In this lab, you will write functions to compute the score of a list of binding sites. Submit `Lab6.py` for participation credit at the end of lab.

Warm Up

`Lab6.py` contains a function `warmup()` that contains a variable called `strMat`. This is a list of strings. **Use FOR loops** to solve the following problems:

1. Print each row of `strMat`. It should look like

   ```
   abc
   def
   ghi
   jkl
   ```

2. Print each character of `strMat`, one for each line. It should look like

   ```
   a
   b
   c
   d
   e
   ...
   ```

3. Print each *column* of `strMat`. It should look like

   ```
   adgj
   behk
   cfil
   ```

Copy a Function to Read Text Files

```
## Given a file name, reads the list of binding sites and
## returns a list of strings.moif
def getBindingSites(fileName):
    myFile = open(fileName,'r') ## open the file
    lineString = myFile.read() ## Read the file into one long string
    myFile.close() ## close the file
    lineString = lineString.strip()   ## remove extra whitespace
    lineList = lineString.split('\n') ## Split the string
    print('\nMotifs:')
    for pattern in lineList:
        print(pattern)
    return lineList
```

Download `toy-motifs.txt` from Moodle and convert the file into a list of strings.

Compute Nucleotide Counts

Given a list of strings of length k, calculate the number if A's, C's, G's, and T's in each column. Return the counts as a table with four rows (one for each nucleotide) and k columns (one for each position). **Important: make sure the nucleotides are alphabetized**, that is the first row captures the counts of As, the second row C's, the third row G's, and the last row T's.

`computeCounts()`: *Computes the numbers of A's, C's, G's, and T's for every column in a list of binding sites.*
Inputs: List of strings representing binding sites.
Returns: A list of lists containing the counts of A/C/G/T for each position.

Compute Nucleotide Frequencies

Given a table of counts, *normalize* each column by dividing each value in `countMat` by the number of binding sites. This is the final frequency matrix (or "profile").

`computeFrequencies()`: *Computes the proportion of A's, C's, G's, and T's for every column in a list of binding sites.*
Inputs: A list of lists containing the counts of A/C/G/T for each position
Returns: A list of lists containing the proportion of A/C/G/T for each position.

Compute the Consensus String

Given the frequency table of nucleotides, compute the **consensus** string of the motifs. Again, be sure that the the first row captures the percent of As, the second row C's, the third row G's, and the last row T's.

Compute the Motif Score

Compute the **score** of the motifs. There are multiple ways to do this: you can sum the differences in the columns, sum the differences in the rows (using the Hamming distance), or design your own method. You need to decide what inputs and outputs you need.

`computeScore()`: *Computes the score of a list of binding sites.*
Inputs: ?
Returns: ?

Visualize the Patterns for NF-κB and CAP Binding Sites

I have provided a function `plotMotif()` that makes a PNG file showing the proportion of nucleotides by position. Call this function with the frequency table (returned from `computeFrequencies()`) and a filename string:

```
plotMotif(frequencyTable,'toy-frequencies.png')
```

Do the same with `nfkb-motifs.txt` (the example from class) and `cap-motifs.txt` (another binding motif).