HOMEWORK FOR LECTURE 10/5/2018
DUE 10/12/2018 [We will only grade Q#3]

PRELIMINARIES:

- Use <*gp9_exons_simple.txt*> for #1-4
  Each line is a different exon; exons are in order from 1st to last; 5 lines total. In principle your script should work for any file with arbitrary (and reasonable) exon numbers. If you are ambitious, you can make another sequence file to test
- Output format (unless noted otherwise) is flexible, but should be easily understood.
- Upload your 5 scripts as ONE (1) zipped file.

1. Count the number of A's, C's, G's, T's for each exon from a file
   - OUTPUT: Print output to screen

2. Join exons into cDNA and convert cDNA into mRNA; want all UPPER case
   - Remember in mRNA, U replaces T
     - Use `replace` method, e.g., `exon.replace('t', 'u')`
     - Try out `replace` in Jupyter to see how it works
     - Google `replace` if necessary
     - Note, this is case sensitive ('T' is different from 't')
   - OUTPUT: Write answer to a text file in fasta format (google 'fasta' if necessary):
     >gp9_cDNA <this line is the name of the sequence on the next line>
     ATG… <sequence, i.e., your computed cDNA >
     >gp9_mRNA
     AUG…

3. Reverse complement each exon as well as the entire sequence
   - The reverse complement of AAAGGCT is AGCCTTT (google if necessary)
     - OUTPUT: Write answer to a text file in fasta format
     - Result should be 12 lines (i.e., 6 pairs of lines)
       - >exon1
       - *the reverse complemented sequence*
       - …
       - >whole gp9
       - *rev comp of entire seq*
   - This is actually a little tricky with the `replace` method
     - You cannot just replace A with T in one line of the script and, then T with A in the next. You will have no T's…
       - Hint… sorry, try first, then if you really must, ask me or a TA
     - There is a better way using regular expressions (week 12)

4. Calculate %AT for each exon as well as the entire sequence
   - If Python 2 (… although you should be using Python 3, in which case you can ignore this.)
     - Put this line at the top of your script: `from __future__ import division`
       - (it is 2 underscores (i.e., 2x '_') on both sides of `future`)
       - This is useful for easy division of integers
       - This is an alternative to doing `float(3)/4` or `3.0/4`
   - OUTPUT: Print output to screen

5. Text munging: Count the number of "the" and "and" words in <*watson_crick_paper.txt*>.
   - Note, want words. E.g., "The" and "the" but not "there".