Python 1 Homework Assignment

Due Monday, November 4 before 2pm

Read chapters 7 and 8 of Haddock and Dunn. Chapter 7 provides an overview of Python and the fundamental elements of programming languages in general. It goes into more detail than I will have time to cover in class. Chapter 8 will lead you through the construction of the script on page 139.

In order to accommodate the variety of operating systems being used by students in this class, we will be using Python in a slightly different way than is shown in the book. In class, I showed you all how to use the Spyder integrated development environment (IDE) to write and execute Python scripts. You will now use Spyder to follow along with the examples in Chapter 8. Once you've finished building the script on page 139, you will extend it in the following ways:

1. **10% (all or nothing)**
   In addition to removing spaces, your script will remove gap "-" characters.

2. **80% (5% per base x 16 bases)**
   In addition to counting the four canonical DNA bases, your script will count occurrences of all characters in the IUPAC nucleic acid alphabet. The Wikipedia entry for these is pretty good: http://en.wikipedia.org/wiki/Nucleic_acid_notation

   Specifically, the characters your script will count are:
   - A - Adenine
   - C - Cytosine
   - G - Guanine
   - T - Thymine
   - U - Uracil
   - W - Weak
   - S - Strong
   - M - aMino
   - K - Keto
   - R - puRine
   - Y - pYrimidine
   - B - not A
   - D - not C
   - H - not G
   - V - not T or U
   - N - aNy base (not a gap)

Most of these will require your script to count and combine multiple characters. For example, the number of puRines in the sequence would be the number of “R” characters, plus the number of “A” characters, plus the number of “G” characters:

puRines = “R”s + “A”s + “G”s

There are several ways to accomplish this, some requiring more lines of code than others. I would encourage you all to aim for roughly two lines of code per character (one line for counting, and one for printing). We will explore alternatives using “for” loops and “if” statements on Monday.

3. **10% (half credit for partial compliance, no credit for noncompliance)**
   Your script will print each of these percentages to three decimal places, instead of just one.

What to submit

Your source code “.py” file, and the output of your program using the sequence “ATGAAC” from the example in chapter 8 of Haddock and Dunn.

How I will grade it

No credit will be given if I do not receive both files by the start of class.

No credit will be given if the program does not run. I will not debug your code. The script from page 139 would only receive 20%, but that is better than the zero that would be received for a script that is one typo away from full credit. **Make small changes and regularly check that your code is still running.**

I will run your script once, as it. I will then modify the sequence variable and run it again. If a feature fails in either run, the script will receive the lower of the two scores for that feature.