CSV MODULE

While this week's work focuses on three primary types of data storage, flat text, JSON, and Python's pickle, there is a wide world of meaningful/accessible data types available to Python practitioners. Another data format (different from the three we used in class) is CSV. CSV which stands for Comma Separated Values is a simple format for storing tabular data in text files. You can compare csv files with Microsoft Excel spreadsheets. You may have already used these files without knowing. Although considered a simple format structure to use, many only use it as the last option when JSON or YAML aren't available, especially for large datasets. JSON (JavaScript Object Notation) & YAML (Yet Another Markup Language) are another two data structure options formats. A strength that CSV module can provide is that it is simple to use and familiar because of its similarity to Excel which many use in recent days. It can be a good option when working with data exported from spreadsheets and databases for data interchange. (http://pymotw.com/2/csv/)

Reading a csv file:

In order to use the csv module you must import csv and use arguments to read the data files being transferred.

```
>>> import csv                      #imports the csv module
>>> myfile = open(filename, 'r')    #opens the csv file
>>> reader = csv.reader(myfile)     #creates the reader object
>>> for row in reader:              #iterates the rows of the file in orders
    >>> print row                  #prints each row
>>> f.close()                       #closes file
```

NOTE: Remember that every file opened to be read, must be closed at the end in order to avoid future issues with the program.

CSV uses statements such as 'r' (reader()) to read files. It as well uses what is called a 'delimiter' to hold a place for spaces, tabs, commas, etc…

(Example: spamreader = csv.reader(csvfile, delimiter=' ')

You can use as well statements such as 'next'(reader) to skip first lines in a csv file.

Writing a csv file:

To write a CSV file you can use the 'w' writer () statement. This will allow the use to begin make to add their writing.

```
>>> fhandler = open('test-write.txt', 'w')
>>> fhandler.write('This is the first line')
>>> fhandler.close()
```

NOTE: Remember that every file opened to be read, must be closed at the end in order to avoid future issues with the program.

A weaknesses is that there is no "CSV standard". What this means is that csv is usually formatted dependent on the application that is being used to read and write it. Users would need to ensure that parameters are defined clearly so that no mistakes or errors are encountered, otherwise data produced & consumed may be different than what is expected. With clear and efficient programming it is easy to manipulate the module and have a clean program.