```
#!/usr/bin/env python

# ------------------------------------------------------------------------------------ //
#
#  ROOT macro for introducing ROOT with examples.
#
#  Run this pyROOT macro by:
#    From prompt: > ./RootIntro.py
#
#  Author: Troels C. Petersen (NBI/CERN)
#  Email:  petersen@nbi.dk
#  Date:   11th of August 2014
#
#  ---------------------------------------------------------------------------------- //

from ROOT import *
from array import array
import math

gStyle.SetOptStat("emr")        # Include "Entries", "Mean" and "Rms" in statistics box!
gStyle.SetOptFit(1111)          # ROOT being told to show all fitting parameters in box!

r = TRandom3()                  # Random generator

SavePlots = False               # Save plots?

verbose = False                 # Print a lot to screen?
Nverbose = 10                   # If so, how much?



#----------------------------------------------------------------------------------
# Histogram with fit (1D):
#
# The lines below gives an example of how to fill histograms and subsequently
# plot and fit them.
#
#----------------------------------------------------------------------------------

# Statistics and fitting result box replaced in:
gStyle.SetStatX(0.92)
gStyle.SetStatY(0.92)

# Define histograms (name, title, number of bins, minimum, maximum):
#Hist_x = TH1F("Hist_x", "Hist_x", 100, -5.0, 5.0)
#Hist_y = TH1F("Hist_y", "Hist_y", 100, -5.0, 5.0)
Hist_x = TH1F("Hist_x", "Hist_x;x-axis;Frequency", 100, -5.0, 5.0)
Hist_y = TH1F("Hist_y", "Hist_y;y-axis;Frequency", 100, -5.0, 5.0)

# Loop to get some random values and fill them into histogram:
Npoints = 10000                       # Number of random points produced
for iexp in range( Npoints ) :
    Hist_x.Fill(r.Gaus()*0.8-0.5)     # Take a Unit Gaussian number (r.Gaus()) and shift
width and mean
    Hist_y.Fill(r.Gaus()*1.3+0.5)     # Same, same, but different!

# Plot result:
# ------------
canvas = TCanvas( "canvas", "canvas", 50, 50, 1200, 600 )

Hist_x.SetTitle("Distribution of Gaussian numbers")
Hist_x.GetXaxis().SetRangeUser(-5.0, 5.0)         # Set the range you want to plot in on X-
axis
Hist_x.SetLineColor(kBlue)                        # Set color (see:
```

```
http://root.cern.ch/root/html/TAttFill.html)
Hist_x.SetLineWidth(2)                            # Guess yourself!
Hist_x.Draw("e")                     # The option "e" means "show errors" (Poisson!)
                                     # For plotting options, see:
http://root.cern.ch/root/html/THistPainter.html

# Fitting histogram (with predefined function):
fit_x = TF1("fit_x", "gaus", -2.0, 1.0)         # Here "gaus" is a predefined function!
fit_x.SetLineColor(kBlue-8)                     # Different kind of blue...
fit_x.SetLineWidth(3)
Hist_x.Fit("fit_x")

# Drawing a second histogram:
Hist_y.SetLineColor(kRed)
Hist_y.SetLineWidth(2)
Hist_y.Draw("same")                  # The option "same" makes it plot on top of previous
plot(s)

# Example of how to get e.g. means from a histogram and result of fit:
print "Means:   mu_x = %6.3f    mu_y = %6.3f"%(Hist_x.GetMean(), Hist_y.GetMean())
print "Fitted mean of x:   mu_hat = %6.3f +- %5.3f"%(fit_x.GetParameter(1),
fit_x.GetParError(1))

# Legend (description of what is in the plot):
leg = TLegend( 0.15, 0.70, 0.38, 0.85 )
leg.SetFillColor(kWhite)
leg.SetLineColor(kWhite)
leg.AddEntry(Hist_x, " Gaussian (#mu = -0.5)", "L")
leg.AddEntry(fit_x, " Fit with Gaussian to x", "L")
leg.AddEntry(Hist_y, " Gaussian (#mu = +0.5)", "L")
leg.Draw()

canvas.Update()
if (SavePlots):
    canvas.SaveAs("Histogram.pdf")





#----------------------------------------------------------------------------------
