NORTHERN ILLINOIS UNIVERSITY

PHYSICS DEPARTMENT

Physics 374 – Junior Physics Lab Spring 2021

Python Tutorial #7

PyFit: Using Dialog controls (Widgets)

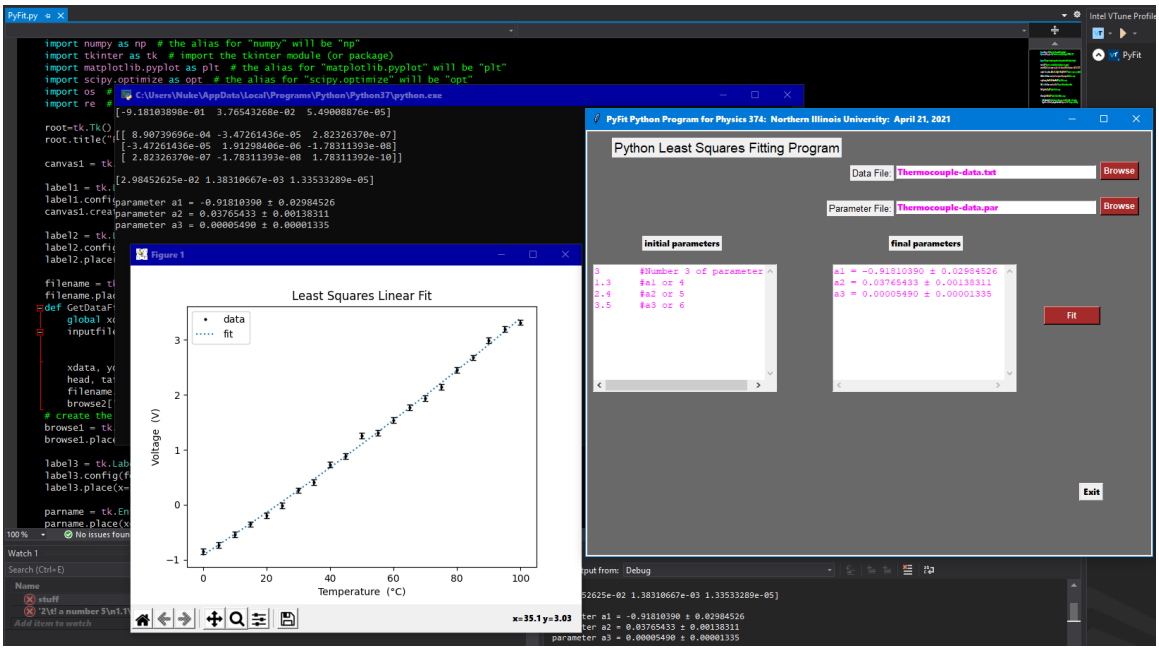
In this tutorial, we will learn how to create a GUI program using *tkinter* and widgets (dialog controls) to make a user friendly, fully interfaceable program. This program, called PyFit, will ask the user for a data file and a parameter file, put the information from the parameter file into a textbox for the user to see, perform a fit when the user presses a button, put the results into another textbox for the user to see, and make a plot using *matplotlib*.

