Introduction to Python
Day Two Exercises

Stephanie Spielman
Email: stephanie.spielman@gmail.com

1 Working with dictionaries

1. Define this dictionary in Python: `molecules = {"DNA":"nucleotides", "protein":"amino acids", "hair":"keratin"}`, and perform the following tasks:

   (a) Create two lists called `molecules_keys` and `molecules_values`, comprised of the keys and values in `molecules`, respectively. Use dictionary methods for this task.

   (b) Add the key:value pair `"ribosomes":"RNA"` to the `molecules` dictionary. Print the dictionary to confirm.

   (c) Add yet another key:value pair, `"ribosomes":"rRNA"`, to the `molecules` dictionary, and print out the new dictionary. Understand why the result contains the key:value pairs shown.

2. Congratulations, you"ve been hired as a zoo-keeper! Now you have to feed the animals. You received these handy Python dictionaries which tells you (a) to which category each animal belongs, and (b) what to feed each category animal category:

```
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"}
```

   Copy and paste these dictionaries into a Python script. Use indexing to determine what you should feed the orangutan and print the result.

2 If/elif/else Exercises

1. In Texas, you can be a member of the elite "top 1%" if you make at least $423,000 per year. Alternatively, in Hawaii, you can be a member once you start making at least $279,000 per year! Finally, if you live in New York, you need to earn at least $506,000 a year to make the cut.

Andrew is CEO of Big Money Company, and he earns $425,000 per year, and Stacey is CEO of Gigantic Money Company with an annual salary of $700,000. Use if-statements to determine, and print, whether Andrew and Stacey would be considered top 1%-ers in Texas, Hawaii, and New York.

2. Copy and paste the list below into a Python script and perform the following tasks.
   `b = [19, 3, 2, 88, 82, 31, -9, 8, 33, -6, -111]`

   (a) Write an if/else statement to determine whether the length of list `b` is above or below 10. If this condition is true, create a new list called `newb` that contains the first 5 numbers in b. If this condition is false, create a new list called `newb` that contains the *last 5 numbers* from list `b`.

   (b) Determine the sum of the list `newb` (Hint: use the `sum()` function, which returns the sum of a list, e.g. `sum([1,2,3])` returns 6).

   (c) Use an if/else statement to determine if this sum is even or odd (Hint: use the modulus operator, `%`), and print a sentence summarizing the result.

3 For-loop Exercises

1. Write a for-loop to print the powers of 2, ranging from 2^0 to 2^{15}. (Note that in Python, the exponent symbol is **, as in 3**2 = 9).

2. Modify your code from (1) such that all each power-of-2 value is saved to a list called `powers2`. Print the resulting list.

3. This is the longest German word[1]: "rindfleischetikettierungsuberwachungsaufgabenubertragungsgesetz" (accents removed), meaning "widow of a Danube steamboat company captain". Save this word to a string variable in Python, and count the number of times each letter in the alphabet (a-z) appears in this word. For every letter a-z, print the number of occurrences, e.g. "a: 4". Hint: Use a *nested for-loop* structure. The outer loop should loop over the word, and the inner loop over the alphabet. Now, modify this code in several ways:

   (a) Modify your code such that letters with a count of 0 do not get printed. Do this task twice, using two different strategies:
      - For each letter, decide *after* counting its occurrences if it should be printed or not
      - Use the `in` operator before determining each letter's count, and then only count the letter if it has a non-zero count. The `in` operator returns True or False, as in: `"a" in "apple"` returns True, and `3 in [4,5,6,7,8]` returns False.

   (b) Modify your code to save the total letter count (a single counter for all letters!) as you loop. For this, you will need to define a variable to store the sum, and you can increment this variable with counts as you go. Once this is complete, use an if-else statement to check that the total sum is equal to length of the full word. Print a statement indicating whether your sum is correct or not.

[1]Sadly, this AP article reported in 2013 that this word has been stricken from the dictionary.

4. A silly 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. This time, *round* the grades to 2 decimal places using the function `round()`. This function takes two arguments: the value to round and the number of decimal places, e.g. `round(75.6789, 1)` returns a value of 75.7.

   (c) Finally, modify your code (if you need to) so that the scaling value (either 0.1 or 0.078325) used in your for-loop is a *pre-defined variable* to avoid hard-coding the scaling value.

   (d) (Note: this question should be skipped if you are pressed for time!) The *nested* list below contains three sets of grades for silly professor's three class sections:
   `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, using a scaling factor of 0.06.

5. Using the zoo-keeper dictionaries from today's second dictionary exercise, loop over the animals in `category` and, using the `feed` dictionary, determine and print what food each animal should eat, e.g. "The gazelle should eat grass."

6. 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". Calculate the molecular weight for this protein sequence. To compute a weight for an ambiguous amino acid "X", 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.

7. The strings methods `.startswith()` and `.endswith()` are useful for determining if a given string starts/ends with a particular substring, and they return True or False. For example, `"banana".startswith("ba")` returns True, and `"oranges".endswith("W")` returns False. Note that both methods are *case-sensitive*, meaning that upper- and lower-case matter!

   Using the method `.startswith()`, determine if the sentence "Dan's dog, dubbed Fluffy, dove deep in the dam and drank dirty water, but he didn't drown" is an alliteration for the letter "d". For the purposes of this example, let's assume that if at least 50% of the words start with the same letter, then it is alliteration. Otherwise, it is not.
   Hint: the methods `.split()` and `.lower()`, as well as the function `len()`, will be useful.