Laboratory 5: Strings and Lists

(1) Write a function of a string that returns the number of digits in the string. (Hint: You can use the string `'0123456789'` and ask whether a letter is in that string.)

(2) Write a function of a string that returns the number of consonants in the string. (Hint: you can use the string of consonants `"bBcCdDfFgGhHjJkKlLmMnNoOpPqQrRsStTvVwWxXzZ"`.)

(3) Write a function of *n* that returns the first *n* Fibonacci numbers. The Fibonacci numbers are defined as the sequence 0, 1, 1, 2, 3, 5, 8, 13, 21, … where an element is the sum of the preceding two elements. Do so in the following manner: Initialize a list to the first two Fibonacci numbers, i.e. [0,1]. Then do *n*-2 times (the Pythonesque way is to say `for _ in range(n-2):`) the following: append the sum of the last two elements in the list to the list. (Recall that the last element of list `fiblist` is `fiblist[-1]` and the pen-ultimate one is `fiblist[-2]`.)

(4) Write a function `fun(a_list, a_number)` with a list as a parameter and a number as another parameter that returns a list consisting of all numbers between 0 and `a_number` that are *not* in the list.

(5) Pig-Latin translator: Our goal is to write a function that transforms a string into its pig-latin equivalent. If a word starts with a vowel and ends in a consonant, then we just add "ay" to the word. Examples are:

omelet —> omeletay

egg —> eggay

If a word starts with a vowel and ends in a vowel, then we add "way" to the word. Examples are

are —> areway

I —> Iway

If a word starts with one or more consonants, "a consonant combination", then move the consonant combination to the end of the word and add "ay" to it. Examples are

pig —> igpay

smiles —> ilessmay

stupid —> upidstay

We solve this problem by generating several helper functions. The first helper function decides (returns a Boolean) whether a word starts and ends with a vowel. The second helper function decides whether a word starts with a vowel and ends with a consonant. The third helper function deals with the remaining case. It takes a word and returns a list with two words. The first element of the list is the word without the leading consonant combination and the second one is the rest of the word. For example

`helper3("frugal")` returns `["ugal", "fr"]`.

You can implement this function by going through the letters in the argument string, maintaining to results list, beginning and end. While you are reading consonants from the string, you add the to the beginning list. When you hit the first vowel, you add to the end list. To implement something like this, we need a state, a Boolean value. Here is a sample implementation:

```
def helper3(word):
    be = [ ]
    en = [ ]
```

```
    seeing_consonants = True
    for letter in word:
        if seeing_consonants:
            if letter not in "aeiou":  #a consonant
                be.append(letter)
            else:
                seeing_consonants = False
                en.append(letter)
        else:
            en.append(letter)
    return ["".join(en), "".join(be)]
```

While we are in the state seeing_consonants, we are looking at consonants and place them into the be(ginning)-list. The moment we see the first vowel, we leave the state (by setting seeing_consonants to False) and start appending to the en(d)-list.