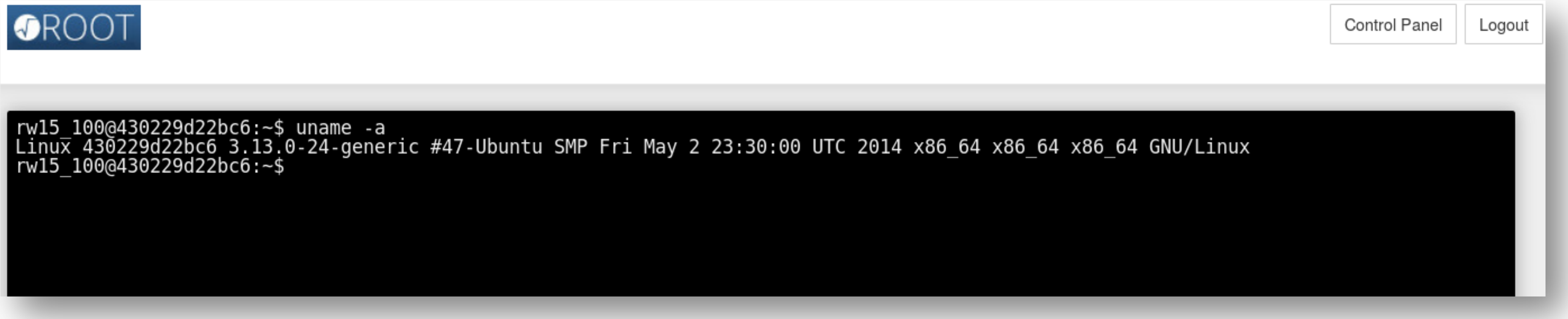
Going the Extra Mile

Make terminal available with one click!



Select items to perform actions on them.

ROOTaaS In Action.ipynb Running



```
rw15_100@430229d22bc6:~$ uname -a
Linux 430229d22bc6 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
rw15_100@430229d22bc6:~$
```

Documentation, Documentation, Documentation

Examples (15 already) from the *new* ROOT Tutorials can be found at https://root.cern.ch/code-examples#notebooks both in Python and C++ (and mixed!)

“Howto”s

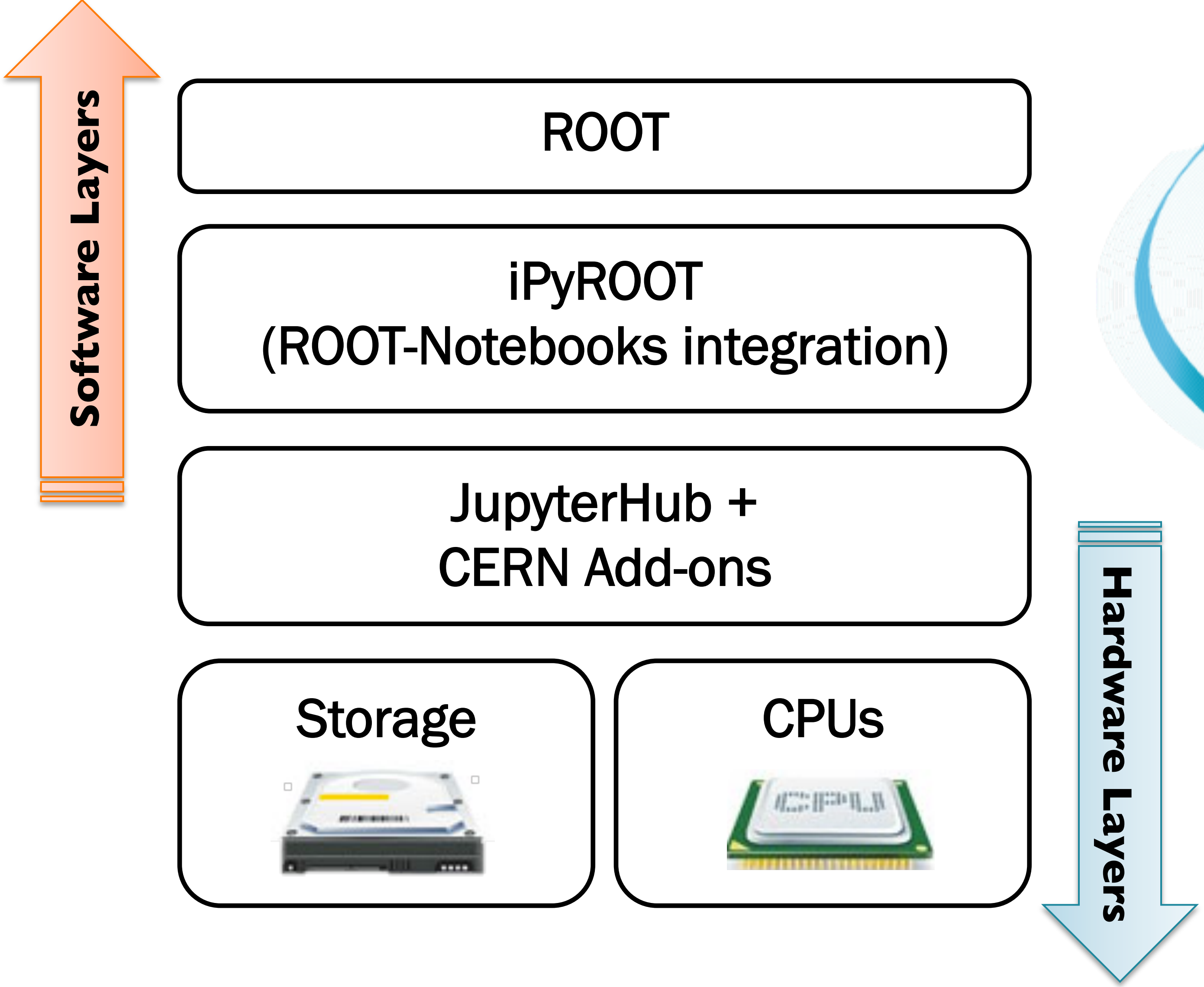
“How To use ROOT in a Notebook” instructions and

“How To activate a *ROOT Prompt* kernel in Your IPython Notebook”

https://root.cern.ch/howtos#Language%20Bindings

Notebook technology also adopted for writing the most “pragmatic” HowTos: https://root.cern.ch/howtos

See backup for more information.



