Introduction to Python, 2018

Day 2 Exercises

Part I: For loops

First, define the following two 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. In other words, 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. Write a for-loop over the list `animals`. At each iteration, print out the uppercase 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 `upper_animals` which should contain the uppercase versions of all items in `animals`. Hint: For this task, you will need to define the list

`upper_animals` *before* entering the for-loop, and you will need to use the method `.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 items 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 (hint: use the function `sum()`) of `negative_numbers` equals -1 times the sum of `numbers`.
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 by modifying the arguments given to `range()`.

Part II: Dictionaries

1. Define this dictionary and perform the following tasks:

   ```
   molecules = {"DNA":"nucleotides", "protein":"amino-acids", "hair":"keratin"}
   ```

   - Add the key:value pair `"ribosomes":"RNA"` to the `molecules` and print to confirm.
   - Add another key:value pair, `"ribosomes":"rRNA"` to the `molecules` dictionary and print. Do you understand why the dictionary contains the key:value pairs shown?
   - Write a for-loop over the dictionary to print each key-value pair in sentence form, " is comprised of ." (i.e., "DNA is comprised of nucleotides").
   - Write a for-loop over the dictionary to make a list of keys which are more than 5 letters long. For this task, you should employ an `if` statement, the `len()` function, and the `.append()` method. Print the list once it is made.
   - Write a for-loop over the dictionary to make a list of *values* which contain the letter `a`. Perform this task in two ways:
     - Use the `.count()` method to count the number of `a`'s in each dictionary value, and use an `if` statement to determine if `a`'s were found
     - Use the `in` statement. This statement returns True or False, for example:

       ```
       x = "Stephanie"
       print("a" in x)
           True
       print("w" in x)
           False
       ```

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. Perform the following tasks with these dictionaries:

```
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"}
```

   - Determine what you should feed the orangutan and print the result.
   - Write a for-loop to loop over "feed" and print out what food each *-vore* eats. 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.Your code should ultimately print the following (in arbitrary order):

     ```
     The carnivore eats meat
     The herbivore eats grass
     The frugivore eats mangos
     ...etc \\
     ```

   - Write a for-loop to print out what each *animal* eats. 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. Your code should ultimately print the following (in arbitrary order):

     ```
      The lion eats meat
       The gazelle eats grass
       The anteater eats termites
     The alligator eats visitors
     ... etc
     ```

   - Modify the previous for-loop so that it creates a new dictionary called `animals_eat`. This dictionary should contain the exact animal:food pairs, e.g. `"lion": "meat"` will be one key:value pair. Print out the resulting dictionary.

Part III: Functions

1. Write a function to compute the GC content of a DNA sequence. The function should accept a single argument, the DNA sequence, and return the GC percentage. Test your function with the nucleotide sequence "AGCTATAGCATAGC".

2. Write a function that calculates the percentage of a given nucleotide from a DNA sequence. The function should accept two arguments: the nucleotide of interest and the DNA sequence. It should return the nucleotide percentage. Test your function with the nucleotide sequence "AGCTATAGCATAGC".

3. Write a function that calculates the percentage each nucleotide in a given DNA sequence. of a given nucleotide from a DNA sequence. The function should accept a single argument, the DNA sequence, and return *a dictionary* containing `key:value` pairs of `nucleotide:percentage`. You can assume that the provided sequence contains only A, C, G, T. Test your function with the nucleotide sequence "AGCTATAGCATAGC".

4. Now, things get tricky: modify the function you wrote for the previous question to instead account for *ambiguities* (for example, "N" nucleotide) in the sequence. You should return a dictionary of percentages for any character present in the provided DNA sequence. (Hint: use a for-loop to "build up" a dictionary of counts for those characters which appear in the sequence). Test your function with the nucleotide sequence "AGCTANTAGNNNNNNNNNCATAJGC".

5. Write a function to guess whether a provided sequence is DNA or protein. For this task, assume that any sequence comprised of $\geq 50\%$ A, C, G, T is a DNA sequence. Test your function with the following two sequences:

   - "AGCTATGCATACGNNNNAGCATAGC"
   - "AGIILLCTATWCAGCATADSARCVLMKGC"

6. Write a function to calculate the molecular weight of a provided protein sequence. The function should accept a single argument, the protein sequence, and return the molecular weight. Assume there are no ambigious amino acids. Use the following dictionary in your function:

```
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}
```

   Test your function with the following sequence:

   - "GAHYADPLVKMPWRTHC"

7. Write a function to calculate the molecular weight of a provided protein sequence. The function should accept a single argument, the protein sequence, and return the molecular weight. This time, assume that the molecular weight of an ambiguous amino acid is the average weight of all amino acids. Test your function with the following sequences:

- “GAHYADPLVKMPWRTHC”
- “KLSJXXFOWXNNCPRHGGYA”