Exercises01Solutions

February 21, 2015

In [1]:
```
import string
import numpy as np
from collections import Counter
import urllib2
```

0.0.1 Exercise 1

In [2]:
```
# Possible solution

x3 = [3*i for i in range(1, (1000+2)/3)]
x5 = [5*i for i in range(1, (1000+4)/5)]
x35 = [15*i for i in range(1, (1000+14)/15)]
print sum(x3) + sum(x5) - sum(x35)
```

```
233168
```

In [3]:
```
# Alternative solution

s  = 0
for i in range(1,1000):
    if i % 3 == 0 or i % 5 == 0:
        s += i
print s
```

```
233168
```

0.0.2 Exercise 2

In [4]:
```
def sample_mean(xs):
    """Samplele mean."""
    return sum(xs)/float(len(xs))

def sample_std(xs):
    """Sample standard deviaiton."""
    n = len(xs)
    xbar = sample_mean(xs)
    s = 0.0
    for x in xs:
        s += (x - xbar)**2
    return (s/(n-1))**0.5

def sample_correlation(xs, ys):
    """Sample correlation coefficient."""
    n = len(xs)
```

```
        xbar = sample_mean(xs)
        ybar = sample_mean(ys)
        sx = sample_std(xs)
        sy = sample_std(ys)
        r = 0.0
        for x, y in zip(xs, ys):
            r += ((x - xbar)/sx) * ((y - ybar)/sy)
        return r/(n-1)

    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]
    print sample_correlation(x, y)
```

```
0.816420516345
```

0.0.3 Exercise 3

```
In [5]: def hailstone(n):
            """Given a positive integer n, return the series of hailstone numbers."""
            acc = []
            while n != 1:
                acc.append(n)
                if n%2 == 0:
                    n /= 2
                else:
                    n = n*3 + 1
            acc.append(1)
            return acc

        seq = hailstone(23)
        print seq
        print len(seq)
```

```
[23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1]
16
```

0.0.4 Exercise 4

```
In [6]: def let2num(c):
            """Convert lowercase character to number."""
            return ord(c) - ord('a')

        def num2let(n):
            """Convert number to lowercase character."""
            return chr(ord('a') + n)

        def encode(cs, n):
            """Caesar cipher with offset n."""
            return ''.join([num2let((let2num(c) + n) % 26) if c.islower() else c for c in cs])
```

```
In [7]: def chisq(os, es):
            """Retruns chi-square score given observed os and expected es frequencies."""
            os = np.array(os)
            es = np.array(es)
            return np.sum((os - es)**2.0/es)
```

```
In [8]: def freqs(text):
            """Returns relative frequenies of lowercase letter in text."""
            ctr = Counter(text)
            counts = np.array([ctr[c] for c in string.lowercase], dtype='float')
            return counts/counts.sum()

In [9]: # Use Pride and Prejudice to build up base frequencies
        text = urllib2.urlopen('http://www.gutenberg.org/ebooks/1342.txt.utf-8').read()
        ref_freqs = freqs(text)

In [10]: def crack(code):
             """Find the original text by identifying the most likely encoding offset."""
             shift = np.argmin([chisq(freqs(encode(code, n)), ref_freqs) for n in range(26)])
             return encode(code, shift)

In [11]: cs = 'Statistical computing and computation is fun!'
         code = encode(cs, np.random.randint(1, 26))
         print code
         crack(code)

Szgzoyzoigr iusvazotm gtj iusvazgzout oy lat!

Out[11]: 'Statistical computing and computation is fun!'

In [11]:
```