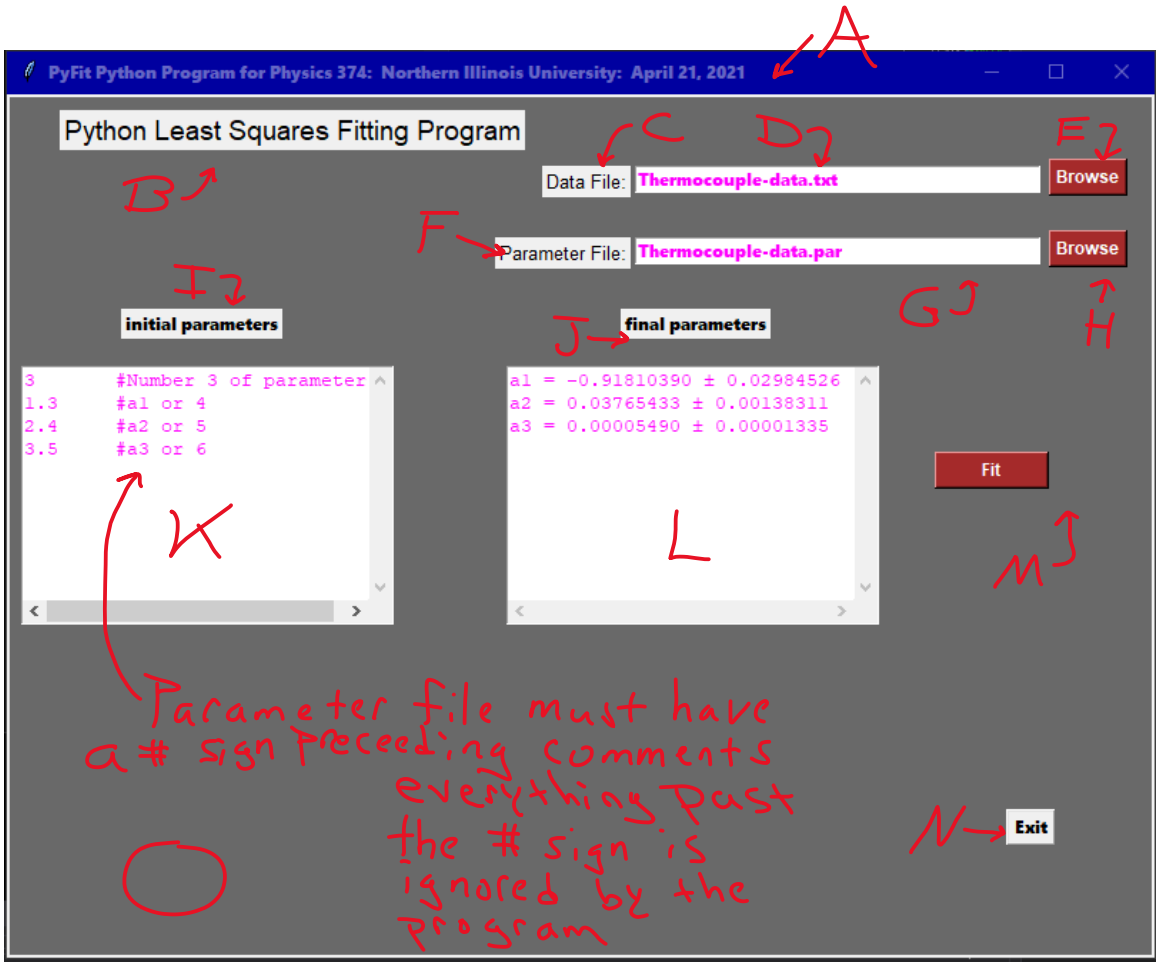
We will need to import new modules into the program called *os* and *re*. The *os* module has functions that allow us to split the filename from its path string, and the *re* module allows us to parse numbers (the parameters) from the input parameter textbox string. The editboxes (called *Entry* in Python) and textboxes (called *Text* in Python) contain only character strings (never floating point numbers or integers).

Good tutorials for *os*, *re*, and *tkinter* can be found at: *see the end of this document*

Open up Microsoft Visual Studio and copy and paste the code in the Word document, *PyFit-code.docx*, into a project called *PyFit*:

The output is shown below after entering the data file and parameter file and pressing the *Fit* button:





The parts of the program associated with the letters above are indicated in the program as shown below:

```
import numpy as np  # the alias for "numpy" will be "np"
import tkinter as tk  # import the tkinter module (or package)
import matplotlib.pyplot as plt  # the alias for "matplotlib.pyplot" will be "plt"
import scipy.optimize as opt  # the alias for "scipy.optimize" will be "opt"
import os  # import the operating system interface module (get the path of a file)
import re  # import the regular expression matching operations module (parsing strings)

root=tk.Tk()  # create a root window (the main window of this program)
root.title("PyFit Python Program for Physics 374:  Northern Illinois University:  April 21, 2021")

canvas1 = tk.Canvas(root, width = 800, height = 600, bg='#696969')  # create a canvas to put widgets on

label1 = tk.Label(root, text='Python Least Squares Fitting Program') # title of canvas
label1.config(font=('helvetica', 14))
canvas1.create_window(400,25,window=label1)  # put label on canvas

label2 = tk.Label(root, text='Data File:')  # create label for filename editbox
label2.config(font=('helvetica', 10))
canvas1.create_window(407,60,window=label2)  # put label on canvas
```