Software Layers

ROOT

iPyROOT
(ROOT-Notebooks integration)

JupyterHub +
CERN Add-ons

Storage

CPUs

Hardware Layers

ROOTaaS and
CERN services’ Portfolio

Glossary

EOS

disk-based low latency storage infrastructure for physics users. Main target: physics data analysis.

CERNBOX

functionality analogous to Dropbox™. Synchronisation capabilities between user machines and central repository. Data stored on EOS.

Indico

Manage complex conferences, workshops and meetings.

CVMFS

HTTP based network FS, optimized to deliver experiment software

Files aggressively cached and downloaded on demand.

Integration With CERN's Ecosystem

- Centrally provide ROOT as a Service
- Authentication with CERN credentials
- Connect to virtual machines in the OpenStack cloud
- Access storage: CERNBox, EOS, CVMFS
  - All data and all software potentially available!
- Synergy with document Sharing (e.g. CERN Indico)
  - Notebook visualiser available in the next Indico release
  - First spinoff of ROOTaaS: C++ highlighter integrated
  - Thanks to *P. Ferreira* (IT-CIS-AVC) for the fruitful collaboration

Collaborating with partners in the CERN IT department, for example Data Storage Services group (IT-DSS)

Potential "Daily Use-Case"

- Launch jobs on the batch farm
- Access notebook on a VM in the OpenStack instance
- Inspect produced data via CERNBox/EOS from the notebook
- Create plots and output data
- Share, access plots (and output data!) on the web with CERNBox web interface
- Security and confidentiality guaranteed by the usual CERN standards

Added value: remote users often cannot open graphical connections over ssh to CERN (latency): Problem automatically solved in the above workflow.

E.g.

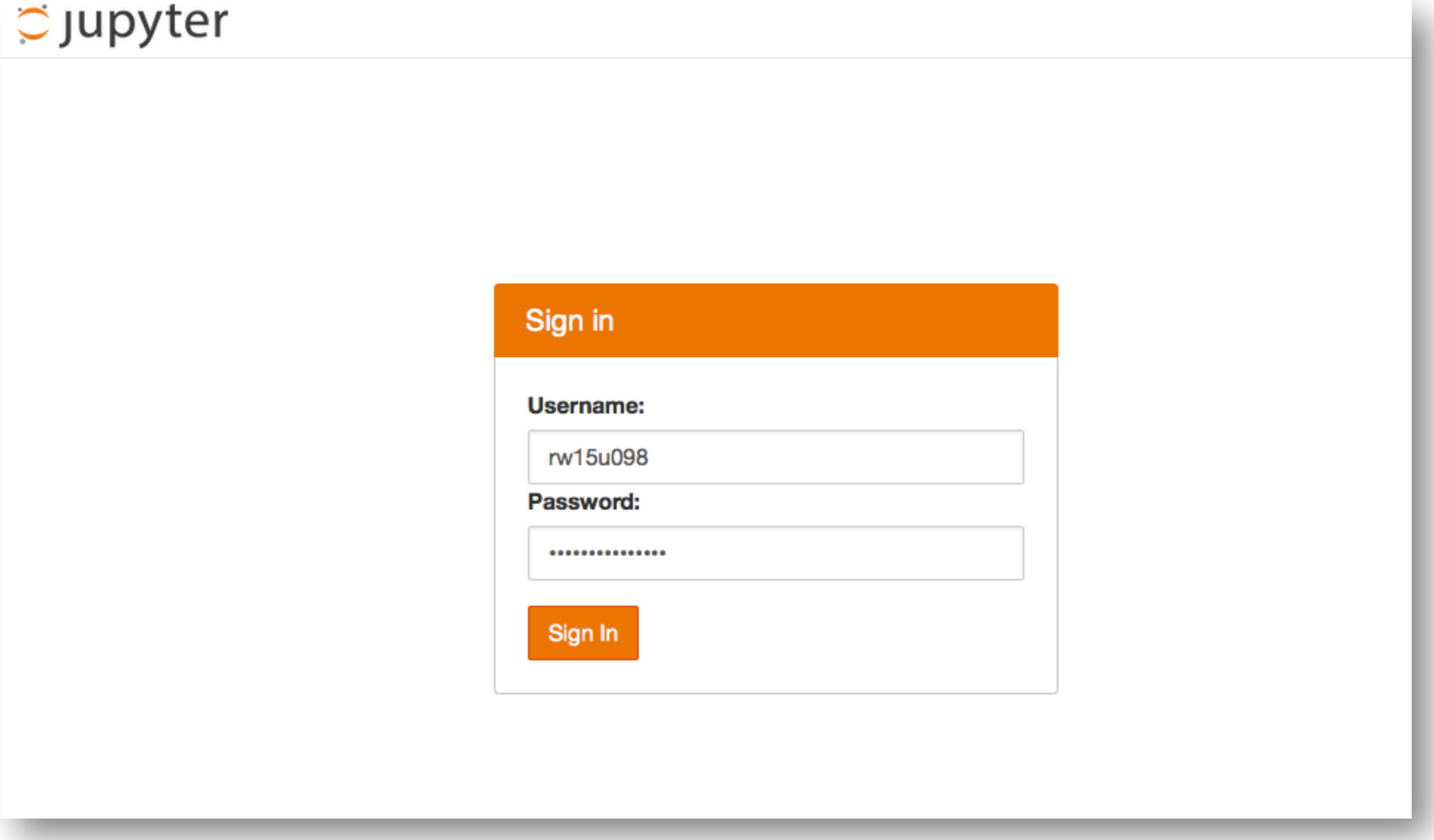
Live Demo

Time to go back see this workflow in action!

We will:

