For the work below, you should type your program in Sublime, and execute that program at the terminal (i.e., don't use the Python interpreter to enter code directly). All of your work below can go in the same source code file. Remember that function definitions should go at the top of your program, and calls to your functions should all go at the bottom in your “main scope”.

1. Write a fruitful function named `getRandomNucleotide` that has no parameters and that returns exactly one randomly-selected nucleotide from among 'A', 'C', 'G', or 'T'.
   
   • You may want to use the Python `random.randint` function to randomly select a character's index within a string, as shown in class. To do so, at the very top of your program include the following line:

   ```python
   import random
   ```
   
   Make sure to call your function in the “main scope”, and test your program for correctness before moving on. It would be good to use a for loop in your testing so that you can easily call your function multiple times. Remember that you must print the result of your function call since it is a fruitful (non-printing) function.

2. Write a fruitful function named `getRandomDNASequence` that has one parameter called `length`. Your function should return a string of that given length corresponding to a randomly-generated DNA sequence.
   
   • Recall that you can start with an empty string and build your DNA sequence. Refer back to last Friday's quiz and to your #7 from the previous in-class work.
   
   • Use your `getRandomNucleotide()` function from above to do the work of generating a random nucleotide to append to your DNA sequence.
   
   Make sure to call your function appropriately in the “main scope”, and test your program for correctness before moving on. Make sure to test with different sequence lengths.

3. Write a fruitful function named `getDNAComplement` that has one parameter called `dnaSeq`. Your function should return a string corresponding to the complement of the DNA sequence given as the argument to your function.

   Make sure to call your function appropriately in the “main scope”, and test your program for correctness before moving on. Again, it would be good to use a for loop in your testing so that you can run several tests easily — in each iteration, generate and print a randomly-generated DNA sequence and then print its computed complement.

4. Write a fruitful function named `findStartCodon` that has one parameter called `dnaSeq`. Your function should look for the start codon "ATG" within the DNA sequence given as the argument to your function, and return the integer index where the first such start codon begins. If the start codon is not found anywhere in the sequence, your function should return -1.

   • If the sequence is "AATGCC", your function should return 1, the character index where the start codon begins. If the sequence is "AATCCC", your function should return -1.
   
   • Remember that you can use string “slicing”, as described here: https://bit.ly/2yLVGgII. An example might be `dnaSeq[7:10]`, which, for a sequence of at least 10 in length, would return the string made up of characters at indices 7, 8, and 9 (10 is excluded).
   
   • Pay attention to Note #4 at the link above, which states the following about string slicing:

     "If i or j is greater than len(s), use len(s)."

     In other words, if the sequence is "AATG" and you were to slice using `dnaSeq[2:5]`, slicing will return "TG", even though there is no character at index 4. In other words, slicing does not inappropriately walk off the end of the string.

   Make sure to call your function appropriately in the “main scope”, and test your program for correctness. Again, it would be good to use a for loop in your testing so that you can run several tests easily — in each iteration, generate and print a randomly-generated DNA sequence and then find and print the corresponding index where the start codon begins.
Following is an example output from the testing of my own program:

```
% python3 dna.py

Testing 10 calls to getRandomNucleotide:
G G C G T T A G T G
-------------------------------
Testing getRandomDNASequence(5):
CGGCT
Testing getRandomDNASequence(10):
TAAGGATTAC
Testing getRandomDNASequence(15):
CACCTATTCCGGCT

Testing 5 calls to getDNAComplement on random sequences of length 20:
Random Sequence: TTGCGTGACTGTTCTTTAA
Complement : AACGCACCTGACCAGAAATT
Random Sequence: CGGCT
Complement : GATATCTGTTTCAATTCAAT
Random Sequence: TAGTTTGTCGTGCCTCGCTT
Complement : ATCAAACAGCAGCAGCGAA
Random Sequence: CGAAACAATTAGATATCATG
Complement : GCTTTGTTAATCTATAGTAC
Random Sequence: CCTGGACATATACGGTCAG
Complement : GGACCTGTATATAGGACAGTC

Testing 5 calls to findStartCodon on random sequences of length 40:
Random Sequence: AACGGCAATGTATGACACTTGAATGGGCCAAACAGAAGTTC
Start Codon @ : 6
Random Sequence: CTGGAATGCCCAGCTAGGTGAGGCAGACCCACGTCACGTG
Start Codon @ : 5
Random Sequence: CCCAAGTGCCCTCGCATTTCCCACAGCCAACTCTATTAAG
Start Codon @ : -1
Random Sequence: ATGGTTTTAGGACAGCAGCAACCAATACGCTCATATACCCGA
Start Codon @ : 0
Random Sequence: AACGGGTTGCGACTGACACACAGCAGCACTGACCTCACCTAAC
Start Codon @ : -1
```