03-0 Reading and Writing Data Files

CSI 500

Spring 2018

Course material derived from:
Downey, Allen B. 2012. "Think Python, 2nd Edition". O'Reilly Media Inc., Sebastopol CA.

"How to Think Like a Computer Scientist" by Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers. Oct 2012
http://openbookproject.net/thinkcs/python/english3e/index.html

Reading and writing files

- Python is often used to "wrangle" data
  - easy to read in data
  - easy to update
  - easy to write out new data
- Let's try out some examples
  - Create a small text file with data
  - Save it in the folder where you keep your Python source code files

```
Humpty Dumpty sat on a wall
Humpty Dumpty had a great fall
All the king's horses and all the king's men
Couldn't put Humpty together again
```

Reading a text file

- Type in the following code to read the file and display line by line
- Notice open() function syntax
  - file_handle = open(name, mode, options)
- Notice we read the entire file at once
  - data = infile.readlines()
- Notice we close the input file when done
  - good programming practice

```
"""
read file example 1.py
demo of how to read and write a file
"""

# read in Humpty Dumpty file
infile = open('Humpty.txt', mode='r', encoding=None)
data = infile.readlines()
infile.close()

# print out line by line
i = 0
for line in data:
    print('line[%4d] = %s' % (i, line))
    i += 1
```

Here's what it looks like

- Notice the extra <cr> after each line
- this is OS-specific: on Windows, it's the default
- to fix, we need to trim off the trailing <cr> after each line with the strip() function

```
line[   0] = Humpty Dumpty sat on a wall

line[   1] = Humpty Dumpty had a great fall

line[   2] = All the king's horses and all the king's men

line[   3] = Couldn't put Humpty together again

```

Reading a text file again

- Type in the following code to read the file and display line by line
  - we've added a call to the strip() function in the print() statement
  - this will trim the data for printing
  - does not affect value of "line" variable

```
"""
read file example 1.py
demo of how to read and write a file
"""

# read in Humpty Dumpty file
infile = open('Humpty.txt', mode='r', encoding=None)
data = infile.readlines()
infile.close()

# print out line by line
i = 0
for line in data:
    print('line[%4d] = %s' % (i, line.strip()))
    i += 1
```

Here's what it looks like

- Notice the extra <cr> after each line is now gone

```
line[   0] = Humpty Dumpty sat on a wall
line[   1] = Humpty Dumpty had a great fall
line[   2] = All the king's horses and all the king's men
line[   3] = Couldn't put Humpty together again
```

Processing lines of text

- What if we want to extract individual words from the line of text?
- We need to use the split() function
- Type in the following code
  - read the file
  - process each line
  - count the number of times we see 'Humpty' or 'Dumpty'

```
"""
read file example 1.py
demo of how to read and write a file
"""

# read in Humpty Dumpty file
infile = open('Humpty.txt', mode='r', encoding=None)
data = infile.readlines()
infile.close()

# print out line by line
i = 0
count = 0
for line in data:
    line = line.strip()
    words = line.split(' ')
    for word in words:
        if word == 'Humpty':
            count += 1
        if word == 'Dumpty':
            count += 1
    # print('line[%4d] = %s' % (i, line.strip()))
    i += 1

print('Humpty or Dumpty appeared %d times'%(count))
```

Here's what it looks like

- We read each word in each line
- We matched on specific words
- We got the count of terms of interest

```
Humpty or Dumpty appeared 5 times
```

Writing a file

- Now let's reverse the process and write a file
  - Notice text contains newlines \n
  - Notice text file contains line continuations \
- Note the open() syntax for writing
  - file_handle = open(file, mode, options)
- Note we've called flush() and close() to cleanly finish writing

```
"""
read file example 1.py
demo of how to read and write a file
"""

# write out Miss Muffet
text = 'Little Miss Muffet\n\
Sat on a tuffet\n\
Eating her curds and whey\n\
When along came a spider\n\
And sat down beside her\n\
And frightened Miss Muffet away'

# open file for writing
outfile = open('Muffet.txt', mode='w')

# write the contents to the file
for line in text:
    outfile.writelines( line )

# all done, flush an close
outfile.flush()
outfile.close()
```

Here's what it looks like

- Notice the <cr> provide line breaks

```
Little Miss Muffet
Sat on a tuffet
Eating her curds and whey
When along came a spider
And sat down beside her
And frightened Miss Muffet away
```