- Create a simple plot and a ROOT file with ROOTaaS
- Share it with CERNBox

Live Demo



Sign in

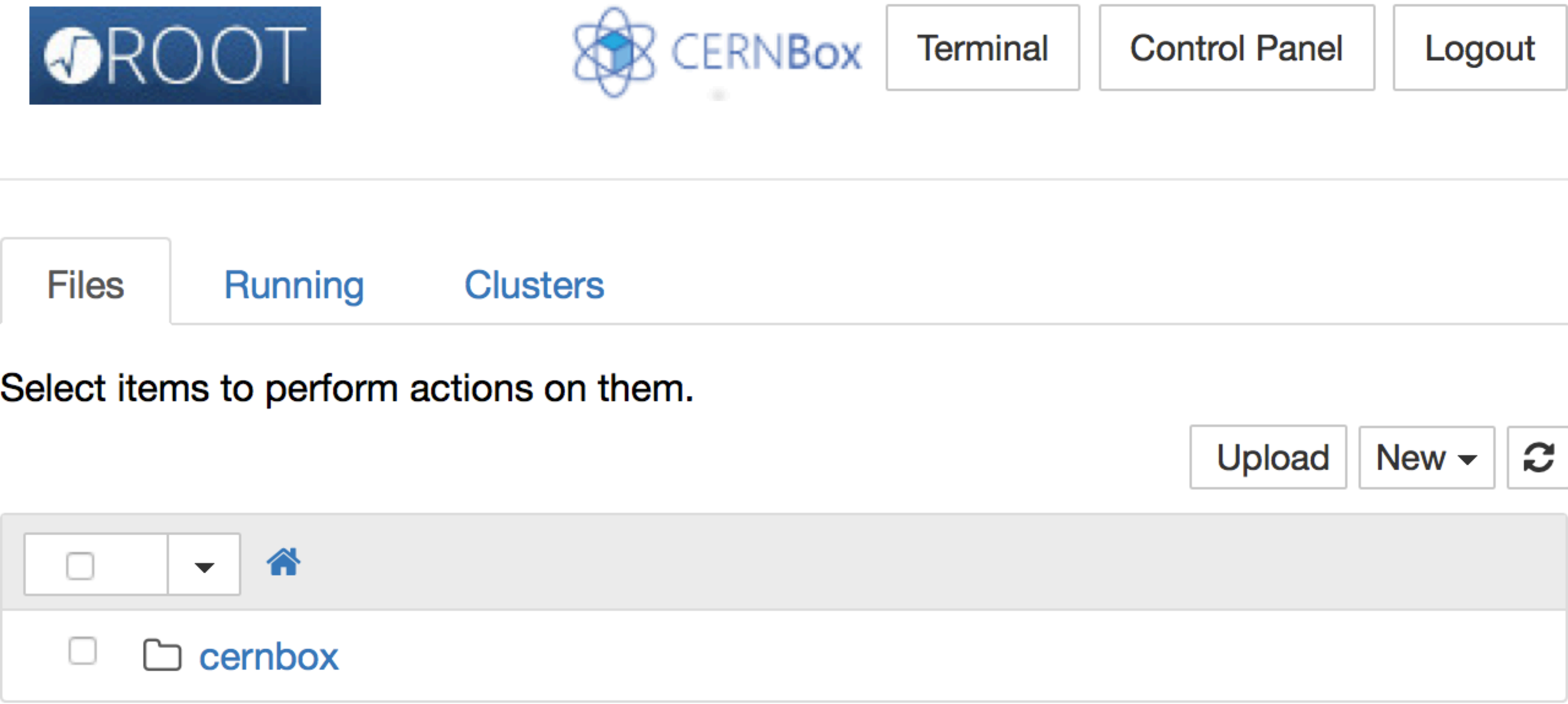
Username:

rw15u098

Password:

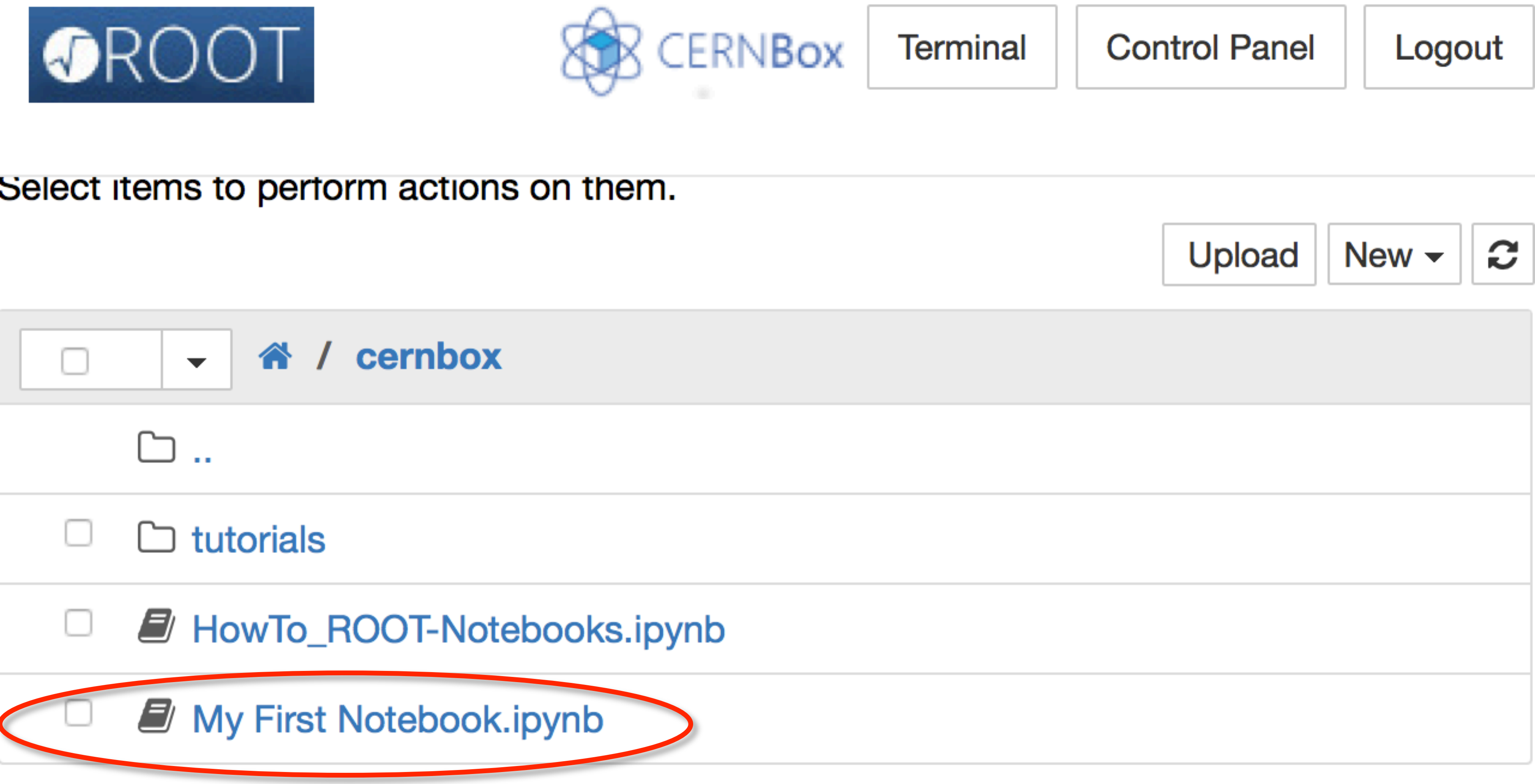
••••••••••••••••

Live Demo



Select items to perform actions on them.

