INTRO TO PYTHON

FOR AOSC

OCT 19, 2016

INTRO TO PYTHON

Day 2

- Arrays
- Plotting
- Module Loading

LET'S TRY OUR 1ST PROGRAM

- Copy ~jeff/aosc652/week08/hello.py

```
end = raw_input('Enter a number between 3 and 10: ')
while n < end:
        print('hello world: {0}'.format(n))
```

READING AND WRITING FILES

- Multiple Methods
  - Built-in functions
    - open(filename, mode)
    - open() combined with a GUI
  - Modules
    - Pandas
      - read_csv

READING AND WRITING FILES

- Reading in
  - Once open files are stored in an iterable variable
    - Text = open('filename.dat', 'r') # w, r+/w+, a, rb, wb, rb+/wb+
    - print(Text.next()) #Prints next line
    - Line_of_text = Text.next() #Stores the next line as a string variable
    - Line_of_text = Text.readline() #Stores the next line as a string variable
    - Full_text = Text.readlines() #Stores ALL lines as a list

READING AND WRITING FILES

- Writing Out
  - Open the file with a write mode
  - Text_out = open('Out_filename.dat', 'w') # r+/w+, a, wb, rb+/wb+
    - If you open with a 'w' or r+/w+ mode it will delete the current content
    - Use 'a' mode to preserve (append) existing content
  - Text_out('String') #Writes string to the file
  - Text_out('String blah blah blah') #writes more to file
  - Text_out.close() # closes file and finalizes write

CLOSE FILES

- VERY important to prevent corruption and memory leaks
  - Text_out.close()
  - Text.close()
  - Open for file as part of a with-as loop
    - This will auto-close the file

NUMPY ARRAYS

- Lists and Tuples can hold multiple data types
- NumPy Arrays: all data stored as same type
- 1-D, 2-D, ….X-D
  - No limit to the number of dimensions
  - A = np.array([4,89,-1])
  - A = np.array([4.89,-1], dtype=*type*) #Page 72
- 2-D
  - B = np.array([[23,41,3],[14,92,1]])

BUILDING SEQUENTIAL NUMPY ARRAYS

- import numpy as np

- np.arange(B, E, S) #Beginning, End*, Stride
  - np.arange(5, 20, 0.5)
- np.linspace(B, E, Num) #Beginning, End, Number of Points
  - np.linspace(1, 10, 20)
- np.logspace(B, E, Num) #Beginning (base10), End(Base10), Num
  - np.logspace(1,10,20)

ACCESSING NUMPY ARRAY DATA

- Start counting at 0
- End point non-inclusive
- : means all
- , separates dimensions

ACCESSING 1-D NUMPY ARRAY DATA

- 1-D array A
- A #All of A
- A[3] #Value at index 3
- A[:4] #Values from beginning to 4
- A[4:] #Values from 4 to end
- A[::-1] #All values reverse order

ACCESSING 2-D NUMPY ARRAY DATA

- 2-D array B
- B #All of B
- B[3,0] #Value at row 3 column 1
- A[:,4] #Column 4 vaules for ALL rows
- A[4:,3] #Column 3 value for rows 4 to end

SIZE AND SHAPE OF NUMPY ARRAYS

- shape(a) # returns dimensions as tuple
- size(a) # number of elements
- len(a) #size(a)

1-D PLOTTING

import matplotlib.pyplot as plt

- plt.plot(x, y, c = 'g', ls = '--', marker = 's')
- plt.semilogx(), plt.semilogy(), plt.loglog()
- plt.xlabels('string'), plt.ylabels('string'), plt.title('string')

MODIFICATIONS FOR CUSTOM MODULES

- Edit your .cshrc file and add this line…
  - setenv PYTHONPATH $HOME/aosc652:$HOME/pythonModules
- Make a directory for custom python modules
  - mkdir ~/pythonModules
- Copy my LHD module
  - Cp ~jeff/pythonModules/LDH.py ~/pythonModules/.
- Logout or "source ~/.cshrc"