Reading a data file

- Python can be used to process data files, such as Comma Separated Value (CSV) files
- Let's try a small example
  - type in the following data into a file named medals.csv

```
country, gold, silver, bronze
Norway, 13, 14, 11
Canada, 11, 8, 10
Germany, 13, 8, 7
United States, 9, 8, 6
Netherlands, 8, 6, 6
```

Reading a data file

- Now type in the following (you may use the same file as before or create a new one)
- Read the first line using readline()
  - it holds column headers
- Read all the rest using readlines()
- Use the split() function to parse data
  - data is returned as String type
  - must convert to use as Integer

```
# read in medals.csv file
  infile = open('medal.csv', mode='r', encoding=None)
  header = infile.readline()
  data = infile.readlines()
  infile.close()

  # print the header
  header = header.upper()
  country,gold,silver,bronze = header.split(',')
  print('%20s %8s %8s %8s'%(country,gold,silver,bronze))

  # print the data
  for line in data:
     line = line.strip()
     country,gold,silver,bronze = line.split(',')
     print('%20s %8s %8s %8s'%(country,gold,silver,bronze))
```

Here's what it looks like

- Notice the formatting is not super clean
- In practice, you would probably use the data for computations, not just echo it back to the user

```
COUNTRY      GOLD   SILVER  BRONZE

          Norway      13      14      11
          Canada     11       8      10
         Germany       13       8       7
     United States       9       8       6
      Netherlands       8       6       6
```

Writing a data file

- Python can be also be used to create data files, such as Comma Separated Value (CSV) files
- Let's try a small example
  - let's use our medals data, but let's add some additional sports statistics
  - total medal count
  - percent of gold, silver, and bronze medals

```
country, gold, silver, bronze
Norway, 13, 14, 11
Canada, 11, 8, 10
Germany, 13, 8, 7
United States, 9, 8, 6
Netherlands, 8, 6, 6
```

Writing a data file

- Now type in the following (you may use the same file as before or create a new one)
- Read the medal file header and data
- Open a new output file
- Create a new header for the output file

```
# read in medals.csv file
  infile = open('medal.csv', mode='r', encoding=None)
  header = infile.readline()
  data = infile.readlines()
  infile.close()

  # open new target CSV file
  outfile = open('medal_stats.csv', mode='w')

  # process new header row
  new_header = 'country,gold,silver,bronze,total,\
  pct_gold,pct_silver,pct_bronze'
  outfile.writelines( new_header + '\n')
```

Writing a data file

- Now process each line of input
  - extract country, gold,silver,bronze
  - convert to int data type
  - do computations for our results
  - create a line for output file
  - write new line to output file
- When all done, flush() and close()

```
# process the data
  for line in data:
     line = line.strip()
     country,gold,silver,bronze = line.split(',')

     # convert to numeric types
     gold = int(gold)
     silver = int(silver)
     bronze = int(bronze)

     # do computations
     total = gold + silver + bronze
     pct_gold = gold / total
     pct_silver = silver / total
     pct_bronze = bronze / total

     # create output message
     msg = ('%s,%d,%d,%d,%d,%8.4f,%8.4f,%8.4f'%\
     (country,gold,silver,bronze,\
     total,pct_gold,pct_silver,pct_bronze))

     # write to file
     outfile.writelines(msg + '\n')

  outfile.flush()
  outfile.close()
```

Here's what it looks like

- Notice the formatting standard CSV
- This new file can be read by your favorite statistical processing software

```
 country,gold,silver,bronze,total,pct-gold,pct-silver,pct-bronze
Norway,13,14,11,38,  0.3421,  0.3684,  0.2895
Canada,11,8,10,29,  0.3793,  0.2759,  0.3448
Germany,13,8,7,28,  0.4643,  0.2857,  0.2500
United States,9,8,6,23,  0.3913,  0.3478,  0.2609
Netherlands,8,6,6,20,  0.4000,  0.3000,  0.3000
```

Summary

- Python supports reading and writing of text and data files
  - The open() function is used to open a file for reading or writing
  - The flush() function clears any pending I/O buffers
  - The close() function closes the file
- The file object has functions to read a single line via readline() or multiple lines via readlines()
  - Be careful to remove unwanted <cr> line-feeds from input lines
- The file object has functions to write data via writeline().
  - Be careful to add <cr> line-feeds to output lines