Live Demo



Select items to perform actions on them.

□ / cernbox

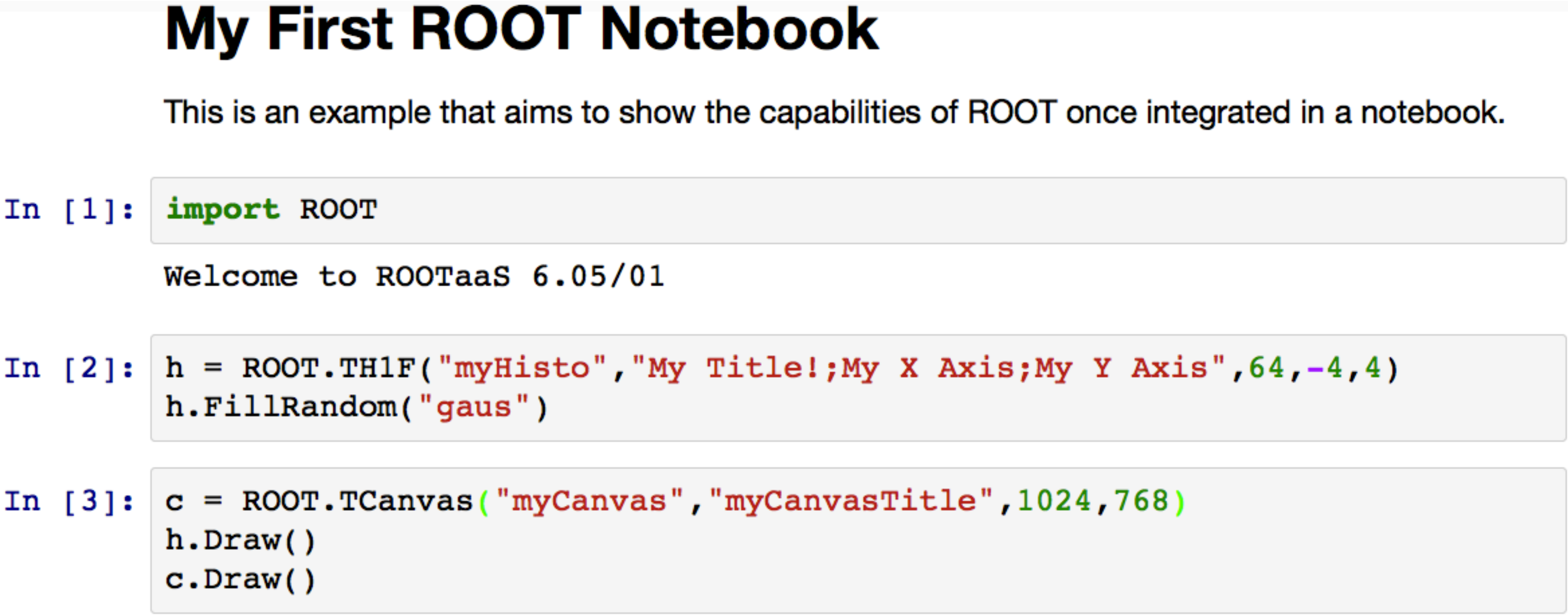
..

□ tutorials

□ HowTo_ROOT-Notebooks.ipynb

□ My First Notebook.ipynb

Live Demo



My First ROOT Notebook

This is an example that aims to show the capabilities of ROOT once integrated in a notebook.

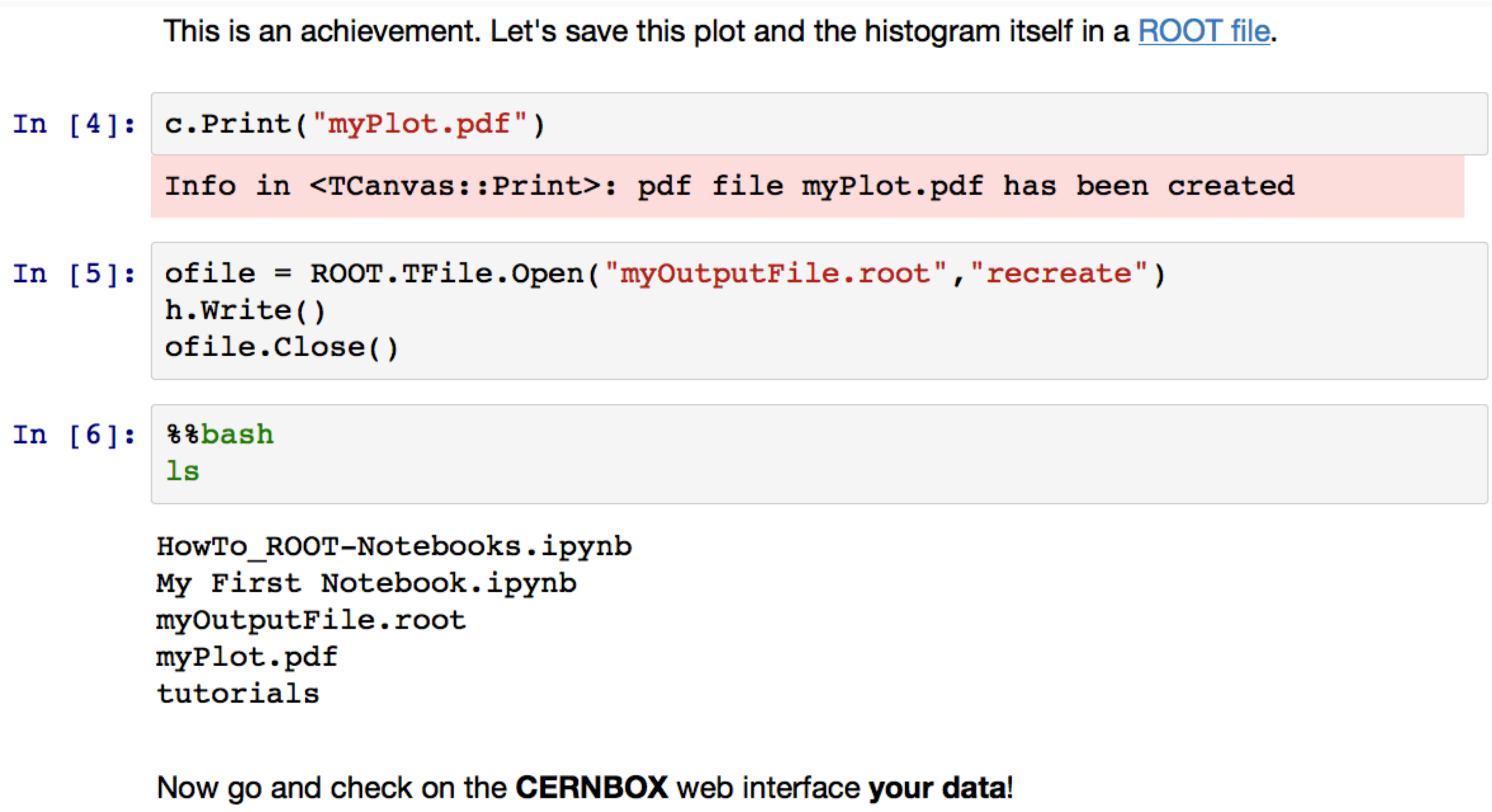
```
In [1]: import ROOT
```

```
Welcome to ROOTaaS 6.05/01
```

```
In [2]: h = ROOT.TH1F("myHisto","My Title!;My X Axis;My Y Axis",64,-4,4)
        h.FillRandom("gaus")
```

```
In [3]: c = ROOT.TCanvas("myCanvas","myCanvasTitle",1024,768)
        h.Draw()
        c.Draw()
```

