REPORT

Reflection:

1. What was the easiest and hardest part of this assignment?

   The hardest part of the assignment was just understanding how to run a program like this through the command line and what exactly sys.argv was doing. I'm used to making programs that use a menu, and because of this, it took me a while to plan out how this program would work and what it would look like. It was also challenging to keep track of what each method took as input and what each method returned. The easiest part of the assignment was learning how to use the methods like count, which required only simple functions to run. Additionally, many of the required functions could be built off of logic established in the previous functions, which helped speed things along.

2. What did you learn?

   I learned why it's so important to keep track of what each method takes and returns. Otherwise, your methods won't work, your functions won't be able to be called, and you'll create a debugging headache for yourself. I also learned the benefits of building a program for the command line: you can more easily work with external files and build a menu alternative much more quickly.

3. What grade would you give yourself?

   I would give myself a 27/30. I completed the basic function requirements and improved my code commenting to include parameters and return values. My extensions were pretty basic, so I wouldn't give myself more than one extra point there.

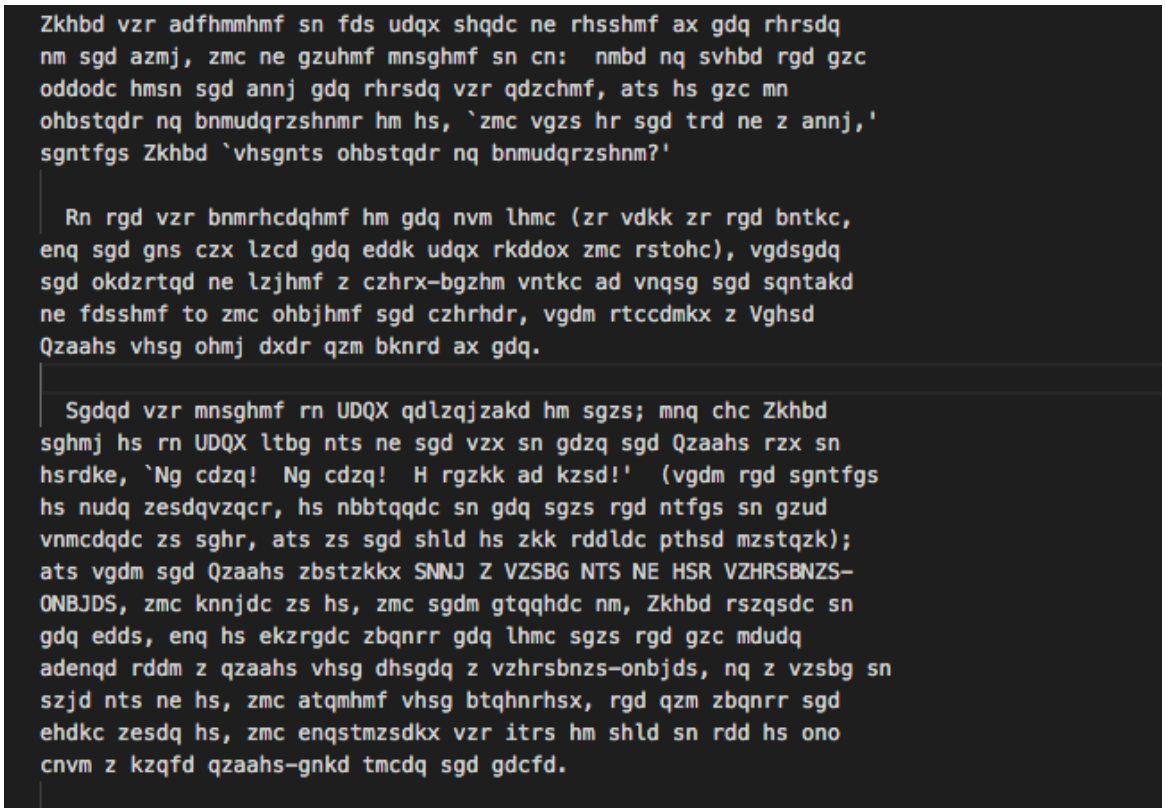
Extension:

