LECTURE 6: FILE I/O, STRINGS

Introduction to Scientific Python, CME 193
Feb. 12, 2014
Please download today's exercises from:
`web.stanford.edu/~ermartin/Teaching/CME193-Winter15`
Eileen Martin

Some slides are from Sven Schmit's Fall '14 slides

Feedback

- Survey at end of class today

- If you want something changed, say so!
  - Talk to me
  - Email
  - Anonymous online survey:

    https://www.surveymonkey.com/s/NSVJDDJ

Overview Today

- Some Numpy & Matplotlib I/O
- Strings
- Basic file I/O
- Overview of more useful modules
- Course survey

Save and load in .npy or .txt format

- You can save/load a numpy array to/from a binary file (.npy) or text file (.txt).
- Example: saveArray.py, loadArray.py

```
import numpy as np

a = np.linspace(0, 1, 10000) # create a numpy array
np.save('aArray.npy', a) # save a binary copy of the array
np.savetxt('aArray.txt', a) # save a text file copy of array
```

```
import numpy as np

np.load('aArray.npy', a_npy) # load the array from binary file
np.loadtxt('aArray.txt', a_txt) # load the array from text file
```

Save/load .png image with Matplotlib

Run smile_image.py:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

# read in image and convert to numpy array, then plot
smileArray = mpimg.imread(‘smiley_face.png’)
print(smileArray)
smilePlot = plt.imshow(smileArray)
plt.show()

# create gray scale image from red channel
redArray = smileArray[:,:,0]
mpimg.imsave(‘smiley_face2.png’,redArray) # save just red
```

More on this in image tutorial: http://matplotlib.org/users/image_tutorial.html

Overview Today

- Some Numpy & Matplotlib I/O
- Strings
- Basic file I/O
- Overview of more useful modules
- Course survey

Strings review

- Contained in single, double or triple quotes
- Immutable objects
- Can be iterated over
- Can be indexed
- You can write a __str__ method to represent any class you define as a string

String formatting

- Special characters:

notation	\n	\t	\b
meaning	new line	tab	backspace

- Add variables or use `format`:

notation	%s	%f	%e	%g	%d
type	string	real float	exponential float	shorter version of float	signed integer

```
fl = 0.235
wo = ‘Hello’
inte = 12
print(“s: {} \t f: {:0.2f} \n i: {}”.format(wo,fl,inte))
# s: Hello    f: 0.23
# i: 12
```

Complete list of format symbols: http://www.tutorialspoint.com/python/python_strings.htm

Split a string

- You can split a string either by spaces and new lines (default) or by a specified delimiter, and return a list of words

```
text = ‘Hello, world!\n How are you?’
splitTxt = text.split()
# splitTxt is now [‘Hello,’, ‘world!’, ‘How’, ‘are’, ‘you?’]

numbers = ‘1, 3, 2, 5’
splitNums = numbers.split()
# splitNums is now [‘1,’, ‘3,’, ‘2,’ ‘5’]
splitNums2 = numbers.split(‘, ‘)
# splitNums2 is now [‘1’, ‘3’, ‘2’, ‘5’]

intNums = [int(i) for i in numbers.split(‘, ‘)]
# intNums is now [1, 3, 2, 5]
```

Upper case, lower case, many more functions

- You can find all functions available for your string use `dir`

```
text = ‘Hello, world!\n How are you?
t = text.upper()
# t is 'HELLO, WORLD!\n HOW ARE YOU?'
t = text.lower()
# t is 'hello, world!\n how are you?'
print(dir(text))
['__add__', '__class__', '__contains__', '__delattr__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__',
'__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__',
'__rmul__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', '_formatter_field_name_split',
'_formatter_parser', 'capitalize', 'center', 'count', 'decode',
'encode', 'endswith', 'expandtabs', 'find', 'format', 'index',
'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle',
'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

Joining strings

- The join function takes a list of strings and puts them together with some specified string between them:

```
myList = [‘hello’,
‘world’]
t = ‘‘.join(myList)
# t is ‘helloworld’

t = ‘ ‘.join(myList)
# t is ‘hello world’

t = ‘, ‘.join(myList)
# t is ‘hello, world’
```

Overview Today

- Some Numpy & Matplotlib I/O
- Strings
- Basic file I/O
- Overview of more useful modules
- Course survey

To start: create a file object

- Python interacts with files with file objects
- Instantiate a file object with `open` or `file`

```
f = open(filename, option)
```

- At the end, use `f.close()`
- filename is a string containing the location of a file
- option can be:
  - `r` to read a file (default option)
  - `w` to write a file
  - `a` to append to a file
  - `r+` to read and write
  - `rb` for binary reading
  - `wb` for binary writing

Reading:

- The file object keeps track of its current location
- Example in first few lines of reader.py
- Some useful commands:
  - `read()` Read next line by default in ‘r’ mode
    - Read next n characters if argument is there in ‘r’ mode
    - Read next n bytes in ‘rb’ mode
  - `readline()` Reads a single line per call
  - `readlines()` Returns a list of lines (splits at newline)

```
with open('text_file.txt', 'r') as f:
   for line in f:
       # count how many a’s are in each line
      acount = line.count('a')
      print("This line has "+str(acount)+" a's")
```

with open() as myFileObject:

- You can use this statement to keep a file open throughout a block of code, and it will automatically close at the end
- Example in last couple lines of reader.py:

```
with open(‘text_file.txt’, ‘r’) as f:
        print f.read()
```

Writing

- Use write() to write to a file
- See example in writer.py

```
f = open('new_text_file.txt','w') # open file object to write
f.write("Today's high is %u degrees.\n" % 75)
for i in range(3):
        f.write("I'm writing the number "+str(i)+"\n")
f.close() # close file object
```

Useful tools & tips for I/O with binary files

- When you read and write, use the struct module's pack and unpack function

  https://docs.python.org/2/library/struct.html
- When you unpack, you must take the [0] element of a tuple
- Use seek function to move to a particular location for reading/writing
- If you get nonsense, try swapping byte-order (little/big endian denoted by >, <, @, !,=)
- See example in binwriter.py (run first) and binread.py

Overview Today

- Some Numpy & Matplotlib I/O
- Strings
- Basic file I/O
- Overview of more useful modules
- Course survey

Some useful I/O related modules

- zlib and gzip for compressed gzip files
- bz2 for compressed bzip2 files
- zipfile for compressed zip archives > 4 GB in size, can decrypt slowly when reading, but can't encrypt when writing
- tarfile for tar archives (including gzip and bz2)
- csv for comma separated value files (databases)
- ConfigParser for default style configuration files
- robotparser for web crawler applications
- io for handling streams (useful for binary)
- HTMLParser for HTML files

Overview Today

- Some Numpy & Matplotlib I/O
- Strings
- Basic file I/O
- Overview of more useful modules
- Course survey

Course survey

- Don’t write your name on it
- When you’re done, please leave your survey on the desk in front.