Live Demo



This is an achievement. Let's save this plot and the histogram itself in a ROOT file.

In [4]:
```
c.Print("myPlot.pdf")
```

```
Info in <TCanvas::Print>: pdf file myPlot.pdf has been created
```

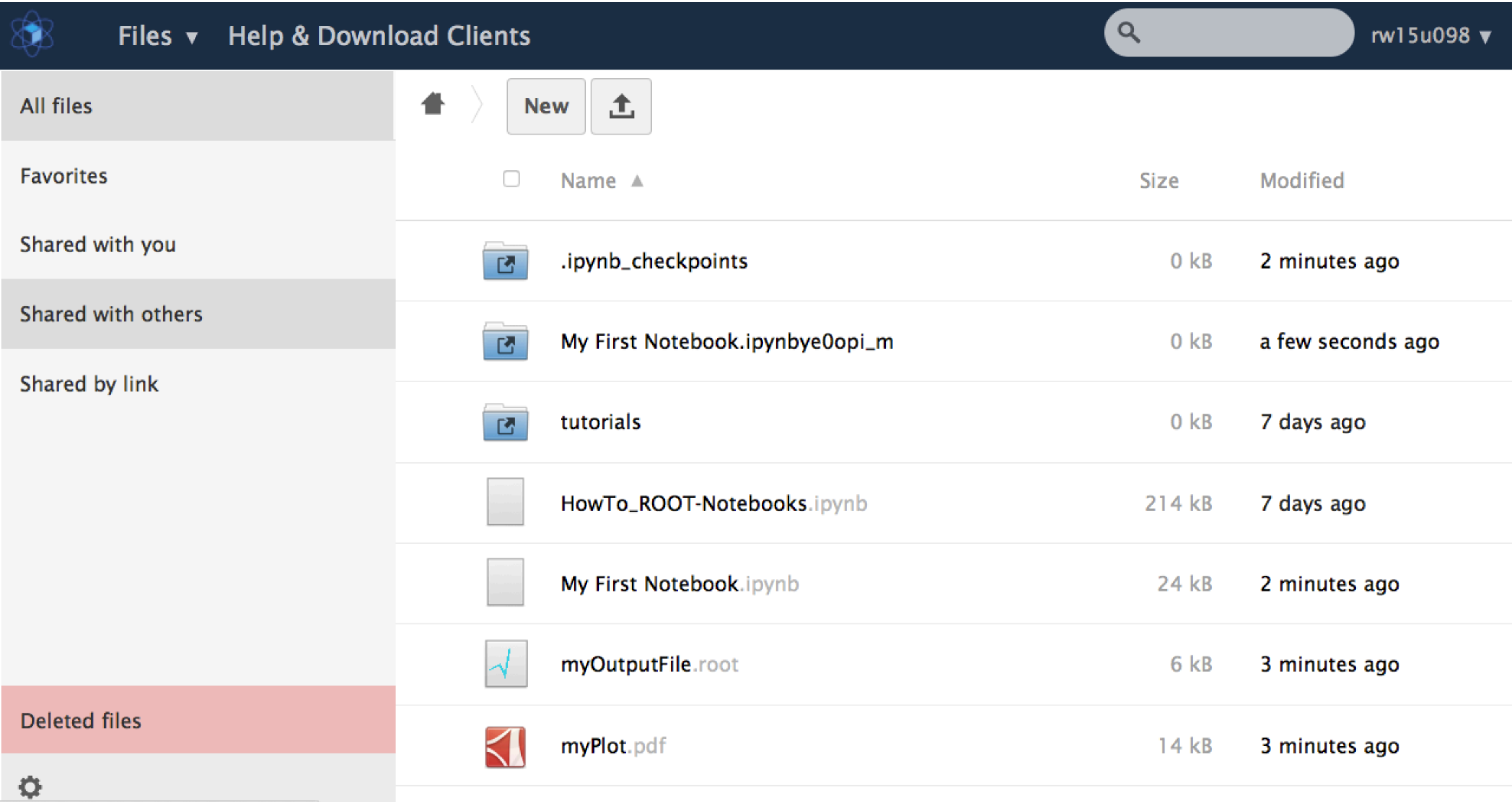
In [5]:
```
ofile = ROOT.TFile.Open("myOutputFile.root","recreate")
h.Write()
ofile.Close()
```

In [6]:
```
%%bash
ls
```

```
HowTo_ROOT-Notebooks.ipynb
My First Notebook.ipynb
myOutputFile.root
myPlot.pdf
tutorials
```

