New York University
School of Continuing and Professional Studies
Division of Programs in Information Technology

Introduction to Python
Homework, Session 3

3.1 Calculate the average Mkt-RF value (1st column of floats) for a given year. Take user input from the keyboard -- a 4-digit year. Reject the input with a polite/rude message if it is not all digits. Looping through Fama-French_data.txt file, calculate the average MktRF value (the 1st column of floating-point values, i.e. the 2nd column in the file) for that year.

Note: it will be much easier to use FF_abbreviated.txt for testing, as you can mentally calculate the correct amount and check it against your program's output. Once this is confirmed, you can then run the program against the Fama-French_data.txt file and compare your output to the sample output below.

Sample runs:

```
$ python myprog.py
please enter a 4-digit year:  hello
sorry, that was bad input
please enter a 4-digit year:  what's wrong?
sorry, that was bad input
please enter a 4-digit year:  1990
count 253, sum -12.77, avg -0.0504743083004

$ python myprog.py
please enter a 4-digit year:  1927
count 301, sum 26.26, avg 0.0872425249169

$ python myprog.py
please enter a 4-digit year:  1945
count 286, sum 32.55, avg 0.113811188811
```

3.2 wc emulation: wc is a unix utility program that counts the number of lines, words and characters in a given file. In the below example command run on a file in the Unix os, wc tells us that the FF_abbreviated.txt file contains 25 lines, 130 words and 1065 characters (note that your count may be off by 1 or 2 lines, or up to 20 characters -- don't sweat such differences):

```
$ wc ../python_data/FF_abbreviated.txt
      26     130    1066 FF_abbreviated.txt
```

Our script will do the same work: reading from file sawyer.txt (found in the source data directory linked from the class website), print the number of lines, words and characters in the file. Note: when counting characters, include spaces and newlines as well.

Challenge 1: please do not open and read the file more than once.

Challenge 2: you will be tempted to loop and count to get your answer, but you are challenged get these counts without looping. See if you can get each count without actually using a loop, but by using len() on various "slice and dices" in the data. Do this without opening the file more than once (hint: read() will read the file into a string; split() will split a string into words; splitlines() will split a string on the newlines!).

Extra credit: attempt to mirror the format of wc by right-justifying each value within an 8-character width (use str.rjust()).

```
please enter a filename:  ../python_data/sawyer.txt
      20     270    1435 sawyer.txt
```

Special note: we have seen anomalies that stem from varying line endings on the Windows and Mac platforms. You may get a count that is off mine by 1 or 2 lines, or up to 20 characters. If so, don't worry. It just needs to be within 20.

SUPPLEMENTARY (EXTRA CREDIT) EXERCISE

3.3 Write a Python program that can download CSV data from the internet and parse with the CSV module: real-world data is often more complex than the data we are working with, and supplementary exercises like these may be more complicated and/or time consuming than the "garden" assignments that are required. For these I will not work out solutions, so please enjoy the adventure and send me your questions.

CSV files need to be parsed with the CSV module, which takes into account some of these complexities (for example, including the delimeter in the data - what if you wanted your data to include a comma? Please see the new section titled "Modules: Internet Data" for details on using the CSV and urllib2 modules.

Select one of the sources listed at the link below - pick something interesting, and something you feel you might be able to learn from based on the (albeit simple) techniques we've described.

```
https://catalog.data.gov/dataset?res_format=CSV&page=1
```

The yellow [CSV] button under each source is actually a link to CSV data (in most cases), but we don't want to download the data to a browser, so don't click the button. When you've found data you'd like to analyze, right-click this yellow [CSV] button an select "Copy Link Location". (If you don't see that choice, move your cursor on and off, and try again.) Copy this link to a plain text file for use in your program.

For example, here's the link copied from the very first data source on that page (the Consumer Complaint Database):

```
https://data.consumerfinance.gov/api/views/s6ew-h6mp/rows.csv?accessType=DOWNLOAD
```

Now use your selected link (not necessarily the one above) with the urllib2 module to download the data of your choice from the website.

Remember that the handle you get back from urllib.urlopen() is readable in the same way a file is readable. Read the data and print the first few lines to see what you've got:

```
# my own urllib.urlopen() read handle I named rh:
lines = rh.readlines()
print(lines[0:4])
```

Then you can decide which values you'd like to analyze (at this point, we can only do sums and averages).

Send me your questions. Enjoy!