```
# practicing fitting histograms
# As is true in many programming languages, a goal can be achieved
# by many methods
# The below example shows different approaches to histogram fitting
# including both the PDF Poisson example and class Gaussian work

import LT.box as B
import numpy as np
import LT_Fit.parameters as P  # get the parameter module
import LT_Fit.gen_fit as G     # load the genfit module
import scipy.special as ms     # need this for the factorial function
                              # called in myfunl(x) for Poisson fitting

# Pdf Poisson Fitting from Data File
file_name = 'histoFitting.data'
f = B.get_file(file_name)

# get the data from column 'A'
A = B.get_data(f, 'A')

# here histo is a method inside LT.box. There are other ways of
# making histograms as well
h2 = B.histo(A, (0.5, 10.5), 10)
# this means A is the data being histogramed, (0.5, 10.5) is the range,
# and 10 is number of bins, obviously the bin width is 1.

h2.plot()
# histograms can be fitted with the build in function for the
# Lt.box.histo object
# for example h2.fit() or h2.fit(2.0, 8.0) will fit the range between 2.0, and 8.
# However, this is limited
# the example below provides a way to fit histograms with any user
# defined function, including linear, polynomial, gaussian, poisson, etc.....


# put the bin centers and bin contents to two arrays
hx_poisson = h2.bin_center
hy_poisson = h2.bin_content
dy_poisson = np.sqrt(hy_poisson)
print("hy is:\n", hy_poisson)
print("dy is:\n", dy_poisson)

# Define parameters for Poisson function
mu = P.Parameter(2., 'mu')
norm = P.Parameter(10., 'norm')

# The function defined here is a pure poisson function. The initial
# "guess" of the parameters are given above
def myfunl(x):
   value = norm()*mu()**x*np.exp(-mu())/ms.gamma(x+1.0)
   return value
```

```
# you need to provide the independent variable x, and the yvalue at x
# in the forms of arrays. in this case, it's the bin_center (hx_poisson, as defin
# being x, and bin_content (hy_poisson) being y
# the way the fitting is done is the same as if you are simply fitting a Y vs X p

fit_poisson = G.genfit(myfunl, [mu, norm], x=hx_poisson, y=hy_poisson)

B.plot_line(fit_poisson.xpl, fit_poisson.ypl, color='red')
h2.plot()
B.pl.title('Poissin fit of the histofitting.data')
B.pl.show()



print("Poisson fit completed")

# From class: Gaussian Fitting from random ages

# Create an array with random ages from class work
age = ([23,19,25,20,18,17,26,30,17,21,22,25,27,7,37,47,19,24,25,20,22,
        30,20,26,27,20,])

# variable to create a histogram including the array age within the range
# (5, 50) separated in different number of 'bins'
# You can try different bin numbers by uncommenting:

# 5 bins version
# a1=B.histo(age, (5,50), 5)
# Plot the histogram
# a1.plot()

# Separate in 10 bins instead of 5
# a2=B.histo(age, (5,50), 10)
# plot
# a2.plot()

# Separate in 40 bins instead
a3 = B.histo(age, (5,40), 40)
# plot
a3.plot()
B.pl.title('Gaussian fit of the smaller age dataset')

#### fitting
# put the bin centers and bin contents to two arrays
hx = a3.bin_center  # Note: originally had hx=a3.bin_content which was wrong
hy = a3.bin_content
unc = np.sqrt(np.maximum(hy, 1.0))  # uncertainty calculation

# Alternative fitting methods (from class):
# fit histogram polynomial
# fith = B.polyfit(hx, hy, order=10)
# B.plot_line(fith.xpl, fith.ypl)
# regular fit
# fith = a3.fit()

# Extended age dataset for better statistics
age2 = ([23,19,25,20,18,17,26,30,17,21,22,25,27,7,37,47,19,24,25,20,22,
         30,20,26,27,20,47,45,39,45,23,36,44,36,24,33,56,32,49,35,35,45,
         54,49,56,75,42,63,45,34,36,48,48,38,24,52,34,24,62,54,32,36,45,34])

# Separate in 60 bins for the larger dataset
a4 = B.histo(age2, (5,60), 60)
# plot
a4.plot()
B.pl.title('Gaussian fit of the larger age dataset')

## fitting the extended dataset
# put the bin centers and bin contents to two arrays
hx2 = a4.bin_center
hy2 = a4.bin_content
unc2 = np.sqrt(np.maximum(hy2, 1.0))  # uncertainty

# fit histogram with Gaussian function
# Define parameters with better initial guesses based on your histogram
p1 = P.Parameter(5, 'p1')   # amplitude - should match peak height (~5)
p2 = P.Parameter(45, 'p2')  # mean/center - peak appears around age 45-50
p3 = P.Parameter(12, 'p3')  # standard deviation - data looks spread out

# define the gaussian function
def gauss(x):
    return p1() * np.exp(-0.5 * ((x - p2()) / p3())**2)

# create fitted with gauss variable - use the larger dataset (hx2, hy2)
fit = G.genfit(gauss, [p1, p2, p3], x=hx2, y=hy2, y_err=unc2)

# Perform the fit
fit.fit()

# Print results manually since show_results() doesn't exist
print("Gaussian fit results:")
print(f"Amplitude (p1): {p1():.3f}")
print(f"Mean (p2): {p2():.3f}")
print(f"Standard deviation (p3): {p3():.3f}")
print(f"Chi-squared: {fit.chi2:.3f}")

# Plot the fitted Gaussian curve using the same x data
B.plot_line(hx2, gauss(hx2), color='blue')

print("Gaussian fit completed!")
print(f"Final parameters - Amplitude: {p1():.2f}, Mean: {p2():.2f}, Std Dev: {p3(
B.pl.show()
```