CSC 111 - Python

Lab 11 Slides

Review: open Function

```
open( 'filename.ext', 'rb' )
```

Keyword

Path to file

from current folder

Mode

- Make sure to save the returned `file` object

```
f_in = open( 'input.txt', 'rb' )
f_out = open( 'output.txt', 'wb' )
```

with Statement

A very nice way to save the file object:

```
with open('input.txt', 'rb') as fin:
    for line in fin:
        print line
# done with fin, file closed for you
```

`open` Modes

Mode	Use	Clears file?
`'rb'`	Reading Only	Does not clear file
`'wb'`	Writing Only	Clears file
`'ab'`	Append (Writing Only)	Does not clear file
`'rb+'`	Read & Write	Does not clear file
`'wb+'`	Read & Write	Clears file

Writing Files

```
with open('out.txt', 'wb') as f_out:
    f_out.write('Line 1\n')
```

- Unlike `print`, `.write()` does not add a newline character
- You must add each newline (`'\n'`) yourself

CSV Files

Structured Files

- Structured files allow representation of specialty files as text
- **`csv`** (**c**omma **s**eparated **v**alue) files are a common format for spreadsheets

`csv` Files

How `csv` files are stored:

```
Name,Test1,Test2,Test3
Ila,1,67,64
Xavier,83,11,54
Phillip,39,95,16
```

- Items separated by commas
- Rows separated by newlines

csv Files

What **csv** files represent:

	A	B	C	D
1	Name	Test1	Test2	Test3
2	Ila	1	67	64
3	Xavier	83	11	54
4	Phillip	39	95	16

- What we're used to seeing
- Same thing, nicer formatting

`csv` Read/Write Concepts

- `csv` module allows specialized file objects
- `reader` objects
  - Read in row string and strip it of whitespace
  - Split on commas into a list for unpacking
- `writer` objects
  - Join row list with commas
  - Adds newline character to separate rows

csv Reader Object

```
import csv
with open('grades.csv') as f_in:
        reader = csv.reader( f_in )
```

- create **reader** from **f_in** → holds a list of rows, where each row is a list of values
- We can unpack this list into variables

csv Reading (Incomplete)

```
with open( 'data.csv' ) as fin:
    reader = csv.reader( fin )
    for row in reader:
        print row[0]
```

```
Name
Ila
Xavier
Phillip
```

`row[0]` isn't a helpful variable name

csv Reading (Complete)

```
with open( 'data.csv' ) as fin:
    reader = csv.reader( fin )
    for name, t1, t2, t3 in reader:
        print name
```

```
Name
Ila
Xavier
Phillip
```

name is a more helpful variable name

csv Reading & Math (1)

```
with open( 'data.csv' ) as fin:
    reader = csv.reader( fin )
    for name, t1, t2, t3 in reader:
        total = t1 + t2 + t3
        print total
```

```
Test1Test2Test3
16764
831154
399516
```

csv data is read as strings

csv Reading & Math (2)

```
with open( 'data.csv' ) as fin:
    reader = csv.reader( fin )
    for name, t1, t2, t3 in reader:
        total = float(t1) + float(…
         print total
```

Convert to **int** or **float**

csv Reading & Math (3)

```
with open( 'data.csv' ) as fin:
    reader = csv.reader( fin )
    for name, t1, t2, t3 in reader:
        total = float(t1) + float(…
        print total
ValueError:
invalid for float(): 'Test1'
```

But the header line gives us problems

csv Reading & Math (4)

```
with open( 'data.csv' ) as fin:
    header = fin.readline()
    reader = csv.reader(fin)
    for name, t1, t2, t3 in reader:
        total = float(t1) + float(…
        print total
```

So let's skip the header line

csv Reading & Math (5)

```
with open( 'data.csv' ) as fin:
    header = fin.readline()
    reader = csv.reader(fin)
    for name, t1, t2, t3 in reader:
        total = float(t1) + float(…
        print total
132.0
148.0
150.0
```

Now it works!

csv Writing

```
h=['Name', 'Test1', 'Test2', 'Test3']
rows=[ ['Ila', 1, 67, 64],
       ['Xavier', 83, 11, 54] ]
with open('data.csv', 'wb') as f_out:
    writer = csv.writer(f_out)
    writer.writerow(h)
    for row in rows:
        writer.writerow(row)
```

`writerow()` takes in a list