PHYS-4007/5007: Computational Physics

Python Tutorial
Making Plots of Spectra in Python
Second Tutorial

1 Introduction: Review of First Tutorial

In the first tutorial on plotting spectra, you copied and pasted program "specplot.py" from the Python web page linked to the course web page. Then from a terminal window, you ran this code from the Linux prompt with the command:

```
python3 specplot.py sun981.wvc
```

where 'sun981.wvc' was one of the spectrum data files that you downloaded from the above mentioned web page. As a reminder, the spectra that you downloaded were of the following stars and wavelength regimes:

Filename	**Star**	**Wavelength Area**
sun981.wvc	The Sun	Ca II H & K lines
sun997.wvc	The Sun	Ca II IR-triplet
betleo634.wvc	β Leo	Ca II H & K lines
betleo540.wvc	β Leo	Hα
betleo543.wvc	β Leo	Ca II IR-triplet
alpboo638.wvc	α Boo	Ca II H & K lines
alpboo560.wvc	α Boo	Ca II IR-triplet
delvir636.wvc	δ Vir	Ca II H & K lines
uuaur620.wvc	UU Aur	Ca II IR-triplet

Please review Section 2 of the first tutorial on spectrum plotting in Python on the contents of the original specplot.py program.

2 Modifying the specplot.py Python Code

At the Linux prompt, make a copy of this code with the following command:

```
cp specplot.py specplot2.py
```

Now edit this new file with emacs:

emacs specplot2.py &

remember that the ampersand ('&') symbol places this emacs session in background mode. In this specplot2.py code add the lines below marked with '**(new)**' at the end of the line.

```
# This program will read in one of my spectrum files obtained
# with the McMath-Pierce Solar Telescope on Kitt Peak, AZ and
# make a plot of the spectrum.  In the future, I will add a
# line ID function and an equivalent width of a spectral line
# function.  This program will also investigate the use of
# the Matplotlib Pyplot savefig function to make hardcopy
# files of the plots without going through the interactive
# GUI.

# When running this program, one needs to pass the name of
# the spectrum data file (in this example 'sun981.wvc' to be
# read and plotted:
#       python3 specplot2 sun981.wvc

# Always include these next import commands.

import sys
import numpy as np
import matplotlib.pyplot as plt

# Retrieve the data filename and make a root-name for the
# plot output files.  Then make an array of plot output
# filenames.  Assume a maximum of 20 plot files.

nfmax = 20  # Maximum number of encapsulated postscript files.
iplt = 0    # Array index of current eps plot to be made (new).

if (len(sys.argv) == 2):
        spcname = sys.argv[1]

iperiod = spcname.find('.')
rootname = spcname[0:iperiod]
fsuffix = '.eps'    # Assume encapsulated postscript files.

# Make filenames for the postscript plot files.
plname = []
```

```
for i in range(0, nfmax):
    plname.append(rootname+'p'+str(i).zfill(2)+fsuffix)  # (new) - add +'p'

# Read in the data from the spectrum file.

rdline = ''
starname = ''

fspc = open(spcname, 'r')

# Look for the name of the object that was observed.

while rdline != '  \n':
    rdline = fspc.readline()
    qstar = rdline.find('Star:')
    if qstar > -1:
        ieql = rdline.find(' = ')
        if ieql > 0:
            starname = rdline[7:ieql]

rdline = ''
sodate = ''
snccd = ''

# Look for the obs date, CCD #, and number of data points in the spectrum.

while rdline != '  \n':
    rdline = fspc.readline()
    qnpix = rdline.find('Number of pixels:')
    qdate = rdline.find('Obs-Date:')
    qccdn = rdline.find('CCD Picture Number:')
    slen = len(rdline)
    if qnpix > -1:
        snp = rdline[19:slen]
        np = int(snp)
    if qccdn > -1:
        snccd = rdline[qccdn+20:slen-1]  # (new) - add `-1' after slen
    if qdate > -1:
        sodate = rdline[11:22]

# Look for the number of data points in the spectrum.

rdline = ''
sfocus = ''
```

```
while rdline != '  \n':
    rdline = fspc.readline()
    qfocus = rdline.find('Telescope focus:')
    if qfocus > -1:
        sfocus = rdline[24:31]
        fwhm = float(sfocus)

rdline = ''

while rdline != '> Wavelength (Angstroms)\n':
    rdline = fspc.readline()

# Read in the wavelength data.

qaduflux = -1
swave = ''

while qaduflux == -1:
    rdline = fspc.readline()
    qaduflux = rdline.find('ADU-Flux:')
    if qaduflux == -1:
        swave = swave + rdline

# Read in the flux data.

qston = -1
sflux = ''

while qston == -1:
    rdline = fspc.readline()
    qston = rdline.find('Signal-to-Noise:')
    if qston == -1:
        sflux = sflux + rdline

fspc.close()

print('Object observed: "'+starname+'"')
print('Date of Observation: '+sodate)
print('CCD Picture #: '+snccd)
print('Number of pixels in spectrum: ', np)
print('Telescope FWHM: '+sfocus+' Angstroms')

# Convert wavelengths to floats.
```

```
sswave = swave.split()
wvlen = len(sswave)
wave = []

for i in range(0, wvlen):
    wave.append(float(sswave[i]))

# Convert fluxes to floats.

ssflux = sflux.split()
fllen = len(ssflux)
flux = []

for i in range(0, fllen):
    flux.append(float(ssflux[i]))

# Print the minimum and maximum of the wavelength axis.  (new)

wvmin = wave[0]        # (new)
wvmax = wave[np-1]     # (new)

print('\nMinimum wavelength of spectrum: ', wvmin, 'Angstroms')  # (new)
print('Maximum wavelength of spectrum: ', wvmax, 'Angstroms')    # (new)

# Show the spectrum.

plt.plot(wave, flux, 'k-')
plt.xlabel('Wavelength (A)', labelpad=10)
plt.ylabel('Flux (ADU)', labelpad=10)
plt.title(starname+' Observation')
plt.figtext(0.18, 0.8, 'CCD Picture #'+snccd)
plt.figtext(0.7, 0.83, sodate)

plt.savefig(plname[iplt], orientation='portrait', format='eps')  # (new)

plt.show()
```

Remember that any text written after the pound-sign (#) symbol tells Python that the following text is a comment. I will highlight each of the new lines and modified lines of this code verbally during the tutorial session.

3 Explore On Your Own

The new lines added to the specplot2.py code shows you how to make a code that will automatically save a plot in encapsulated postscript format. Such files then can easily be incorporated into a LaTeX file as explained in §III of the course notes. Now on your own, figure out how to modify this code so that one can make additional plots of a spectrum you have just made by adjusting the minimum and maximum wavelengths to plot for the spectrum (*i.e.*, expand or focus in on part of the spectrum). You will first need to increment the iplt counter parameter after the plt.show() command. Following this you should ask the user whether or not they wish to make a new plot. If so, ask the user to enter a new minimum and maximum wavelength (stored in the wvmin and wvmax parameters), then send the user back to the code just after the defining lines for wvmin and wvmax parameters. I leave it to you to figure out how to do this by searching the web with Google and examining the Python web pages (note that I have links to some of these web pages on the course web pages.