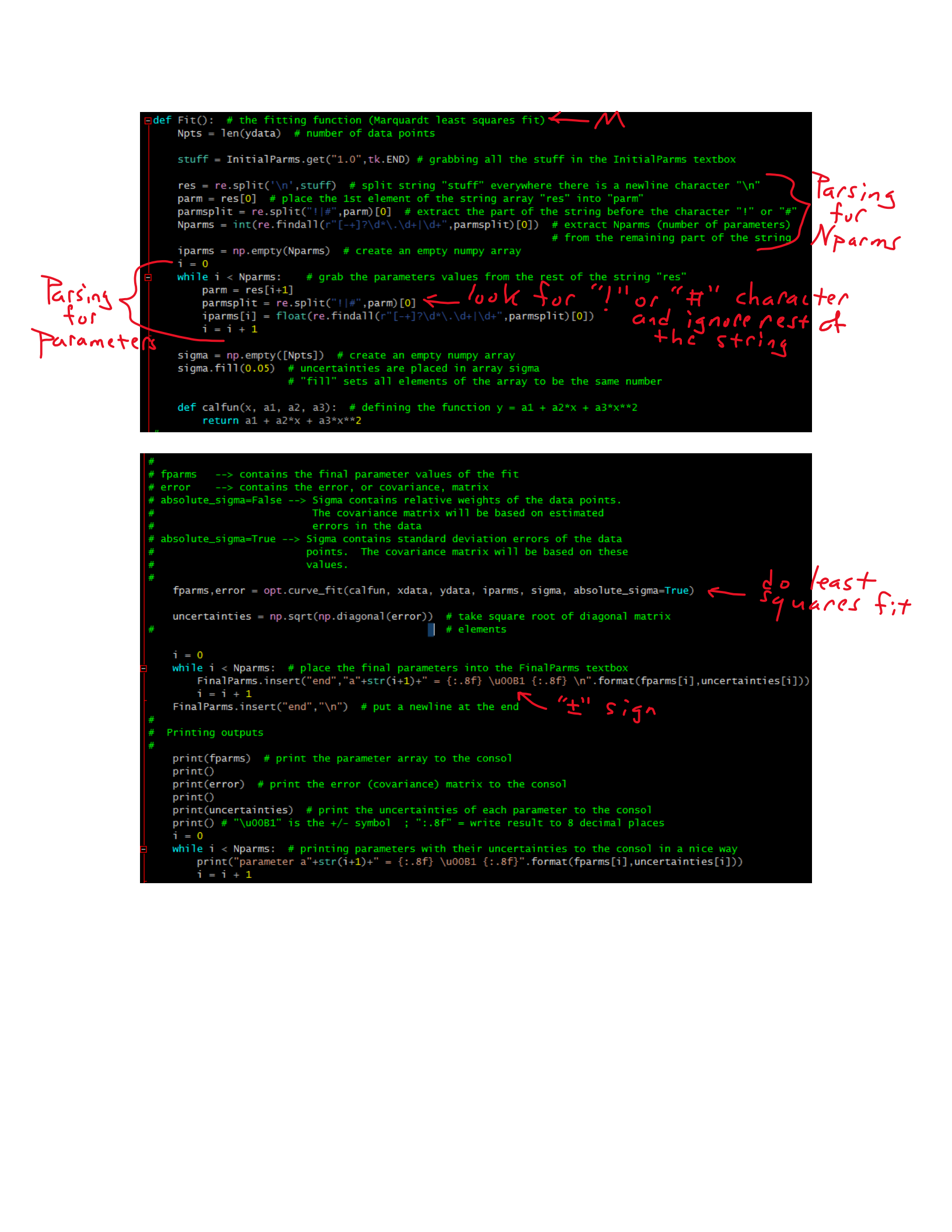
```
# get the position of mouse cursor to help find coordinate position to place Widgets on the Canvas
#def get_absolute_position(event=None):
#    x = root.winfo_pointerx() - root.winfo_rootx()
#    y = root.winfo_pointery() - root.winfo_rooty()
#    print(x,y)
#root.bind('<Motion>', get_absolute_position)

filename = tk.Entry(root,width=40) # editbox for filename
canvas1.create_window(582,59,window=filename)  # put editbox on canvas
def GetDataFile():  # do this command when browse button for filename is pressed
    global xdata,ydata  # making xdata and ydata arrays assessible outside of the function
    inputfile = tk.filedialog.askopenfilename(title="Input Data File",  # opens a file dialog
                                      filetypes = (("Text files","*.txt*"),
                                                   ("all files","*.*")))
    xdata, ydata = np.loadtxt(inputfile, unpack=True)  # reading in data into xdata and ydata arrays
    head, tail = os.path.split(inputfile)  # splits the filename from the rest of the path
    filename.insert(0,tail)  # only grab the filename and insert it into the filename editbox
    browse2['state'] = 'normal'  # make the parameter browse button active
# create the browse button for inputting the data file
browse1 = tk.Button(text='Browse', command=GetDataFile, bg='brown', fg='white', font=('helvetica', 9, 'bold'))
canvas1.create_window(755,59,window=browse1)  # put button on canvas

label3 = tk.Label(root, text='Parameter File:')  # create label for parameter file edit box
label3.config(font=('helvetica', 10))
canvas1.create_window(390,110,window=label3)  # put label on canvas
```

```
parname = tk.Entry(root,width=40) # editbox for parameter file
canvas1.create_window(582,109,window=parname)  # put editbox on canvas
def GetParFile():  # do this command when browse button for parameter file is pressed
    parfile = tk.filedialog.askopenfilename(title="Input Parameter File",  # opens a file dialog
                                      filetypes = (("Par files","*.par*"),
                                                   ("all files","*.*")))
    file1 = open(parfile)  # open the parfile
    stuff = file1.read()  # reading the contents of the parfile into string stuff
    file1.close()  # close the parfile
    head, tail = os.path.split(parfile)  # splits the parfile name from the rest of the path
    parname.insert(0,tail)  # only grab the parfile name and insert it into the parfile editbox
    InitialParms.insert("1.0",stuff)  # insert the contents of the parfile into InitialParms textbox
    FitButton['state'] = 'normal'  # make the Fit button active
# create the browse button for inputting the paramater file
browse2 = tk.Button(text='Browse', command=GetParFile, bg='brown', fg='white', font=('helvetica', 9, 'bold'), state="disabled")
canvas1.create_window(755,110,window=browse2)  # put button on canvas

label4 = tk.Label(root,text='initial parameters') # label for initial parameters textbox
canvas1.create_window(136,160,window=label4)  # put label on canvas
label5 = tk.Label(root,text='final parameters') # label for final parameters textbox
canvas1.create_window(470,160,window=label5)  # put label on canvas
