COMP 141: Notes on Files

(1) Reading a predetermined number of items from a file (this code assumes the first line of the file is an integer that represents the number of following lines in the file):

```
my_file = open("filename.txt", "r")
how_many_items = int(my_file.readline())
for ctr in range(0, how_many_items, 1):
  line = some_file.readline()
  # process this line, or read in more lines if each item
  # in the file spans multiple lines
```

```
4
this is line one
this is line two
this is line three
this is line four
```

(2) Reading from a file that has a sentinel at the end (here, the word STOP is the sentinel):

```
my_file = open("filename.txt", "r")
while True:
  line = my_file.readline()
  if line == "STOP\n"   # end the loop when we see STOP
    break
  # process this line, or read in more lines if each item
  # in the file spans multiple lines
```

```
this is line one
this is line two
this is line three
this is line four
STOP
```

(3) Reading from a file that has no sentinel at the end (end the loop when the file ends):

```
this is line one
this is line two
this is line three
this is line four
```

```
my_file = open("filename.txt", "r")
while True:
  line = my_file.readline()
  if line == ""          # if we read the empty string ("") from the file, that
    break                # means we've reached the end, so stop the loop
  # process this line, or read in more lines if each item
  # in the file spans multiple lines
```

(4) Another way to read in each line and end the loop when the end of the file is reached:

```
my_file = open("filename.txt", "r")
for line in my_file:
  # At this point, use the variable "line" just
  # like you would in the code immediately above.
```

Reminders and Helpful Ideas

- When writing your program, print the lines from the file as you read them. This is helpful for debugging, and you can comment out the print statements after you're sure your code is working.
- The return value from `readline()` will be a string with a newline (`"\n"`) attached to the very end, unless you've reached the end of the file. This is why in method 2 we have to compare the line variable against `"STOP\n"` rather than just `"STOP"`. The only time `readline()` will return a string without a newline at the end is when the end of the file is reached (see method 3).
- Usually, the first step in processing each line of the file is to remove the newline from the string variable holding the contents of the line of the file you just read with `readline()`. You can do this by using the `rstrip()` function, which returns a copy of a string with any whitespace (newlines, spaces, tabs) removed from the right side of the string:
  ```
  # removes any spaces, tabs, or newlines from the end of the variable line
  line = line.rstrip()
  ```
- The conversion functions `int()` and `float()` will accept a string argument with a newline; that's why in method 1 we can use "`how_many_items = int(some_file.readline())`" directly without a call to rstrip().
- Method 2 only works when the sentinel value only occurs *once at the very end of the file*. If the sentinel value occurs multiple times in a file, use method 3 or 4 instead and test for the sentinel using an "if" statement inside the loop.