Motif Finder Algorithm

In this Jupyter Notebook, you will be writing functions to search for an inputted motif within protein sequences. We will be giving Python some protein sequences in a FASTA file, asking the user for an amino acid sequence (single-letter), then using regular expressions to search for the amino acid pattern within the FASTA sequences. This will require some knowledge of regular expressions and how to write your own functions.

We will be walking through this project in multiple parts:

- **Part 0: Libraries** will introduce you to some very useful packages containing some very useful functions. Many of them will be familiar.
- **Part 1: `FindMotif`** will require you to fill in and complete a function. This function will use regular expressions to search for a short motif pattern within a larger sequence of characters (in this case, an one-letter amino acid sequence).
- **Part 2: `FastatoCSV`** will require you to fill in and complete a function. This function will do three things at once: open the data file containing our FASTA sequence, use the `FindMotif` function that you completed in Part 1, then write the results to a CSV file.
- **Part 3: Putting It All Together** will show you some ways to make the code and functions user-friendly.

The following code is courtesy of Dr. Hall's lab. I've made some modifications to adapt it to Python 3. I've also made some changes that will require you to do some coding yourself.

A reminder

As you start working through this notebook, I'll remind you of SSP's policies on collaborating on code together:

- You may ask your fellow participants for help. You may show them your computer screen and any error messages that arise. **However:** the person helping you cannot assume control of your keyboard and type anything.
- Likewise, **you are not allowed to copy-and-paste and/or send code to anyone else. You are also not allowed to copy-and-paste any code you find online, unless you give it proper citation and acknowledgement.**
- You may not show your working code as an "example" to anyone else.

Part 0: Libraries

First, we'll need to import some libraries and packages in order to use some nice functions. We'll use the following libraries:

- `re` : Regular expression operations; sequence of characters that define a search pattern
- `csv` : Comma Separated Values; allows for import and export for spreadsheets and databases
- `Bio` : Biopython; a set of tools for biological computation. A very powerful package that you are sure to encounter in biological research
- `SeqIO` : Sequence Input/Output; allows for input/output of assorted sequence file formats

In [ ]:
```
import re, csv
from Bio import SeqIO
```

Part 1: `FindMotif`

At the core of this project is a function that will search for a short motif within a longer protein sequence. This is where our old friend, regex, will come in handy.

Suppose we have the user give us a short amino acid sequence, called `motif` . (We'll worry about how to get this sequence later.)
We'll want to compile the entered motif into a more "friendly" format. Below we've entered code which will replace any " `X` " with a " `\w` ", which is a regular expression for any alpha-numeric character. It is your responsibility to **further clean the motif by:**

- replacing any "x" (lowercase) with a "\w"
- replacing any "-" with empty quotes
- replacing any newline ("\n") with empty quotes

In [ ]:
```
motif = "S-P-x-X"
motif = re.sub("X", "\\w", motif)

# Your Code After This Line

print(motif)
```

Next, we'll use the `compile` function to turn our cleaned motif into a variable called `needle`. (As in, this is the `needle` in the haystack we will be searching for.)

We should also assume that the user has given us a FASTA file that contains our protein sequence, called `seq`. We'll worry about getting this later. For now, we'll use a short "test" protein sequence.
We'll also use the `findall` function to look for the `needle` within the `seq` (our haystack).

You don't need to fill anything in in the following code, or run it.

In [ ]:
```
needle = re.compile(str(motif_clean))

# Scans from left-to-right and returns matches
# IF it finds the motif (needle) in the protein sequence (seq) then perform th
e following...
if needle.findall(seq):

    # Creates an empty matrix to store the matches in
    group_matches=[]
    position_matches=[]

    # For EACH 'match' in the protein sequence (seq), find the motif instances
over the protein sequences
    for match in needle.finditer(seq):

        # Appends the matching group (motif) and matching span (position in pr
otein sequence)
        # to the empty matrix (position_matches) created earlier
        group_matches.append(match.group(0))

        position_matches.append(match.start(0)+1)
        print("Position matches found.")
```

Notice that we supposed that the `needle` was indeed found in the `seq`. If it wasn't found, then we would want to return something like `NA` to signify that nothing was found.

Finish the Function

Below, you'll find some beginnings of a function. I've entered in comments signifying where to enter code, and what it should look like. Your responsibility is to **use the code from above** to fill out this function. Be sure to keep **indentation** in mind.

In [ ]:
```
# FindMotif; Function that finds the inputted motif within the protein sequenc
es in the FASTA file
def FindMotif(motif,seq):

    # Clean the motif using regex.
    motif_clean = ... # [ADD CODE HERE]

    needle = re.compile(str(motif_clean))
    # use findall to search for any matches
    if needle.findall(seq):
        # What happens when the needle is found in the sequence?
        # [ADD CODE HERE]

        return (group_matches, position_matches, len(group_matches))
    else:
        # Returns "NA", "NA" because there are no matches in the sequence to r
eport
        return (["NA"],["NA"],0)
```

Part 2: `FastatoCSV`

Before we can start using protein sequences from FASTA files, we need to do some special preparation to the FASTA file in order to read it correctly. Here, we'll choose to open it as a CSV file.
We've uploaded a file for you called `test_file.fasta` . It's a FASTA sequence of *Candida glabrata* CDC14.
Remember that the following code will open it as a CSV file:

In [ ]:
```
fasta_file = "test_file.fasta"

# Opens the FASTA file using the built-in python function 'open'
handle = open(fasta_file, "r")
```

We'll also need a new, blank CSV file to save our results in. The following code will open a new CSV file called `motif search results.csv` , and write in a "header" of information. This "header" will tell us the information that is held in each column of the spreadsheet.

