Lab 6: Repetition: Loops, Lists, and Strings

1. Implement a function of a single variable *n* that creates a list of all numbers *i* between 1 and *n* (included) that satisfy the following conditions simultaneously:

    1. the square of i divided by 4 has remainder 1
    2. the cube of *i* divided by 5 has remainder 4
    3. the fourth power of *i* divided by 6 has remainder 3

2. Write a function of a single string that returns True or False according to whether the string has a double letter. (Hint: Since you need to look at the adjacent letter, you need to have a loop on an index into the word.)

3. Write functions that take as parameters two lists. The functions then return a list calculated from the two parameter lists. Function 1 will return all elements in the first list that are not in the second list. Function 2 will return all elements that are in both lists. Function 3 will return all elements that are in one of the lists, but not in both of them. (Don't use set functions and operations that we have not yet looked at. You can do this with a single loop for the first two functions and two loops for the last two.)

4. Write functions that takes a list as its single parameter.

    a. Return a list that returns a list of all elements of the parameter that only appear once in the list.

    b. Return a list that returns all elements of the parameter that appear at least twice.

    c. Return a list that returns all elements of the parameter that appear exactly twice.

5. Implement a function that translates a word into Tutnese[1]. You can assume that the word is in lower case. You can find the rules for Tutnese on the wikipedia page. The difficult part is the treatment of double letters in a word. Just like in the exercise 2, you need to use an index into the word to figure out whether the letter is repeated. Where there are alternatives given, just pick one. Then try out your code on all the examples in the wikipedia article.

[1] Described in Angelou, Maya, I know why the caged bird sings.