Introduction to Python
Day Two Exercises

Stephanie Spielman
Email: stephanie.spielman@gmail.com

1 Working with lists and strings

First, define the following variables:

- `numbers = [0, 1, 1, 2, 3, 5, 8, 13]`
- `mammal = "orangutan"`
- `bird = "sparrow"`

1. Use indexing to print out the *fourth* element of the list called "numbers". Now, use indexing to *redefine* the fourth element of the list "numbers" to be -10. Print the list to check.
2. Create a new variable called "original_length" which contains the length of the list "numbers" (use the `len()` function for this!). Now,...
   (a) Use the method `.append()` to add the new entry "21" to the end of the list "numbers".
   (b) Create another variable called "update_length" which contains the length of "numbers" *after* you have appended 21.
   (c) Print the variables "original_length" and "updated_length". Make an `if/else` statement to check if "updated_length" is one larger than "original_length". Print "append worked!" if this condition is met, and "append failed" if not. Keep trying until you get "append worked"!
3. Write an `if/else` statement to check if the sum of the list "numbers" is below 50 (Hint: use the `sum()` function, which adds up all items in a list). Print informative statements accordingly.
4. Use the code `.count()` to count how many "a"'s are in the variable "mammal" (e.g. `"hello".count("l")` will return 2). Once you have this working, write an `if/elif/else` statement to check if the variable "mammal" or "bird" has more a's. Print informative statements accordingly.
5. Use the `.upper()` to *redefine* the variable "bird" as all uppercase. Print the updated variable to confirm.
6. Create a new variable called "both_animals" which contains the contents "SPARROWorangutan". Make sure to do this entirely with variable names (not with the actual words themselves!!). Now, use `.count()` to count how many lower-case "o"'s are in this string. Did you expect this answer? Why or why not?
7. Using indexing, change the first entry of "numbers" to be the string contained in "both_animals" (again, use only variable names for this! not the words!). Print to double check.
8. Create a new list: `numbers2 = [-4, -8, -12, -16]`. Use negative indexing to change the final entry in "numbers" to be this "numbers2" list. This creates a *nested list*. Print the final length of the list "numbers". Did you expect this? Why or why not?
9. Finally, determine the length of the final entry in "numbers" (Hint: you will need to use indexing and the `len()` function).

2 For-loops

First, define the following variables:

- `numbers = [0, 1, 1, 2, 3, 5, 8, 13]`
- `animals = ["gorilla", "canary", "frog", "moth", "nematode"]`

1. Write a for-loop over the list "numbers". At each iteration, print the value in "numbers" plus 2. Your code should print out the following:
   2
   3
   3
   4
   5
   7
   10
   15

2. Write a for-loop over the list "animals". At each iteration, print out the value in "animals" followed by its length. Your code should print out the following:
   gorilla 7
   canary 6
   frog 4
   moth 4
   nematode 8

3. Modify the previous for-loop to print out the capitalized version of each animal (do not redefine anything, just print!). Your code should print out the following:
   GORILLA
   CANARY
   FROG
   MOTH
   NEMATODE

4. Write a new for-loop to create a new list called "cap_animals" which should contain the capitalized versions of all entries in "animals." For this task, you will need to define the list "cap_animals" before the for-loop, and use `.append()` to build up this list as you go. Print the final list after the for-loop.

5. Write a for-loop to create a new list called "negative_numbers" which should contain the negative values of the entries in "numbers", following a similar procedure to the previous task. Once this list is complete, write an `if/else` statement to check if the sum of "negative_numbers" equals -1 times the number of "numbers". Use the function `sum()` in the if statement. Print informative messages in the `if/else` construct.

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

7. Modify the previous for-loop to print out *every other* power of 2. Perform this task using two different approaches:

   - Modify the arguments given to `range()`
   - Write an if-statement within the for-loop to check if the power should be printed (Hint: check if the *iteration* count is even or odd).

3 Working with dictionaries

1. Define this dictionary: `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 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"}
```

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

   (b) Write a for-loop to loop over "feed" and print out what food each animal type eats. Your code should ultimately print the following (in any order!):
   The carnivore eats meat
   The herbivore eats grass
   The frugivore eats mangos
   ...etc
   Hint: You might find it helpful to first loop over the "feed" dictionary and simply print the loop variable. Extend the code from there to print the full sentence.

   (c) Write a for-loop to print out what each animal eats. Your code should ultimately print the following (in any order!):
   The lion eats meat
   The gazelle eats grass
   The anteater eats termites
   The alligator eats visitors
   ... etc
   For this task, you should loop over the dictionary "category" and use indexing to obtain the relevant information from the "food" dictionary to create your sentence.
   Finally, modify the previous for-loop so that it creates a new dictionary called "animals_eat" while looping over "category". This dictionary should contain the exact animal:food pairs, e.g. "lion": "meat" will be one key:value pair. Print out the resulting dictionary.

4 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) 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.

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.