In [ ]:
```
# Remember what "with open( ) as" does?
with open ("motif search results.csv", "w") as csvfile:

    # Assigns headers to the CSV output
    # Creates a generic file writer (csv.writer) that writes to csvfile
    writer = csv.writer(csvfile)

    # Header values for our CSV output
    header = ['Protein ID', 'Motifs', 'Positions', 'Number of Matches']

    # Write the header values into the empty CSV file output
    writer.writerow(header)
```

Once we open this CSV file, we can use our `FindMotif` function to search for motifs within the sequence, and write in the results one row at a time.
Remember that we saved our FASTA sequence into a variable called `handle`. This variable name `handle` will stand in for our FASTA sequence each time we want to refer to it.

In [ ]:
```
# For each protein ('record') in the FASTA file (handle, format 'fasta')...
# Parse (break into subcomponents) using the known FASTA format
for record in SeqIO.parse(handle, "fasta"):

    # This a function that writes data into the new CSV file
    # We'll call it "spamwriter" (as in "green eggs and spam")
    # Allows you to set the delimiter and the quote character to allow for spe
cial characters
    spamwriter = csv.writer(csvfile, delimiter=',', quotechar='"')

    # Stores the output of FindMotif using the inputs 'motif_input' and...
    # record.seq (from line 49) all of the protein sequences as strings (str)
    motif_row = FindMotif(motif_input, str(record.seq))

    # Using the CSV writer (spamwriter) it writes the Protein (record.id)...
    # the Sequence (record.seq), and the output of FindMotif
    # [0] the matching sequence motif, and [1] the number of matches

    for i in range(0,motif_row[2]):

        spamwriter.writerow([record.id, motif_row[0][i], motif_row[1][i], moti
f_row[2]])
```

We should always remember to close the CSV file that we opened:

In [ ]:
```
# Closes the active CSV file
csvfile.close()
```

Finish the Function

Below, you'll find some beginnings of a function. I've entered in comments signifying where to enter code, and what it should look like. Your responsibility is to **use the code from above** to fill out this function. Be sure to keep **indentation** in mind.

In [ ]:
```
# Preparing the FASTA file as a CSV file
def FastatoCSV (fasta_file):

    # Open the FASTA file
    # Opens the FASTA file using the built-in python function 'open'
    # [ADD CODE HERE]

    # Open a new, blank CSV file called 'motif search results.csv' and write i
n its header
    # [ADD CODE HERE]

        # Since we have an input file with multiple sequences, use a for loop
 to
        # use the FindMotif function to search for motifs in a handle,
        # then our open CSV file to write in the results
        # [ADD CODE HERE]


    # Remember to close the active CSV file!
    # [ADD CODE HERE]

    # Add a print statement so we can know when the function is done
    print("Done!")
```

Note: Just because Python willfully says "Done!", doesn't necessarily mean that Python did what you wanted it to do. It may be misunderstanding what you are trying to tell it altogether. Be cautious with indentations, the order of your code, and any code you are omitting or including.

Part 3: Putting It All Together

We're very close to being able to use our functions! The last step is to make it user-friendly.

Remember how I said that would worry about getting a motif input and sequence later? We'll do that now.

Python has a useful function called `input` that will ask the user for an input. We can use that function to get the amino acid sequence that we want to search for. **Through the power of regular expressions, we can also add in any "wild card" characters by using an `X` , or allow a choice of a few amino acids for any single position.**

It would be nice for the user to understand what kind of input the functions will take. So we'll also also print out some nice instructions and tips for the user.

In [ ]:
```
# General instructions for the user on the required input format
print("This script needs a motif input in a regex format.")
print("The amino acids must be in single letter format.")
print("Example: S-P-x-[KR], where [KR] means either K or R. For any character, use *")

# Sequence input; assigns "motif_input" to a value;
# The input function requests information from the user.
motif_input = input("Input the motif:")
```

We can also use the `input` function to get the name of the file where the user has some FASTA sequences stored.

In [ ]:
```
# Input the FASTA file with the sequences.
# It must be in the same folder of your Python Script.
fasta_input = input("Input the FASTA file name:")
```

The Big Reveal!

Run the following cell to use the functions you wrote and test it out!

In [ ]:
```
# Runs the FastatoCSV function using the inputted FASTA file (fasta_input)
# It will search for the inputted motif (motif_input)
FastatoCSV(fasta_input)
```

If your code ran, you should see a file called `motif search results.csv` in your the same folder as your Python notebook. Be sure to open the file and check it to see that the output is what you expected.

Part 4 (Optional): Bells and Whistles

If you feel comfortable with the above code and want to add some extra functionality, here are some options to add to your motif search algorithm. Using the code that has already been constructed for you, modify it or add extra functions to:

- Let a user type in "basic" or "acid" to represent any basic or acidic residue, for instance
- Display the extra amino acids on either side (~2 or ~3 residues) of a detected motif
- Create a heatmap or some other visual representation of your results, and display it in the Jupyter Notebook
- Sort and rank the hits before writing your results to the CSV
- Add more options for user input, such as opening a pop-up to navigate through directories and select the FASTA file

Great work, and good luck!