```

```
# create Frame for initial parameters
iParmFrame = tk.Frame(root, borderwidth=1, relief="sunken")  # iParFrame will contain a textbox and two scrollbar widgets
InitialParms = tk.Text(iParmFrame, width=30, height=10, wrap="none", borderwidth=0)  # creating a textbox within the frame
iVsb = tk.Scrollbar(iParmFrame, orient="vertical", command=InitialParms.yview)    # vertical scrollbar widget
iHsb = tk.Scrollbar(iParmFrame, orient="horizontal", command=InitialParms.xview)  # horizontal scrollbar widget
InitialParms.configure(yscrollcommand=iVsb.set, xscrollcommand=iHsb.set)  # configuring vertical and horizontal scrollbars
InitialParms.grid(row=0, column=0, sticky="nsew")  # need the next 7 lines for everything to work
iVsb.grid(row=0, column=1, sticky="ns")
iHsb.grid(row=1, column=0, sticky="ew")
iParmFrame.grid_rowconfigure(0, weight=1)
iParmFrame.grid_columnconfigure(0, weight=1)
iParmFrame.pack()
canvas1.create_window(140,275,window=iParmFrame)  # put frame on canvas

# create Frame for final parameters
fParmFrame = tk.Frame(root, borderwidth=1, relief="sunken")  # fParFrame will contain a textbox and two scrollbar widgets
FinalParms = tk.Text(fParmFrame, width=30, height=10, wrap="none", borderwidth=0)  # creating a textbox within the frame
fVsb = tk.Scrollbar(fParmFrame, orient="vertical", command=FinalParms.yview)    # vertical scrollbar widget
fHsb = tk.Scrollbar(fParmFrame, orient="horizontal", command=FinalParms.xview)  # horizontal scrollbar widget
FinalParms.configure(yscrollcommand=fVsb.set, xscrollcommand=fHsb.set)  # configuring vertical and horizontal scrollbars
FinalParms.grid(row=0, column=0, sticky="nsew")  # need the next 7 lines for everything to work
fVsb.grid(row=0, column=1, sticky="ns")
fHsb.grid(row=1, column=0, sticky="ew")
fParmFrame.grid_rowconfigure(0, weight=1)
fParmFrame.grid_columnconfigure(0, weight=1)
fParmFrame.pack()
canvas1.create_window(475,275,window=fParmFrame)  # put frame on canvas
```

```
def Fit():  # the fitting function (Marquardt least squares fit)
    Npts = len(ydata)  # number of data points

    stuff = InitialParms.get("1.0",tk.END) # grabbing all the stuff in the InitialParms textbox

    res = re.split('\n',stuff)  # split string "stuff" everywhere there is a newline character "\n"
    parm = res[0]  # place the 1st element of the string array "res" into "parm"
    parmsplit = re.split("!|#",parm)[0]  # extract the part of the string before the character "!" or "#"
    Nparms = int(re.findall(r"[-+]?\d*\.\d+|\d+",parmsplit)[0])  # extract Nparms (number of parameters)
                                                                 # from the remaining part of the string
    iparms = np.empty(Nparms)  # create an empty numpy array
    i = 0
    while i < Nparms:    # grab the parameters values from the rest of the string "res"
        parm = res[i+1]
        parmsplit = re.split("!|#",parm)[0]
        iparms[i] = float(re.findall(r"[-+]?\d*\.\d+|\d+",parmsplit)[0])
        i = i + 1

    sigma = np.empty([Npts])  # create an empty numpy array
    sigma.fill(0.05)  # uncertainties are placed in array sigma
                      # "fill" sets all elements of the array to be the same number

