Today we are going to make Python read data from a text file.

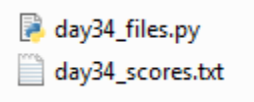
Download two files from the class worksheets/links folder on my web site:

day34_starter.txt
day34_scores.txt

Create a new Python program. Copy everything from day34_starter.txt and paste it into your new Python file. Save the file to your Z: drive named "day34_files.py" (Python automatically adds the ".py" so just type in the name and save it.)

Switch to the Windows Desktop. Create a folder on your Z: drive named "day34_files" and put the program you just created as well as the day34_scores.txt file into it.

To recap, you should have a folder on your Z: drive named day34_files, which now contains the following two files:



Edit the Python file. Put your name in the comment at the top.

Here is the code inside the Python file:

```
myFile=open("day34_scores.txt","r")
for line in myFile:
    print(line)
myFile.close()
```

This opens the text file "day34_scores.txt" and reads the lines out one at a time and prints them to the screen. These are fake scores for a pretend test. This code automatically reads from the file using a for loop until the file has no more data.

Note the first line that opens the file (the "r" means in "read mode") and the last line that closes the file.

Run it to see what happens. Here are a few lines of sample output:

```
Aaron,81.3

Abigail,94.1

Alex,75.4 ...
```

These are the names and scores from the text file.

(Continued on next page)

Add the following line before the print line to remove the line feed character from the end of each line. This tells Python to remove all line feeds (that is, to replace any line feeds with nothing.)

```
line=line.replace('\n',"")
```

Run it again and you get the same data without the extra lines in between:

```
Aaron,81.3
Abigail,94.1
Alex,75.4
...
```

We want to be able to access the scores and names separately, right now they are together in the variable "line" each time the loop repeats. To be able to access the score, we're going to use a new string method "split". Add this command to your program just after the replace line:

```
line=line.split(",")
```

This tells Python to take the string variable **line** and split it into a list variable using commas to split things up. Now when you run the program your printout should look like this:

```
['Aaron', '81.3']
['Abigail', '94.1']
['Alex', '75.4'] ...
```

This might seem worse to you, but we're making progress. Now we can access item 0 of the list (the name) or item 1 of the list (the score) independently. Why would we want to do this? Well, part of today's assignment is for you to read all of these scores and calculate the average. Once you've done a split, the variable **line** becomes a list, and you can access the name using line[0] and the score using line[1]. By default the score is a piece of text (a string), so you'd need to convert it into a float variable using the float() call before you try to add it to anything or find the average.

Your assignment for today:

> You are going to make a test score reporter:
>
> Make this program read all of the scores and find the overall class average. Also find and report the maximum score and the minimum score and report how many scores you processed. See the sample output below.

You'll want to make a total variable and set it to zero before the for loop. Then inside the for loop add each score to the total.

You'll also want a counter variable, so you know how many scores you have read from the file and added to the total variable. Again, create a count variable before the for loop and set it to zero. Then each time through the loop add one to the count variable.

To find the highest score or lowest score you have to again set up variable before the for loop. This time, for the max score, set the variable to zero, and for the minimum score, set the variable to 100. That's because the maximum possible score is 100, so if you set the variable to zero to start, you know you will find a score higher than that. Then, inside the for loop check each score to see if it is higher than your

previous maximum and if so, you have a new maximum. Do the same thing for the minimum. You also should set up variables to hold the names of the highest scorer and the lowest scorer.

When the for loop is over you can print out the average and the max and min data.

Sample output:

```
The following are scores for a class, followed by
the class average, the highest score, and the lowest score.

Processed 80 scores.
Class average: 74.02874999999999
Highest score: 99.8 by Emily
Lowest score: 50.1 by Taylor
```

Call me over to check you off when you are done.

Note: two people got the high score, not just Emily, so if your program reports the other person (Ryan), that's fine. If you want extra credit, make your program report both people's names using code, not by hard-wiring it in a print call. You'll have to figure out a way to keep track of multiple high scorers if you want to do this.

Also optionally for additional extra credit, make Turtle draw a graph of the scores. Here's what I did, you can do something totally different, just use each score to draw something and have them all integrate somehow.