```
# practiging fitting plots with various functionsimport numpy as np
import numpy as np
import LT.box as B

file_name = 'examples.data'
f = B.get_file(file_name)
# get the data
# current
A = B.get_data(f, 'A')
b = B.get_data(f, 'b')
db = B.get_data(f, 'db')
C = B.get_data(f, 'C')
D = B.get_data(f, 'D')
# The following examples fit1 to fit4 fits C vs A, using linear fit,
#polynomial fits, in either the whole ranges (fit1, fit2) or a
#subrange (fit 3, 4)
B.plot_exp(A, C, db)
B.pl.show()
#You shoudl uncomment the next line to see what it does to the x-axi
#Can you set the y-axis title now?
#B.pl.xlabel("x (unit)")
#fit1 = B.linefit(A, C, db)
#B.plot_line(fit1.xpl, fit1.ypl)
#the following two lines selecting the ranges
r1 = B.in_window(4.0, 10.0, C)
r2 = B.in_window(2.0, 12.0, C)
#only fit the selected ranges as specified by r1 or r2
fit3 = B.linefit(A[r1], C[r1], db[r1])
B.plot_line(fit3.xpl, fit3.ypl)
#the fit below use second order-- defined by the "2" in the argument
#polynomial; you should change 2 to other integers and see how the
#fits are different
fit4 = B.polyfit(A[r2], C[r2], db[r2], 2)
B.plot_line(fit4.xpl, fit4.ypl)
speed = fit4.parameters[1].value #or speed = fit4.par[1]
d_speed = fit4.parameters[1].err #or d_speed = fit4.sig_par[1]
#printing out the fitting parameters with proper significant digits
print "\nspeed is %.3E +/- %.3E m/s\n" % (speed, d_speed)
```