# Graph with fit (1D):
#
# The lines below gives an example of how to make a graph with errors on the points
# (i.e. not a histogram, which is a number of counts in bins) and subsequently
# plot and fit it in three different ways, illustrating the versatility of ROOT.
#
#----------------------------------------------------------------------------------

# Statistics and fitting results replaced in:
gStyle.SetStatX(0.52)
gStyle.SetStatY(0.86)

# Define a graph with errors:
# Graphs in ROOT requires arrays, which is why these are produced below:
Ndata = 20
x  = array( 'f', [0.0]*Ndata )
y  = array( 'f', [0.0]*Ndata )
ex = array( 'f', [0.0]*Ndata )
ey = array( 'f', [0.0]*Ndata )

# Having defined four arrays filled with zeros, below we fill it with values:
for i in range ( Ndata ) :
    x[i] = 0.4 + 0.1*i
    y[i] = (x[i]-1.0)*(x[i]-1.0) - 2.0 + r.Gaus(0.0, 0.2)    # Adding a Gaussian error of 0.2
    ex[i] = 0.0                                              # Not really needed
    ey[i] = 0.2                                              # Assigning the error to each
point

# Define the graph:
Graph_x = TGraphErrors(Ndata, x, y, ex, ey)


# Plot graph:
# -----------
canvas2 = TCanvas( "canvas2", "canvas2", 100, 100, 1200, 600 )

Graph_x.SetTitle("Fit of a graph")
Graph_x.GetXaxis().SetRangeUser(-5.0, 5.0)
Graph_x.GetXaxis().SetTitle("Deciliters of alcohol")
Graph_x.GetYaxis().SetTitle("Spirit")
Graph_x.SetMarkerStyle(20)
Graph_x.SetLineColor(kBlue)
Graph_x.SetLineWidth(1)
Graph_x.Draw("AP")                      # Draw the Axis and the Points!


# Fit graph:
# ----------
# There are three ways of fitting!
#   1: Predefined function (ROOT has gaus, expo, polX, etc.)
#   2: Writing function explicitly (for simple functions)
#   3: Defining external function (for advanced functions)

# Predefined function:
fit_x1 = TF1("fit_x1", "pol1", 0.3, 2.4)        # Note how ROOT understands "pol1"...
fit_x1.SetLineColor(kRed)
Graph_x.Fit("fit_x1")

# Writing function explicitely:
fit_x2 = TF1("fit_x2", "[0] + [1]*x + [2]*sqrt(x)", 0.3, 2.4)     # Here we wrote the
function out!
fit_x2.SetParameters(-1.0, -1.0, 1.0)           # Remember to give good starting values!
fit_x2.SetLineColor(kMagenta)
Graph_x.Fit("fit_x2","+")                       # Option "+" adds fit (not deleting old
one!).

# Defining external function:
def func_advanced (x, p) :
    if (x[0] < 1.25) :
        return p[0] + p[1]*x[0] + p[2]*x[0]*x[0]
    else :
        return p[0] + p[1]*x[0] + 0.9*x[0]*x[0]

fit_x3 = TF1("fit_x3", func_advanced, 0.3, 2.4, 3)   # Here you need to define number of
variables (3)
fit_x3.SetParameters(-1.0, -1.0, 1.0)           # Remember to give good starting values!
fit_x3.SetLineColor(kBlue)
Graph_x.Fit("fit_x3", "+")

canvas2.Update()
if (SavePlots):
    canvas2.SaveAs("Graph.pdf")


# ----------------------------------------------------------------------------------
raw_input('Press Enter to exit')
```