    def calfun(x, a1, a2, a3):  # defining the function y = a1 + a2*x + a3*x**2
        return a1 + a2*x + a3*x**2
#
```

```
#
# fparms   --> contains the final parameter values of the fit
# error    --> contains the error, or covariance, matrix
# absolute_sigma=False --> Sigma contains relative weights of the data points.
#                          The covariance matrix will be based on estimated
#                          errors in the data
# absolute_sigma=True --> Sigma contains standard deviation errors of the data
#                         points.  The covariance matrix will be based on these
#                         values.
#
    fparms,error = opt.curve_fit(calfun, xdata, ydata, iparms, sigma, absolute_sigma=True)

    uncertainties = np.sqrt(np.diagonal(error))  # take square root of diagonal matrix
#                                                # elements

    i = 0
    while i < Nparms:  # place the final parameters into the FinalParms textbox
        FinalParms.insert("end","a"+str(i+1)+" = {:.8f} \u00B1 {:.8f} \n".format(fparms[i],uncertainties[i]))
        i = i + 1
    FinalParms.insert("end","\n")  # put a newline at the end
#
#  Printing outputs
#
    print(fparms)  # print the parameter array to the consol
    print()
    print(error)  # print the error (covariance) matrix to the consol
    print()
    print(uncertainties)  # print the uncertainties of each parameter to the consol
    print() # "\u00B1" is the +/- symbol  ; ":.8f" = write result to 8 decimal places
    i = 0
    while i < Nparms:  # printing parameters with their uncertainties to the consol in a nice way
        print("parameter a"+str(i+1)+" = {:.8f} \u00B1 {:.8f}".format(fparms[i],uncertainties[i]))
        i = i + 1
```



```
#
#  Plotting outputs
#
    plt.plot(xdata,ydata,"ko",label="data",ms=2)         # make a scatter plot for the data
    a1, a2, a3 = fparms  # set a1,a2,a3 parameters       # "ms" sets the size of the circle
    x = np.linspace(min(xdata), max(xdata), 100)  # array of 100 x-axis points
    yfit = calfun(x,a1,a2,a3)  # array of yfit values for each value of x array
    plt.plot(x,yfit,":",label="fit")  # plot the fitting curve with a dotted curve ":"
    plt.errorbar(xdata,ydata,yerr=sigma,capsize=3,fmt='ko',ms=1)  # plot error bars
    plt.legend()  # show the legend
    plt.title("Least Squares Linear Fit")  # plot title of the plot
# "\N{DEGREE SIGN}" is the degee symbol for degrees Celcius
    plt.xlabel("Temperature  (\N{DEGREE SIGN}C)")  # plot the x-axis label
    plt.ylabel("Voltage  (V)") # plot the y-axis label
    plt.show()  # command to draw the plot on the screen

# create the Fit button for starting the fitting routine
FitButton = tk.Button(text='Fit', command=Fit, bg='brown', fg='white', font=('helvetica', 9, 'bold'), width=10, state="disabled")
canvas1.create_window(690,270,window=FitButton)  # put button on canvas

exit_button = tk.Button(root, text="Exit", command=root.destroy)  # button for closing (exiting) the program
canvas1.create_window(720,510,window=exit_button)  # put button on canvas

canvas1.pack()  # organize all widgets on the canvas

root.mainloop()  # start the program and wait for commands
```