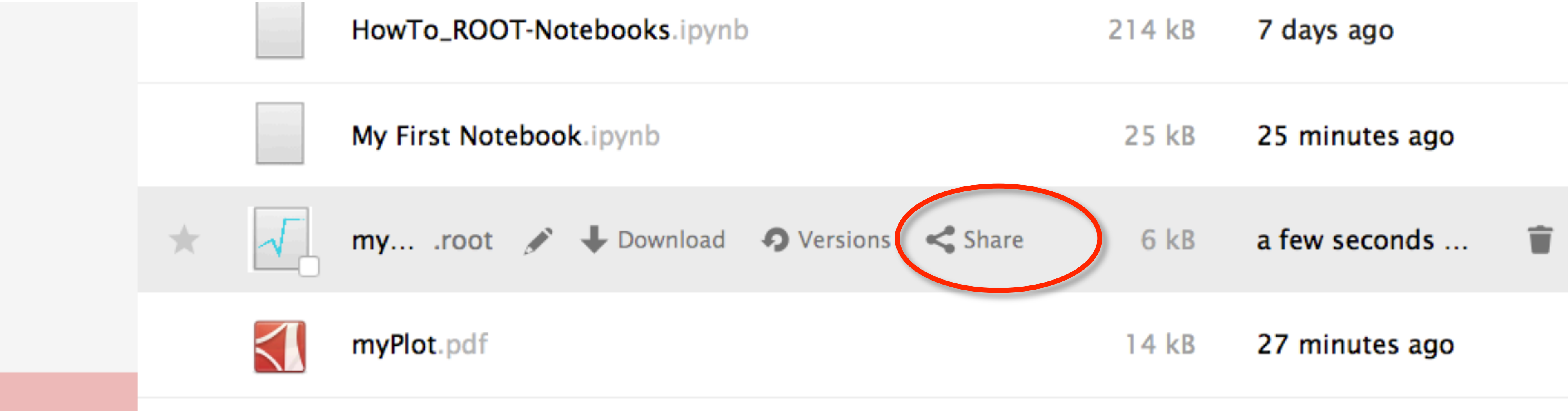
Now go and check on the **CERNBOX** web interface **your data**!

Live Demo



□	Name ▲	Size	Modified
	.ipynb_checkpoints	0 kB	2 minutes ago
	My First Notebook.ipynbye0opi_m	0 kB	a few seconds ago
	tutorials	0 kB	7 days ago
	HowTo_ROOT-Notebooks.ipynb	214 kB	7 days ago
	My First Notebook.ipynb	24 kB	2 minutes ago
	myOutputFile.root	6 kB	3 minutes ago
	myPlot.pdf	14 kB	3 minutes ago

Live Demo



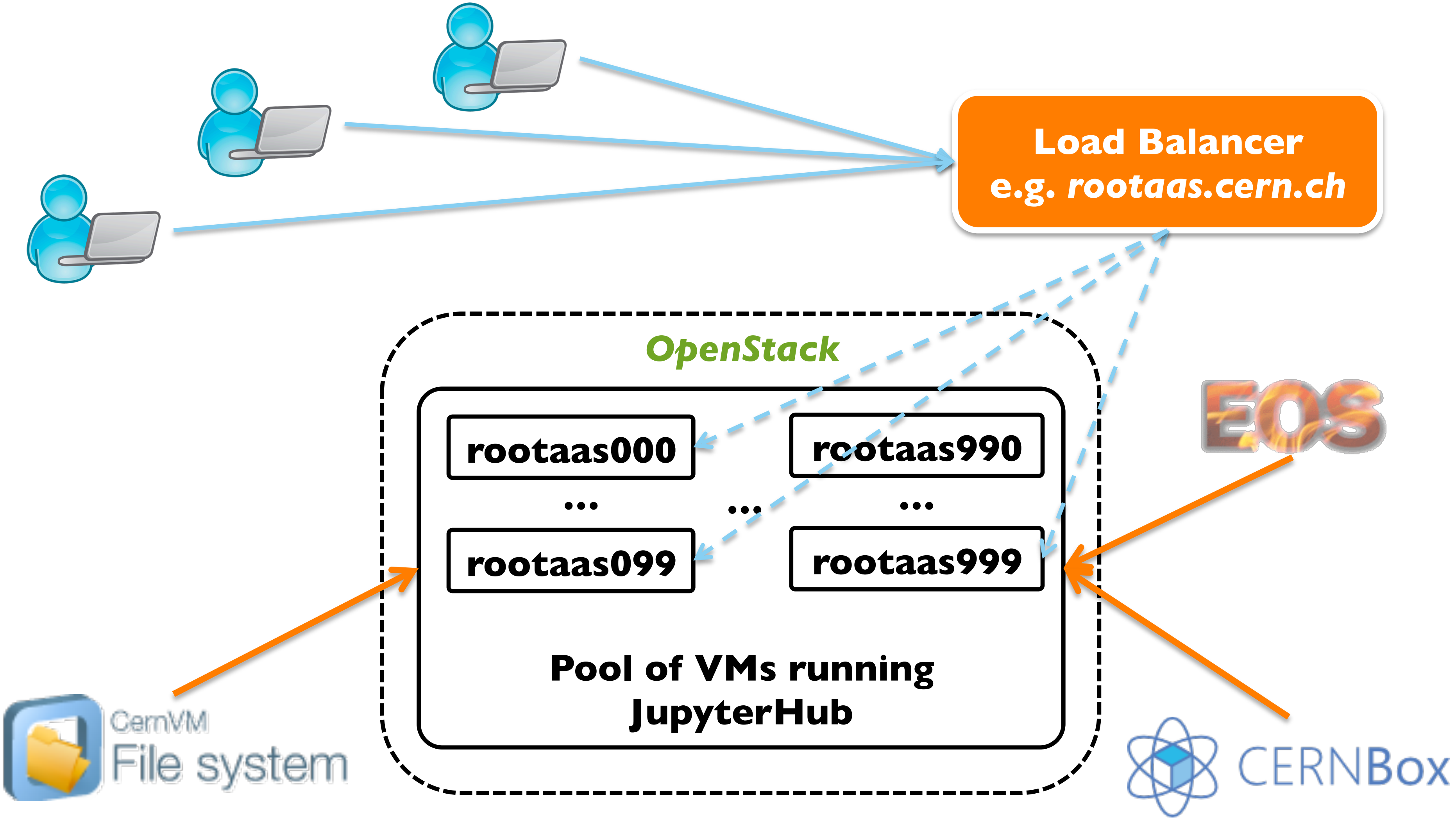
	HowTo_ROOT-Notebooks.ipynb		214 kB	7 days ago
	My First Notebook.ipynb		25 kB	25 minutes ago
★	my… .root	Download Versions Share	6 kB	a few seconds …
	myPlot.pdf		14 kB	27 minutes ago

Chosen Technology

Jupyterhub: manages login of users and redirection to notebook

- Existing solution: https://github.com/jupyter/jupyterhub
- Allows encapsulation: spawn Linux container at logon
  - User isolated from the host, modulo volumes explicitly mounted (cvmfs, CERNBox)
- Needs to be customised, e.g.:
  - CERN sign-on procedure
  - Docker image for the container

One of the many possible designs



Load Balancer
e.g. *rootaas.cern.ch*

OpenStack

rootaas000

rootaas990

…

…

…

rootaas099

rootaas999

Pool of VMs running JupyterHub

- Experiments' Software
- LCG Externals and Releases

- Experiment's Data
- Users' Data

A Success Story

- CERN Summer Student Program
- >100 Students hosted at CERN for 8-13 weeks
- Internship + Lectures program
  - ROOT Tutorial for students organised, 4 sessions

Last session of the tutorial: interactive notebooks offered

- Single 24 cores box, Beta version of the software layer
- **50 participants, perfect scaling, a success!**
  - https://indico.cern.ch/event/407519

The Demo

Try ROOTaaS Now!

