**Playing with a Text File**
**CS 5001 Intensive Foundations of Computer Science**

We are going to get away from graphics this week and switch to text manipulation. This is an extremely useful skill when it comes to data science or any type of data manipulation. I can't tell you how often I had to do it. To make it more interesting, you will accept the filename for the input file and output file and then convert one to the other. Researching lists & strings to learn about their built-in functions will be invaluable in this lab. Remember to work smarter not harder.

Any plain text file will do for testing, but here's a copy of a chapter from Alice in Wonderland I often use: https://www.dropbox.com/s/7svjmzk81cgsvq4/alice.txt?dl=0

1. **Goals**
    - Practice with lists
    - Explore string manipulation
    - Learn command line arguments
    - Learn file IO

2. **In Recitation**

    1. With the help of your TA start a project and build a frame for this project.
    2. Have your main accept three command-line arguments and store those in `in_file`, `out_file`, and `fun_call`
    3. Read in the text file into a list of strings
    4. Write out the file using the out_file variable
    5. Review the required functions for the assignment and ask any questions you have for your TA
    6. Watch as your TA demonstrates the final product

    Note: These starter assignments are meant to help you get started with each assignment by allowing you to code as a group and provide each other assistance. Make sure you understand everything that you code.

Submit the working program based on the above description and submit for your recitation ICE assignment.

### 3. Lab Assignment Instructions

Implement the following functions:

`main()` - checks to make sure the correct number of arguments were given, reads in the file, processes based on the function requested and creates the output. If a problem occurs, print a nice usage message explaining what functions are available and how to use the program.

`usage_msg()` - prints a usage message on how the program should be called and lists the available functions and what they do.

`all_upper_case(in_file, out_file)` - converts everything in in_file to upper case and saves it in out_file

`remove_a_word(in_file, out_file)` - gets as input a word or letter, removes it from the input file, and writes a new file with filename out_file

`reverse_file(in_file, out_file)` - reverses, by word not letter, in_file and writes it out as out_file

`pattern_count(in_file)` - gets as input a word or letter, and prints how many time that pattern occurs

`encode_file(in_file, out_file)` - Writes all the data out to the new file but shifts each letter forward by 1. You can do this simply by adding one to each value. Just keep it simple. Don't try to circle around alphabetically.Tip: Research the following functions: chr() and ord()

For example: 'a' -> 'b'

`decode_file(in_file, out_file)`- Reverses encode_file

Tips:

- Work smarter not harder.
- In most cases you are going to want to read in the file to a list, manipulate the items in the list, and then write the result back out
- Get basic functionality working first
- Backup as you go in case you get confused
- Build a frame first and then fill in each function
- You may create helper functions for repeated code
- You don't have to make the user enter the function names as you put them in code
- Read the directions carefully and review the rubric
- Test each function before submitting your final product

**4. Extension Suggestions:**

Remember you can get up to 26/30 points for finishing the lab, but for 30 points you have to go above and beyond. You don't have to use any of the below extensions. They are just examples. Explore on your own and come up with something fun.

Extension ideas:

1. Add a few more file manipulations of your own.
2. Come up with a different(very different and more complicated) encode/decode scheme
3. Add more feedback to your user
4. Test for other mistakes and providing the appropriate feedback to the user
5. Get "fancy" with the output
6. Separate out your code to increase functionality
7. Look for ways to create helper functions to decrease code

**5. Report**

With a more complicated assignment I'm satisfied with a shorter report, but make it easy for your grader.

Reflection:

What was the easiest and hardest part of this assignment?
What did you learn?
What grade would you give yourself?

Extension:

What extension(s) did you add to the assignment

**6. Submission:**

You should be submitting these at the least, but you may submit more files if you separated out your assignment for organizational purposes:

- Report05.pdf
- textFileManip.py

Submit your project code on canvas as Lab5_”Your_name”.zip

When you submit, double-check the following.

- ☐ Is your name and an appropriate header at the top of each Python file?
- ☐ Does every function have a comment or docstring specifying what it does?
- ☐ Does your report have all six sections completed?
- ☐ Is your report a pdf document?

**7. Rubric**

| | Possible | Given |
|---|---|---|
| Requested functions | | |
| main | 2 | 0 |
| all_upper_case() | 3 | 0 |
| remove_a_word() | 3 | 0 |
| reverse_file() | 3 | 0 |
| pattern_count() | 3 | 0 |
| encode()/decode() | 3 | |
| Error Handling | | |
| Usage Message | 1 | 0 |
| Correct number of arguments | 1 | 0 |
| Function name not found | 1 | 0 |
| Misc | | |
| Report (all or nothing) | 2 | 0 |
| Code Quality (correct indentation, comment blocks, variable naming, etc) | 4 | 0 |
| Not included in total possible: | | |
| Extensions (Not calculated without report) | 4 | 0 |
| Creative or went above and beyond | 4 | 0 |
| Code does not compile | -30 | 0 |
| Late penalty | -6 | 0 |
| Not implemented as requested | -30 | 0 |
| | | |
| TOTAL POINTS POSSIBLE out of 30 | 26 | 0 |