Plotting data and fit curve

M

N

infinite loop → the code just sits looking for inputs (pressing buttons, writing in textboxes) before doing any commands

Homework

Complete the x-ray diffraction lab fitting your diffraction spectra with a background function and a series of gaussian functions. You will be doing a 21 or 16 parameter fit (see the x-ray diffraction handout). Make appropriate modification to the code above to do your fit.

Good tutorials for *os*, *re*, *tkinter*, and python programming can be found at (you will find that I copied and pasted a lot of the code from these sites):

https://www.geeksforgeeks.org/file-explorer-in-python-using-tkinter/

https://docs.python.org/3/library/dialog.html

https://www.geeksforgeeks.org/writing-to-file-in-python/

https://www.python-course.eu/numpy_reading_writing.php

https://numpy.org/doc/stable/reference/generated/numpy.savetxt.html

https://docs.python.org/3/tutorial/inputoutput.html

https://stackoverflow.com/questions/7994461/choosing-a-file-in-python3

https://www.programcreek.com/python/example/95889/tkinter.filedialog.asksaveasfile

https://www.tutorialspoint.com/asksaveasfile-function-in-python-tkinter

https://www.geeksforgeeks.org/how-to-save-a-numpy-array-to-a-text-file/

https://stackoverflow.com/questions/11497376/how-do-i-specify-new-lines-on-python-when-writing-on-files

https://python4mpia.github.io/fitting_data/least-squares-fitting.html\
https://www.geeksforgeeks.org/python-tkinter-entry-widget/

https://datatofish.com/entry-box-tkinter/

https://www.codegrepper.com/code-examples/python/how+to+add+a+thing+to+a+textbox+in+tkinter+python

https://stackoverflow.com/questions/8384737/extract-file-name-from-path-no-matter-what-the-os-path-format

https://stackoverflow.com/questions/21739336/positioning-a-textbox-on-tkinter

https://stackoverflow.com/questions/58009398/tkinter-what-is-a-window-when-calling-tk-canvas-create-window

https://www.i-programmer.info/programming/python/5105-creating-the-python-ui-with-tkinter-the-canvas-widget.html?start=1

https://www.w3schools.com/python/python_variables_global.asp

https://www.digitalocean.com/community/tutorials/how-to-convert-data-types-in-python-3

https://stackoverflow.com/questions/4289331/how-to-extract-numbers-from-a-string-in-python

https://www.tutorialspoint.com/How-to-extract-floating-number-from-text-using-Python-regular-expression

https://docs.python.org/3/library/re.html

https://regex101.com/r/kKz9oo/1

https://stackoverflow.com/questions/904746/how-to-remove-all-characters-after-a-specific-character-in-python

https://www.kite.com/python/answers/how-to-remove-everything-after-a-character-in-a-string-in-python

https://www.geeksforgeeks.org/python-split-multiple-characters-from-string/

https://stackoverflow.com/questions/16046743/how-to-change-tkinter-button-state-from-disabled-to-normal

https://stackoverflow.com/questions/53580507/disable-enable-button-in-tkinter/53580612

https://www.pythontutorial.net/python-basics/python-read-text-file/

https://docs.python.org/3/library/os.html?highlight=os%20module#module-os

https://stackoverflow.com/questions/38428593/getting-the-absolute-position-of-cursor-in-tkinter