Introduction to Python
Day Three Exercises

Stephanie Spielman
Email: stephanie.spielman@gmail.com

1 For-loop and if/else

1. A professor has decided to curve grades in a very special way: grades above 95 are reduced by 10%, grades between 75-95 (inclusive) remain the same, and grades below 75 are raised by 10%. You have been tasked with crunching the numbers.

   (a) Create a list of new grades that reflects these rules from the following grade list:
   `grades = [45, 94, 25, 68, 88, 95, 72, 79, 91, 82, 53, 66, 58]`

   (b) The professor has changed his mind: he now wants to use a scaling factor of 0.078325 (instead of 0.1), because why not! Recompute the grades from part 1 using this new scaling.

   (c) The *nested* list below contains three sets of grades for silly professor's three classes:
   `all_grades = [[45, 94, 25, 68, 88, 95, 72, 79, 91, 82, 53, 66, 58], [23, 46, 17, 67, 55, 42, 31, 73], [91, 83, 79, 76, 82, 91, 95, 77, 82, 77]]`
   Create a new nested list with the curved grades for each of these groups.

   (d) Now, imagine that those three sets of grades correspond, in order, to the classes indicated in this list:
   `class_names = ["Psychology 101", "Sociology 101", "Political Science 101"]`
   Create a *dictionary* representing the *curved* grades for each of these classes. Your final dictionary should have the class name as keys, and each list of curved grades as values. **Do not recompute the curved grades** for this question. Instead, use only the nested list of curves grades and this class names list to create the final dictionary.

2. This dictionary provides the molecular weight for all amino acids:
   ```
   amino_weights = {"A":89.09, "R":174.20, "N":132.12, "D":133.10, "C":121.15,
   "Q":146.15, "E":147.13, "G":75.07, "H":155.16, "I":131.17, "L":131.17,
   "K":146.19, "M":149.21, "F":165.19, "P":115.13, "S":105.09, "T":119.12,
   "W":204.23, "Y":181.19, "V":117.15}
   ```
   Perform the following tasks with this dictionary (for ease, copy/paste it into a python script):

   - Determine the molecular weight for this protein sequence:
   `"GAHYADPLVKMPWRTHC"`.
   - This protein sequence, `"KLSJXXFOWXNNCPR"` contains some ambiguous amino acids, coded by "X" and "J". Calculate the molecular weight for this protein sequence. To compute a weight for an ambiguous amino acid "X" and "J", use the *average* amino acid weight.
   Hint: the `len()` and `sum()` functions and the `.values()` dictionary method will be useful! The `len()` and `sum()` functions can be used together to compute a mean value of a list. In addition, the python code `in`, which we have used for looping, can also be used to check if a certain key is in a dictionary. For example, `"X" in amino_weights` would return `False`.

2 File Input and Output

Be sure to always close the file once you are done with it! If you use `with` control flow, Python will close the file automatically. The following exercises make use of files distributed in today's course materials. Be sure that you are in the correct directory when interacting with the files!

1. Open the file "file1.txt" in read-mode, and print its contents to screen. For this, you should use the `.read()` method, which saves the contents of the file to a single string. Perform this task twice: once using `open` and `close`, and once using `with` control-flow.

2. Open the file "file1.txt" in read-mode, and save all lines in this file to a list using the `.readlines()` method. Write a new file called "upper_file1.txt" which contains the same contents of "file1.txt" but in upper-case. Try to do this task using a single for-loop, and don't forget that in order to write newlines (the "enter" key) to a file, you must include `"\n"` in the string you are writing to file!

3. Open the file "upper_file1.txt" in append-mode, and *append* the sentence: "I just created this upper-case file!" to the bottom of the file. Close the file and open it (separately) to examine the contents.

4. Again, open the file "upper_file1.txt", this time in read-mode. Loop over the file lines *without* using `.read()` or `.readlines()`. Print out lines as you loop.

5. Modify the previous for-loop to only print out lines in "upper_file1.txt" which contain at least (i.e. >=) 5 letter "E"s. This will require an `if` statement as well as the method `.count()`.

6. You should notice 20 files named file1.txt, file2.txt, ...file20.txt. Write a for-loop to open each of these files (Hint: use the `range()` function to loop over file names). For each file, write a for-loop over the file lines (this does not use `.read()` or `.readlines()`). Print each line that contains more than 25 characters (determine this with `len()`.)

7. Using the zoo-keeper dictionaries from the **Working with dictionaries** exercises (also provided below), create and write to a new file called "animal_food.txt" which contains the sentences you created earlier:
   The lion eats meat
   The gazelle eats grass
   The anteater eats termites
   The alligator eats visitors
   ... etc.

```
category = {"lion":  "carnivore", "gazelle":  "herbivore", "anteater":
"insectivore", "alligator":  "homovore", "hedgehog":  "insectivore", "cow":
"herbivore", "tiger":  "carnivore", "orangutan":  "frugivore"}

feed = {"carnivore":  "meat", "herbivore":  "grass", "frugivore":  "mangos",
"homovore":  "visitors", "insectivore":  "termites"}
```