Exercises01

February 21, 2015

```
In [2]: from IPython.display import Image
```

0.1 Computer lab

These exercises are designed to give you some practice coding in Python and proivde familiarity with the language syntax.

Exercise 1 [10 pts]. Write code to solve the Project Euler puzzle 1

```
In [3]: # Your code here
```

Exericse 2 [10 pts]. Wikipedia gives the sample correlation coefficient formula as shown below. Write functions to calculate the sample mean, the sample standard deviation and the sample correlation coefficient. Calculate the sample correlation coefficent for the following lists of numbers:

```
x = [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0]
y = [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68]
```

Do NOT use `numpy` functions - the idea is to code the functions yourself! The square root of x can be calculated as `x**0.5`.

```
In [4]: Image(url="http://upload.wikimedia.org/math/0/4/e/04e3ee493ddb1f01e03d8bf024fbd0a5.png")
```

```
Out[4]: <IPython.core.display.Image at 0x7f9126a78550>
```

```
In [5]: Image(url="http://upload.wikimedia.org/math/7/0/d/70df5220933ae8298cd5ef1c719360bf.png")
```

```
Out[5]: <IPython.core.display.Image at 0x7f9126a78590>
```

```
In [6]: # Your code here
```

Exercise 3 [10 pts]. Write a function to calculate hailstone numbers.

Start with any positive integer (an initial seed) and obtain a sequence of numbers by following these rules.

1. Base case: If the number is 1, stop.
2. Recursive case: If the current number is even, divide it by two; else if it is odd, multiply it by three and add one.

The seqeunce of numbers obtained are known as *hailstone numbers* - although it is not proven that all sequences will eventually terminate with 1 (Collatz conjecture), no counter-example has yet been found.

For example, starting with 7, we get the sequence 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

A recursive function to find the sequence of hailstone numbers is given below. Write a *non-recursive* version of the function that gives the same result. Using your function, what is the length of the sequeence obtained when we start with 23?

```
def hailstone(n, acc=None):
    """Given a positive integer n, return the series of hailstone numbers."""
    if acc is None:
        acc = []
    acc.append(n)
    if n == 1:
        return acc
    else:
        if n%2 == 0:
            return hailstone(n/2, acc)
        else:
            return hailstone(n*3 + 1, acc)
```

```
In [7]: # Your code here
```

Exercise 4 [20 pts]. A Caesar cipher replaces each in a string with the letter k modulo 26 positions down. For example, with $k = 3$, 'a' would become 'd', 'b' would become 'e' and so on. The Caesar cipher is very easy to crack, since the relative letter frequencies are preserved - that is, 'e' is the most commonly used English letter, so the most frequent letter in the cipher is likely to be the code for 'e'. We can download a large corpus of English text (e.g. a few books from Project Gutenberg), and estimate the individual letter frequencies. Using for example, a multinomial model where each letter is one of 26 possibiilites with frequencies given by the the previous estimates, and assuming that each letter is independent, we can simply go through all 26 possible values of k and choose the model with the highest likelihood. Or just use χ^2 statistic to compare observed and expected frequencies.

Here is an extended exercise to write a Caesar encoder and code to crack the cipher.

Writing the encoder/decoder: - Convert a sentence to lowercase - Write a function let2int to convert a lowercase letter to a number from 0 to 25 - Hint: look up ord() function - Write the reverse function int2let to convert a number to a lowercase letter - Hint: look up chr() function - Write a functiton to encode a sentence with shift k - Hint: Use list comprehension with ternary operator to check for a lowercase letter - Write a function to decode a sentence with shift k - Hint: Use the encode function - Confirm that sentence = decode(encode(sentence))

More hints for encoding/decoding:

There are two slightly tricky aspects to encoding/decoding. First is that we want to convert any character into an integer in $[0, 26)$. Because letters are ordered ('b' comes after 'a'), this can be done by subtracting `ord('a')` for the lowercase letters. After adding k, we need to take the number modulo 26 since the Caesar cipher "wraps around". Second is that this will only work for lowercase letters - so we need to pass along any characters that are not in 'a-z' unchanged.

Writing the cracker: - Write a function to download a book from Project Gutenberg - Hint: use `python text = urllib2.urlopen('http://www.gutenberg.org/ebooks/1342.txt.utf-8').read()` - The book is now in the Github repository as book.txt. Instead of using the above code, just use `python text = open('book.txt').read()` - Write a fucntion to estimate letter frequencies in text - Hint: recall various ways to count letters in a string from Lecture 1 - Write a function to calculate the χ^2 statistic given observed and expected frequencies - Hint: $\chi^2 = \sum_{i=1} k \frac{(\text{Observed} - \text{Expected})^2}{\text{Expected}}$ - Write a cracker to decode encrypted text by choosing the shift k with the smallest χ^2 score - Hint: xs.index(min(xs)) gives location of minimum value in the list xs. - Any reasoanable Englsh sentence should work - just maek one up or copy and paste from somewhere.

```
In [8]: # Your code here
```