1. What extension did you add to the assignment?

   Rather than having the user write the function name they want to use each time they run the program, I created logic that allowed the user to enter a number that corresponds to one of the function options instead. The function number options are explained when the program is run without any arguments in the usage message.

```
def main():
    if len(sys.argv) == 4:
        in_file = sys.argv[1]
        out_file = sys.argv[2]
        fun_call = sys.argv[3]
        if fun_call == "1":
            all_upper_case(in_file, out_file)
            print("you entered 1")
        elif fun_call == "2":
            input_to_remove = input("What would you like to remove?\n: ")
            remove_a_word(in_file,out_file,input_to_remove)
            print("you entered 2")
        elif fun_call == "3":
            reverse_file(in_file, out_file)
            print("you entered 3")
        elif fun_call == "4":
            pattern_count(in_file,input("What word or letter would you like to count?\n: "))
            print("you entered 4")
        elif fun_call == "5":
            encode_file(in_file, out_file)
            print("you entered 5")
        elif fun_call == "6":
            decode_file(out_file)
            print("you entered 6")
        else:
            print("The function options are 1, 2, 3, 4, 5, or 6.")
    else:
        usage_msg()
main()
```

I also learned about and used the writelines method to keep the text's format after it went through the functions. Otherwise, the text would be written into the output file as a single line. The writelines method writes a sequence of strings as a list and returns them on separate lines.



```
Zkhbd vzr adfhmmhmf sn fds udqx shqdc ne rhsshmf ax gdq rhrsdq
nm sgd azmj, zmc ne gzuhmf mnsghmf sn cn:  nmbd nq svhbd rgd gzc
oddodc hmsn sgd annj gdq rhrsdq vzr qdzchmf, ats hs gzc mn
ohbstqdr nq bnmudqrzshnmr hm hs, `zmc vgzs hr sgd trd ne z annj,'
sgntfgs Zkhbd `vhsgnts ohbstqdr nq bnmudqrzshnm?'

  Rn rgd vzr bnmrhcdqhmf hm gdq nvm lhmc (zr vdkk zr rgd bntkc,
enq sgd gns czx lzcd gdq eddk udqx rkddox zmc rstohc), vgdsgdq
sgd okdzrtqd ne lzjhmf z czhrx-bgzhm vntkc ad vnqsg sgd sqntakd
ne fdsshmf to zmc ohbjhmf sgd czhrhdr, vgdm rtccdmkx z Vghsd
Qzaahs vhsg ohmj dxdr qzm bknrd ax gdq.

  Sgdqd vzr mnsghmf rn UDQX qdlzqjzakd hm sgzs; mnq chc Zkhbd
sghmj hs rn UDQX ltbg nts ne sgd vzx sn gdzq sgd Qzaahs rzx sn
hsrdke, `Ng cdzq!  Ng cdzq!  H rgzkk ad kzsd!'  (vgdm rgd sgntfgs
hs nudq zesdqvzqcr, hs nbbtqqdc sn gdq sgzs rgd ntfgs sn gzud
vnmcdqdc zs sghr, ats zs sgd shld hs zkk rddldc pthsd mzstqzk);
ats vgdm sgd Qzaahs zbstzkkx SNNJ Z VZSBG NTS NE HSR VZHRSBNZS-
ONBJDS, zmc knnjdc zs hs, zmc sgdm gtqqhdc nm, Zkhbd rszqsdc sn
gdq edds, enq hs ekzrgdc zbqnrr gdq lhmc sgzs rgd gzc mdudq
adenqd rddm z qzaahs vhsg dhsgdq z vzhrsbnzs-onbjds, nq z vzsbg sn
szjd nts ne hs, zmc atqmhmf vhsg btqhnrhsx, rgd qzm zbqnrr sgd
ehdkc zesdq hs, zmc enqstmzsdkx vzr itrs hm shld sn rdd hs ono
cnvm z kzqfd qzaahs-gnkd tmcdq sgd gdcfd.
```