Introduction to Bioinformatics – AS 250.265
Assignment 2

As we progress through the course, we will be developing a balance between our understanding of how to use bioinformatics tools on the Internet and how those tools actually work. This assignment tests both of those skill sets directly. In class today, you were able to practice using the NCBI and BioWorkbench websites to perform simple searches of biological databases. The answers to your laboratory questions will make up the first part of this assignment. The second part of the assignment uses the Python programming language to study functions and libraries, and how those programming concepts can be applied to make convenient toolkits such as those found on the web.

While you will be asked to write programs for this assignment, the most important part of the programming aspect is understanding how Python interprets already written code. To that end, we will use the STATTools Python module to investigate how functions work in Python. This module is a collection of four statistical functions: mean, median, mode, and standard deviation. As you learn how these functions work, you will be developing intuition for how Python is processing the programs you have written and will write.

In the final part of the assignment, you will be asked to apply what you have learned from the statistics functions to another library of functions, DNATools. You will implement two functions, one that calculates the reverse-complement of a DNA sequence, and one that translates an RNA sequence to a protein sequence.

As before, your programming submissions should be named after the problem (e.g. q02c.py) and uploaded to the course web server. Written solutions may be turned in on paper or digitally. This assignment is due on February 17 and is worth 100 points.

Question 1: Laboratory Questions (40 points)

Submit your answers to the questions you answered for lab 1. These answers may be submitted through the web site as a Word document or text file, or they may be handed in on paper at the start of class.

Question 2: The STATTools Module (40 points)

Read the handout for assignment two on functions and modules. When you have finished, answer the questions below:

a. [written] Below is a variable assignment statement in Python. Show, *making one simplification at a time*, how Python evaluates the following statement (for an example, see the bottom of page two in the handout). You should evaluate the most deeply nested parenthesis first, then the second, etc. Python will also evaluate multiplication and division before addition or subtraction. Where there is true uncertainty in the order of operations, Python evaluates from left to right. Don't worry about more than three significant figures. (2 points)

```
val = ( (4. + 8.) / ( ( 6. * 12. – 4. ) * ( (6. + 3.) – 8. * 6.) ) ) + 8.
```

b. [written] Perform the same analysis from part (a) on the expression below. You may assume that sin and cos are the standard trigonometric functions, and their arguments are in radians. (2 points)

```
val = sin(3.14 / 2.) / (sin(0.3)*cos(0.3) + sin(0.3)*sin(0.3))
```

c. [written] Consider the factorial function defined in the handout. Of the standard Python types (number, list, string, dictionary), what type should the argument (n) be? What type of object (again, number, list, string, or dictionary) is returned? (2 points)

Download and examine the STATTools module and test program from the course website. Run the test program and make sure it works. Then, open up both programs and examine the code.

d. [written] For each of the functions mean, mode, median, and standard deviation, what type of argument do they expect? What type of value does each return? (2 points)

e. [written] Read over the code carefully. We learned modules (such as math) provide additional functions that can be used in the code we write. What function is the math module supplying to `stattools.py`? (2 points)

f. [written] Why are we able to call functions in stattools.py from stattest.py? What special syntax must we use to access the functions in stattools.py? Try to keep your answer to three sentences or less (5 points).

g. [programming] Examine the code for the mean function. In class we learned that there are multiple ways to represent loops in Python (see the supplemental handout from last week for more information). Rewrite the for loop in this function using a while loop. Submit only your modified stattools.py program (as q-02g.py or similar) and not the stattest.py program. (5 points)

h. [written] We are going to experiment with overwriting function names. At the end of the stattools.py file, add the following line:

```
mean = ‘This is a test message’
```

Save your program, and try running the test program. What happens? What does this tell you about function names? Again, try to keep your answer to three sentences or less. When you are finished with this question, remove the line you appended. (5 points)

The Python documentation page on dictionaries can be found at the following website:

http://docs.python.org/lib/typesmapping.html

Read over this page. It contains all of the relevant functions that may be called on a dictionary object *a*. The function mode initializes a dictionary and uses two of these operations, has_key and keys.

i. [written] Explain what both of these functions do. (2 points)

j. [written] In the mode function, what is a general description of the information stored in the keys of the dictionary? What is the general description of the values? (5 points)

k. [written] At the end of the mode function, there is no else statement (compare to the median function, which has an else). Why does the function still work without the else statement? (3 points)

l. [programming] Copy the factorial function from the handout to a new, blank program. Write a program that uses STATTools to calculate the mean, standard deviation, and median of the set of the first 10 factorial numbers (1-10, inclusive). Your program should print this information to the screen. (5 points)

Question 3: The DNATools Module (20 points)

In this question, you will implement two functions for a library of DNA utilities. The first function calculates the reverse complement of a given DNA sequence. If you are given a DNA sequence, the reverse complement of that sequence is the complement of the original sequence read back-to-front. For example, if the given sequence is:

```
5′ - ACGGTACA - 3′
```

Then the complement of that sequence is:

```
3′ - TGCCATGT - 5′
```

And the reverse complement is simply:

```
5′ - TGTACCGT - 3′
```

The second question will require you to translate an RNA sequence into a one-letter protein string using the standard genetic code. This is very similar to the assignment you had last week where you converted the three-letter amino acid codes into their corresponding one-letter codes. Because creating dictionaries can be tedious, the genetic code has been provided for you.

To begin this question, download the DNATools programs to your computer. When you open the module (dnatools.py), you will notice that skeletal functions are already written for you. It is your responsibility to write your solutions so they operate correctly within the framework of the outline of what is already there. *Do not modify code above or below the comments that delineate where you solution should go.* You should, of course, fill in your name and information in the comments at the top of the code.

Although this question contains two parts, please only submit one final file for your solution, dnatools.py, which contains your code for both parts (a) and (b) (and optionally the bonus question).

a. [programming] Implement the reverse complement function by filling in the code between the comments. Below is an algorithm that will do this: your object is to convert this description into Python code. Note that if `l` is a list, then `l.reverse()` will reverse the list. (10 points)

Algorithm for Reverse Complement

1. Convert the original string to a list
2. Reverse the list
3. Initialize a new empty list that will hold our result
3. Scan through each element in the original (reverse) list. If the element is A, append T to our new list, if the element is a T, append A to our new list, etc.
4. Convert our new list into a string (recall last week's assignment)
5. Return the string as the value of the function

b. [programming] Implement the translate function. You are on your own for this function, but remember that last week you implemented a very similar function for translating three letter codes of proteins. The genetic code has been provided for you.. You may assume that the sequence passed to this function will always be of an appropriate length (a multiple of three). In your implementation, use a star (*) for any of the three stop codons. (10 points)

   Understanding how to "slice" strings in Python may be helpful for your solution, although it is not required. Slicing is a technique that can extract parts of a string, like codons, while leaving the original string in tact. Information on slicing can be found in the Python tutorial. If you choose not to use string slicing, consider using two loops: one to iterate through codons (it would skip three bases at a time), and an inner loop to build the bases up into a single codon string.

c. [programming] **Bonus Question:** In the DNATools library you will find a third function, sixframe, that is supposed to translate all six frames of the genetic code given an mRNA sequence. Using your answer from part (b), finish this function. (Hint: it may be convenient to write a helper function that computes the reverse complement of an mRNA sequence. Additionally, you may need to change your answer to part (b) to be more robust about accepting strings of inappropriate lengths) (10 points)