Laboratory 3: Strings and Lists

1. Use the: `for x in lista` construction in the implemtation of a function that takes as argument a list and returns the product of all list elements. For example
   `product([2,3,5,7,11,13])`
   returns 30030.

2. Write a function fibonacci of a number *n* that starts out with a list [0,1]. It then appends *n-2* times the sum of the last two elements in the list. If the list is called lista, then the last two elements are lista[-1] and lista[-2]. The function returns the list. For instance, calling fibonacci(10) yields [0, 1, 1, 2, 3, 5, 8, 13, 21, 34].

3. 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. You can also use the isdigit function in python e.g. ask whether letter.isdigit() is True or False before you increment a counter.) Example: nnr_digits("3 wise monkeys and 4 dumb buffalo:") returns 2.
4. Write a function that takes two lists as input and returns a list with elements that are in both of the lists. For example, `intersection([1,3,"hello", 3.45], [3.45, "hello"])` returns `['hello', 3.45]`.

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

6. Write a function that replaces every vowel in a string with a digit, starting with 0. Example: `change_str("Thomas Schwarz")` returns `'Th0m1s Schw2rz'`

7. 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.