- Get ROOT, try it in a notebook on your laptop or…
- Access the demo server:

www.cern.ch/rootaasdemo

Get a ROOTaaS account now (talk to Enric or Danilo)!

- Take a look to the provided notebooks, modify them, run them
  - Produce results!
  - Access data and plots via CERNBox (https://cernbox.cern.ch)
  - Develop locally, sync directory, run your code in the notebook
  - Share with others results and more

Thanks to the IT-DSS group, in particular *L. Mascetti, K. Moscicki and M. Lamanna* for their fundamental contribution to the creation of this demo!

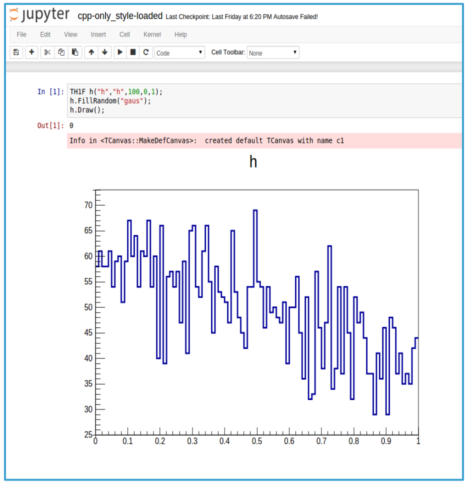
Conclusions

- ROOT is now integrated with notebooks
  - Python and C++ interactive shells
  - Tab completion, C++/Python integration, syntax highlighting, graphics inlining, shell commands
  - Available now (6.05/02)!
- Integration with the CERN services portfolio
  - Collaborating with IT department: started to capitalise on interplay with storage services
  - Work in progress, usable demo available to be tried at the ROOT workshop!
  - Bright future ahead of us: e.g. r&d on containers scheduling, job submission steering from notebook (e.g. with Ganga), software provision models.

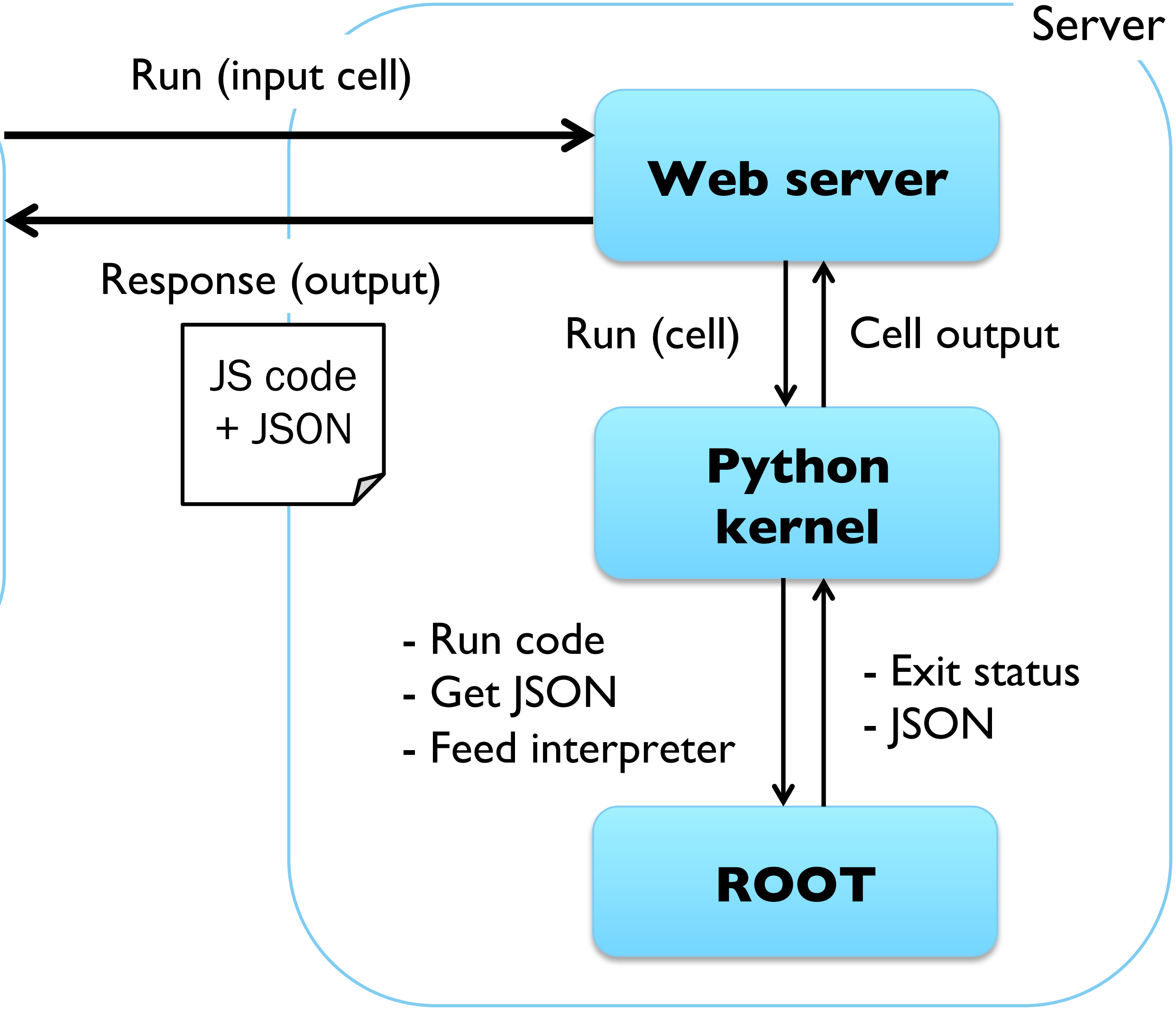
Backup

How this works On One Machine

Client



```
In [1]: TH1F h("h","h",100,0,1);
        h.FillRandom("gaus");
        h.Draw();
Out[1]: 0
        Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
```



Server

Run (input cell)

Response (output)

JS code + JSON

Web server

Run (cell)

Cell output

Python kernel

- Run code
- Get JSON
- Feed interpreter

- Exit status
- JSON

ROOT

Need to scale on many machines!

