**Specific comments for this assignment:**

Students must have previous exposure to regular expressions (see my related homework that provides practice with learning regular expression syntax by solving English word puzzles). In particular, I strongly recommend the use of a linguistic metaphor, that is, DNA is a language (cf. LeBlanc and Dyer, 2007 for ideas).

Allow time for students to work in pairs at the board to apply regular expressions to sequences of DNA. At first, you may want to apply the regex to English sentences, for example: given this sentence (or DNA sequence) write the regex to find all five nucleotide palindromes (mirror repeats in DNA). For some examples, reverse the process by giving students a regex and ask them to produce a sentence (or sequence) where the regex will match a certain number of times. Students love to produce problems for their peers to solve as well.

The reader will notice that I have provided the Python syntax for compiling a regex and then iteratively applying it in the specification. From experience, I find students spend more time thinking about the larger solution rather than being stumped by the syntax of finditer() and groups.

Unlike in the previous two assignments, I do not provide a starter kit for this assignment. Thus students are required to use previous functions (e.g., getDNA() ) and put together an entire solution on their own.

**General comments for the entire set of DNA-focused programming assignments:**

(0) Regularly allocate time in your class sessions to bring in colleagues, in particular, a biologist who can talk (briefly) of the beauty of DNA. In this case, regulatory regions (biological "switches") are a hot topic in systems biology.

(1) Give students a sense of the real demand for future scientists who can work in multidisciplinary groups on computationally-intensive problems. Don't be shy to share that the skills to be learned in this course can "get students a job" at the start of an exciting career. Many students in the life sciences have general plans to "be a doctor"; these assignments may be an undergraduate's first exposure to the excitement in and demand for computational scientists in regards to research and medicine.

(2) Use a "flipped classroom" where students watch lectures before/after class in order to maximize the amount of hands-on Python play. For example, (a) require students to watch lectures on biological topics (cf. Udacity's 'Tales of the Genome' MOOC) outside of class; and (b) leverage programming practice sites for student practice outside of class (cf. codeAcademy's Python course).