Reading to and Writing from Files: A Quick Primer

To read from a data file, you need to open the file using a file name and the option 'r' for read. For example, let's say we have the following data file named `months.txt` containing text:

```
January
February
March
April
```

To open the file for reading, we would execute the command:

```
inputfile = open("months.txt", 'r')
```

The variable `inputfile` represents the contents of the file. Once the file is open, we can read each line, one at a time using a loop. In the example below, we create a list of the months that are in the `inputfile`:

```
monthlist = []
for line in inputfile:
    monthlist.append(line)
```

Note that each line of the file is a string and the line included the newline character (`'\n'`). So if we run the code above, the `monthlist` would be:

```
['January\n', 'February\n', 'March\n', 'April\n']
```

If a line is blank in the file, then the loop above will read the blank line as `'\n'` (i.e. only a newline character).

We probably don't want to store an extra newline with each data item, so we can eliminate the last character by appending `line[0:len(line)-1]` or `line[0:-1]`) instead of `line`. This will append everything in the string except the last character, which eliminates the newlines:

```
monthlist = []
for line in inputfile:
    monthlist.append(line[0:-1])
```

Now `monthlist` is:

```
['January', 'February', 'March', 'April']
```

Now let's see how to output the contents of a list to a file. Say we have a list daylist stored as follows:

```
daylist = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

To write to a file, we open up a text file for writing with the option `'w'` (for writing):

```
outputfile = open("days.txt", 'w')
```

Then we can use a loop to write each day into the file, one per line:

```
for day in daylist:
    outputfile.write(day + '\n')
```

Note that we need to include an extra newline for each day so that each day is written on its own line in the output file.

Be careful: If you open a file for writing and the file exists already, it will be erased!

By default, input from a file gives you a string, and you have to write strings to files. If you want to read an integer from a file, you read it in as a string, remove the newline and then convert it to an integer using the `int()` function. If you want to store an integer in a file, you need to convert it to a string first using the `str()` function and then you can write that to the output file along with a newline.

For example, let's say our input file `measurements.txt` has the following integers:

```
18
32
91
4
-75
42
```

Then to read these into a list called `datalist`, we can use the following code:

```
inputfile = open("measurements.txt", 'r')
datalist = []
for line in inputfile:
    datalist.append(int(line[0:-1]))
```

This yields the `datalist` :

```
[18, 32, 91, 4, -75, 42]
```

Likewise, to write this `datalist` to the output file `scores.txt`, we can use the following code:

```
outputfile = open("scores.txt", 'w')
for value in datalist:
    outputfile.write(str(value)+'\n')
```

Remember: if scores.txt already existed, you would lose its original contents! Be careful!

Reading in multiple values per line

Let's say your file `numbers.txt` has multiple values per line, like this:

```
1, 2, 3, 4
5
6, 7
8, 9, 10
```

How do you read in these values individually? If you just read in line by line, you get strings that contain everything in each line together, not as separate values. But there is a string function `split` that takes a string and splits it into a list based on a delimiter which is a special character that separates values. For the file above, the delimiter is the comma.

Here is a function that reads the file above and adds up all of the values in the file:

```
def addup(filename):
    datafile = open(filename, 'r')
    sum = 0
    for line in datafile:
        cleanstring = line[0:-1]   # remove the newline
        datalist = cleanstring.split(',')
        for value in datalist:
            sum = sum + int(value)
    return sum
```

Sample usage:

```
>>> addup("numbers.txt")
55
```