Python and IDL 2

Using the Bright Star Dataset: "BrightStars.dat"

Write an IDL or Python program that will create a new dataset "BrightStarsIDL.dat" or "BrightStarsPy.dat with the following 5 columns:

Catalog# RA DEC Distance Absolute Magnitude

With RA and DEC in decimal degrees (i.e 12.2345 34.2365) and Distance in light years.

RA and DEC should have 4 digits after the decimal (i.e 12.2345) Distance and AbMag should have 2 digits after the decimal (i.e 4.39)

Your rows should look something like:

BXXXX XX.XXXX XX.XXXX X.XX X.XX

Email To Me:

1) The lines of BrightStars[IDL or Py].dat for the 5 nearest stars.

2) The lines of BrightStars[IDL or Py].dat for the 5 absolute brightest stars.
You may use UNIX sort in 1) and 2).

3) Your IDL or Python program

Due: October 15 by 5pm

Hints and Help Files

Below is an outline of some hints and help files that will assist you in writing your program.

1. Get the help files.

IDL Programmers:

```
% cd Astro192
% wget http://www.astro.washington.edu/groups/premap/seminarfiles/idl.tar.gz
% tar zxmfv idl.tar.gz
% cd idl/
% emacs IDLNotes.txt &
```

You want a print out?

```
% lpr -p -PHP4200 IDLNotes.txt
```

Python Programmers:

```
% cd Astro192
% wget
http://www.astro.washington.edu/groups/premap/seminarfiles/python.tar.gz
% tar zxmfv python.tar.gz
% cd python
% emacs PythonNotes.txt &
```

You want a print out?

```
% lpr -p -PHP4200 PythonNotes.txt
```

2. Change to the directory with test.pro or test.py

3. Open test.pro or test.py in emacs

3. Go through the program to understand what it does. Try to translate it into words

4. Compile and run the program in the terminal.

IDL Programmers:

As you expect, you can write IDL programs to files. To use them in IDL you need to compile them with the ".compile" command in IDL.

```
IDL> .compile test.pro
% Compiled module: CONVERTRA.
% Compiled module: TEST.
IDL> test
% Compiled module: READCOL.
% Compiled module: REMCHAR.
% Compiled module: GETTOK.
% Compiled module: STRSPLIT.
% Compiled module: STRNUMBER.
% READCOL: 10 valid lines read
```

Python Programmers:
As you expect, you can write Python programs to files. Create a program by writing the commands in a text editor (like emacs). Save the file as programname.py. To use them type python porgramname.py in the command line.

```
% python test.py
```

5. Examine the output, where in test.pro or test.py does it write to a file?

6. Save test.pro or test.py as a new file (with the correct extension) in emacs

7. Now let's try to write some code.
Use the IDL> or python >>> interpreter to test equations etc.

Comments: For each function, make a comment that tells the user what variables (and their units) are fed into the function and what variable (and units) are returned.

Break the Program Into Parts

Like all programs, try to break the task up into its parts. The original BrightStars.dat has RA and Dec different units than what we want. There is no "distance" or "absolute magnitude" in Brightstars.dat, but there are parallax and magnitude. We'll need to use them to derive the distance and absolute magnitude.

The math stuff:
a. We will have to make some functions to convert RA and DEC units.
b. We will need to make a function to convert parallax to distance (in light years)
c. We will need to use magnitude and parallax to calculate absolute magnitude.

The programming stuff:
d. We need to read data from file and play around with it.
e. We need to write our new variables to a file with the proper formatting.

The checking (to do after you can successfully run the code):
f. Are all the units correct?
g. Can we do any sanity checks?

Now that we have divided up the problem, start by making sure you know the math that is needed. Just like how you don't write a paper without having an idea what you will write about, don't write a program before knowing what it's going to be about!

a. How do you convert RA and Dec. in hours, min, sec to decimals hours or degrees?
ra (in decimal degrees) = RAhours + RAmin/60. + RAsec/3600.
dec (in decimal degrees) = Decdeg + Decmin/60. + Decsec/3600.

b. The distance in parcsecs is just 1/parallax if parallax is in arcseconds. There are 3.2616 lightyears/parsec. *Hint: you'll need to multiply parsecs by 1e-3 since BrightStars.dat uses milli-arcseconds instead of arcseconds.*

c. Absolute magnitude is related to parallax (P) and apparent magnitude (mag) by:
AbsMag = mag + 5.*(Log10(P) + 1)
(careful, Log10 is log base 10.)

8. After writing the functions, write in the main part of the program a "call" to the functions you just wrote.

9. Edit the output section of the main program as the assignment requires.

IDL Formatting syntax:

`iN`	Integer value with up to N chars
`fN.M`	float of N chars with M chars after decimal
`aN`	string with N chars
`Nx`	skip N chars

Python Formatting syntax:

`%sN`	string with up to N chars
`%.Mf`	floating of M chars after decimal
`%iN`	integer with up to N chars

10. Compile and run your program.

11. Debug! Are you doing integer multiplication or division? Be sure you have 1./variable not 1/variable. Make sure all () are there, all math is done with floats or doubles, check syntax in functions and programs.

Use online tutorials, help(), your neighbors, and me! Remember, you may work together, but everyone must do their own work!

Python: test.py

```
from numpy import *

#
# define a function called convertra
#
def convertra(rah,ram,ras):
# Comment here to say what variables and
# their units come in and go out
    ra = rah + ram + ras
    return ra

# Main part of Program
# Comment here to explain what the point of this
# program is

# This opens the file '10.dat' to read
#
ifile = open("10.dat",'r')

#
# This opens the file 'output.txt' to write
#
ofile = open("output.txt", 'w')

for line in ifile:
# Look up line.split() in python help
    element = line.split()
    name = element[0]
    rah  = int(element[1])
    ram  = float(element[2])
    ras  = float(element[3])

    #
    # Do stuff with the data
    #

    ra = convertra(rah,ram,ras)

    #
    # Write stuff to the ofile (defined above)
    #

    ofile.writelines('%s %.4f %.1f %.2f %.2f \n' % (name, rah, ram, ras))

ifile.close()
ofile.close()
```

IDL: test.pro

```
FUNCTION convertra, rah,ram,ras
; Comment here to say what variables and
; their units come in and go out
    ra = rah + ram + ras
    return,ra
END

PRO test
; Comment here to explain what the point of this
; program is

;
; This reads in the file '10.dat'
; Look up readcol in your manual, idlhelp, or on the Internet

    readcol,'10.dat',$
    name,rah,ram,ras,$
    f='a,i,i,f'
;
; Do stuff with the data
;

    ra = convertra(rah,ram,ras)

;
; This is all for output to the file "output.txt"
; The only things you should change are the lines
; beginning with "format" and "name". You may change
: the name of the file output.txt too if you wish.
;
    limit = n_elements(name) - 1

    openw, lun, 'output.txt', /Get_lun
    for i=0,limit do printf,lun,$
        format='(a5,i9,i9,f9.2,f9.4)',$         ; change this line
        name[i],rah[i],ram[i],ras[i],ra[i]      ; change this line
    close,lun
    free_lun, lun
END
```