# Finite_Sums_Answers_by_Coding

January 10, 2018

```
In [1]: from datascience import *
        import numpy as np

        %matplotlib inline
        import matplotlib.pyplot as plots
        plots.style.use('fivethirtyeight')
```

## 0.1 Checking Math Answers by Coding

That's a skill we're trying to teach in 140. If you're not sure whether you've solved a math problem correctly, you can check your answer by coding the problem or at least a special case of it. Use numerical calculations, plots, and so on as **sanity checks** to help you confirm or correct your initial attempts.

Course staff will expect that you'll have done that or tried out special cases algebraically before you bring your answer to them and ask, "Is this right?"

This notebook has some examples of this way of checking answers to exercises in the Math Prereqs sheet. You should answer first by using math, not code. For example, here's what you'd do in 3a:

$$\sum_{i=1}^{100}(4c_i + 5) ~=~ 4\sum_{i=1}^{100}c_i + \sum_{i=1}^{100}5 ~=~ 540$$

If you're not sure about the 540, you can scroll down to see the relation between the math and the code, and then run the cell for the answer.

### 0.1.1 Note on the Choice of Coding Methods Below

The code below is typically neither the most efficient nor the most elegant way to do the computation. Instead, the methods have been chosen to demonstrate direct connections with the math notation.

For example, the notation $\sum_{i=1}^{5} a_i$ is equivalent to: - Start with a sum of 0 - One by one, for each index $i$ in {1, 2, 3, 4, 5}, add $a_i$ to the sum

### 0.1.2 Preliminary on Indexing

A professor of computer science once told me, "1 is such a silly place to start." I get that, and Python uses 0-origin indexing. But ranges such as "from 1 to $n$" are ubiquitous in math, and for what we're doing here it's nice to have math ranges available in Python.

Hence the following function definition:

```
In [2]: def math_range(m, n):
            """For integers m < n, return sequence m, m+1, m+2, ... , n"""
            return np.arange(m, n+1)

In [3]: math_range(1, 10)

Out[3]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

### 0.1.3 Checking Answers to Prereq Sheet

Look at the math notation, then look at the code. Go back and forth between the two a couple of times to confirm that you see the equivalences.

```
In [4]: # 1a

        c = np.arange(11)
        # Confirm that the sequence c has the right values
        # over the indices 1 through 10.
        # Its values outside that range (e.g. at index 0),
        # are irrelevant. You can define them to be any numbers.

        s = 0
        for k in math_range(1, 10):   # Try replacing k with any
            s = s + c.item(k)         # other variable such as i
                                      # or Fred throughout.
        s

Out[4]: 55
```

**1b**

Code can find the sum for each numerical value $n$ (until the system runs out of space) but it doesn't give you a general formula. Math does.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

**Proof.** There are many, but here's one by induction on $n$. You're going to need the method of induction in several places in the course.

The result is clearly true for $n = 1$. Now assume the *induction hypothesis* that the result is true for some value of $n$.

Then for $n+1$ the result is

$$\begin{aligned}
\sum_{i=1}^{n+1} i &= \sum_{i=1}^{n} i \; + \; (n+1) \\
&= \frac{n(n+1)}{2} + (n+1) \quad \text{by the induction hypothesis} \\
&= (n+1)(\frac{n}{2}+1) \\
&= \frac{(n+1)(n+2)}{2}
\end{aligned}$$

as claimed.

```
In [5]: # 2

        s = 0
        for i in math_range(1, 10):
            s = s + 2
        s

Out[5]: 20
```

```
In [6]: # 3
        # Here are two very simple sequences that have the given sums.
        # Python can work with these
        # or with any other examples you come up with.
        # Math says that the choices of sequences don't matter
        # as long as they sum to 10 and 20.

        c = np.append(0, np.ones(100)/10)
        print(sum(c))

        d = np.append(0, np.ones(100)/5)
        print(sum(d))

10.0
20.0
```

```
In [7]: # 3a

        s = 0
        for i in math_range(1, 100):
            s = s + (4*c.item(i) + 5)

        s

Out[7]: 539.999999999999
```

```
In [8]: # 3b

        s = 0
        for i in math_range(1, 100):
            s = s + 4*c.item(i)

        s + 5

Out[8]: 44.99999999999992

In [9]: # 3c

        s = 0
        for i in math_range(1, 100):
            s = s + (4*c.item(i) - d.item(i) + 5)

        s

Out[9]: 519.9999999999993

In [10]: # 3d left hand side

         s = 0
         for i in math_range(1, 100):
             for j in math_range(1, 100):
                 s = s + c.item(i) + d.item(j)

         s

Out[10]: 2999.999999999098

In [11]: # 3d right hand side

         s = 0
         for i in math_range(1, 100):
             s = s + c.item(i) + d.item(i)

         s

Out[11]: 30.000000000000004
```