Adding the “ROOT Prompt Kernel”

How To Add the ROOT Prompt Kernel to the IPython Notebooks?

The best way to access a ROOT Prompt flavoured notebook is to incorporate the ROOT Prompt kernel in your IPython installation. Then, these are the steps to follow:

```
# Install IPython notebook 3.2. Note that C++ highlighting
# is supported up to this version.
sudo pip install -lv ipython[notebook]==3.2.0

# Create an installation directory, fetch the notebook settings
export NBINSTDIR=ROOTPromptNBKernel
http://root.cern.ch/notebooks/local_inst/rootnb_local.tar.gz

# Unpack the ROOT notebook local installation file
tar xvzf rootnb_local.tar.gz -C $NBINSTDIR

# Set environment
export IPYTHONDIR=$NBINSTDIR/rootnb_local

#Launch ROOT notebook
ipython notebook
```

You will be now able to select a ROOT Prompt kernel. Note that this procedure assumes that executable python is the executable of Python2.

Start a Notebook in a Laptop

```
$ jupyter notebook
```

That command:

1. Starts a notebook
2. Opens it in the browser

Proxy through an SSH Tunnel - 1

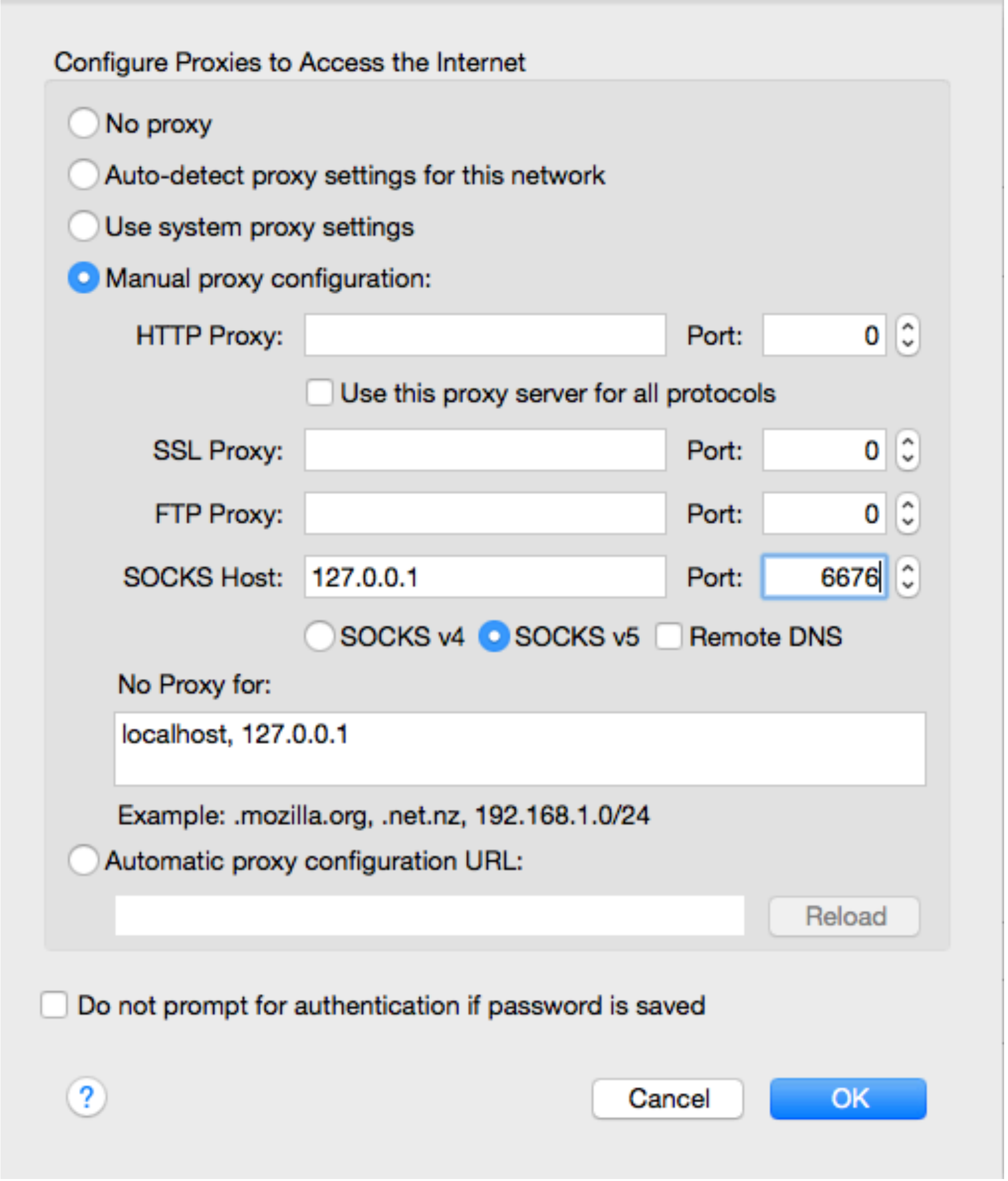
Open the tunnel (on Windows, use Putty):

```
ssh –D portNumber user@server
```

Set up the proxy (Firefox):

- Preferences->Advanced->Network-> Connection
- Radio Button: Manual Proxy Configuration
- SOCKS Host, set port to *portNumber*

Proxy through an SSH Tunnel - 2



Configure Proxies to Access the Internet

○ No proxy

○ Auto-detect proxy settings for this network

○ Use system proxy settings

● Manual proxy configuration:

HTTP Proxy: Port: 0

□ Use this proxy server for all protocols

SSL Proxy: Port: 0

FTP Proxy: Port: 0

SOCKS Host: 127.0.0.1 Port: 6676

○ SOCKS v4 ● SOCKS v5 □ Remote DNS

No Proxy for:

localhost, 127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

○ Automatic proxy configuration URL:

□ Do not prompt for authentication if password is saved