Version 1.00

Assignment 1
DNA Editing

1. Submission Guidelines

Deadline:	11:59 PM on Thursday 11 November 2021
Submission procedure:	Submit only one file labelled `dna.py` through blackboard (via TurnItIn)
Version requirement:	Your code must run using the **Python 3.10.0 IDLE on a PC**
Allowable import modules:	No importing of module is allowed for this assignment.

2. Overview

Write Python code in `dna.py`, which contains functions that process nucleic acid code sequences. Your code should be importable into another Python script (e.g., `script.py`). However, your code, `dna.py`, should not import any module into itself.

3. FNA format

A sequence in the FNA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol at the beginning. All lines of text must be less than or equal to 80 characters in length. An example sequence in the FNA format is:

```
>fragment sequence of BRCA1.fna
GAGTCCCGGGAAAGGGACAGGGGGCCCAAGTGATGCTCTGGGGTACTGGCGTGGGAGAGTGGATTTCCGAAGCTGACAGA
TGGGTATTCTTTGACGGGGGGTAGGGGCGGAACCTGAGAGGCGTAAGGCGTTGTGAACCCTGGGGAGGGGGGCAGTTTGT
AGGTCGCGAGGGAAGCGCTGAGGATCAGGAAGGGGGCACTG
```

Blank lines are not allowed in the middle of FASTA input. Also, assume that there is only 1 sequence in each `.fna` file.

Sequences are expected to be represented in the standard IUB/IUPAC nucleic acid codes, with these exceptions: lower-case letters are accepted and are mapped into upper-case; and a single hyphen or dash can be used to represent a gap of indeterminate length.

The **17 nucleic acid codes**:

A	adenosine	C	cytidine	G	guanine
T	thymidine	N	A/G/C/T (any)	U	uridine
K	G/T (keto)	S	G/C (strong)	Y	T/C (pyrimidine)
M	A/C (amino)	W	A/T (weak)	R	G/A (purine)
B	G/T/C	D	G/A/T	H	A/C/T
V	G/C/A	-	gap of indeterminate length		

4. DNA functions

Write 9 functions that can be used with the import system. We will type `import dna` at the top of our own script (eg, `script.py`) and then use your functions to perform tasks. All code examples described here assume that `import dna` has been executed prior to each function call. While we are providing example code and `.fna` files here and on blackboard, we will mark your code using a far wider range of test code and test `.fna` files, which are not being provided to you.

1 *sequence*, *description* = **load**(*filename*)

Take the filename of a `.fna` file and return the DNA sequence and description line.

The *filename* argument should be of type `str` and is the path to the `.fna` file.

The returned *sequence* should be a linear array of type `list` that contains values of the type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

The returned *description* should be a string of type `str`. It should contain the description line without the '>' character and without ? leading or trailing whitespace characters.

Further details:

- If *filename* is not a `.fna` file or if the file could not be opened, the returned *sequence* should be an empty list and the returned *description* should be `'file was not loaded'`.
- If the `.fna` files includes nucleic acid codes that are in lower case, convert them to upper case in *sequence*.
- If the `.fna` file includes alpha-numeric characters (`0-9`, `a-z`, `A-Z`) that are not nucleic acid codes, they should still be transferred to *sequence*. If the character is lower case, convert it to upper case.

The following code...

```
seq, info = dna.load('BRCA1.fna')
print(info)
print(seq[0:80])
```

should produce the following output to the console:

```
672 17:43044295-43125364
['G', 'C', 'T', 'G', 'A', 'G', 'A', 'C', 'T', 'T', 'C', 'C', 'T', 'G', 'G', 'A',
'C', 'G', 'G', 'G', 'G', 'G', 'A', 'C', 'A', 'G', 'G', 'C', 'T', 'G', 'T', 'G',
'G', 'G', 'G', 'T', 'T', 'T', 'C', 'T', 'C', 'A', 'G', 'A', 'T', 'A', 'A', 'C',
'T', 'G', 'G', 'G', 'C', 'C', 'C', 'C', 'T', 'G', 'C', 'G', 'C', 'T', 'C', 'A',
'G', 'G', 'A', 'G', 'G', 'C', 'C', 'T', 'T', 'C', 'A', 'C', 'C', 'C', 'T', 'C']
```

2 *table* = **stats**(*sequence*)

Take a sequence and return a table that includes the number of times a nucleic acid code occurs.

The *sequence* argument should be a linear array of type `list` that contains values of the type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

以外 others?

The returned *table* should be of type `dict`. It should contain only 18 entries of type `str`: 17 nucleic acid codes listed in the FNA format section (Section 3), and `'other'` to represent a character that is not one of the 17 nucleic acid codes. Each entry represents the number of times a nucleic acid code has appeared in the *sequence*. If a character is not one of the 17 nucleic acid codes, count it in the `'other'` entry.

The following code...

```
table = dna.stats(seq)
print(table)
```

should produce the following output to the console:

```
{'A': 22752, 'C': 16928, 'G': 17864, 'T': 23526, 'N': 0, 'U': 0, 'K': 0, 'S': 0,
'Y': 0, 'M': 0, 'W': 0, 'R': 0, 'B': 0, 'D': 0, 'H': 0, 'V': 0, '-': 0, 'other':
0}
```

3 *formatted_sequences* = **format_sequence**(*sequence*, *first_index*, *last_index*)

Take a sequence along with two indices and return the subsequence with a particular format.

The *sequence* argument should be a linear array of type `list` that contains values of type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

The *first_index* and *last_index* arguments represent the first and last nucleic acid codes that should be extracted from *sequence*.

The returned *formatted_sequences* should be of type `list` that contains values of type `str`. However, rather than each entry being a single nucleic acid code, each entry should be a `str` type with up to 80 nucleic acid codes.

The following code...

รวม300

```
data = dna.format_sequence(seq,100,300)
print(data)
```

should produce the following output to the console:

160+41

```
['GAGTCCCGGGAAAGGGACAGGGGGCCCAAGTGATGCTCTGGGGTACTGGCGTGGGAGAGTGGATTTCCGAAGCTGACAGA',
'TGGGTATTCTTTGACGGGGGGTAGGGGCGGAACCTGAGAGGCGTAAGGCGTTGTGAACCCTGGGGAGGGGGCAGTTTGT',
'AGGTCGCGAGGGAAGCGCTGAGGATCAGGAAGGGGGCACTG']
```

4 **write**(*filename*, *description*, *sequence*, *first_index*, *last_index*)

Take a description, sequence, and sequence range, and write to a .fna file.

The *filename* argument should be of type `str` and is the path to the `.fna` file. If the file exists, overwrite it.

The *description* argument should be of type `str` and contain the description line text to write to the `.fna` file.

The *sequence* argument should be a linear array of type `list` that contains values of the type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

The *first_index* and *last_index* arguments represent the first and last codes that should be written from *sequence*.

The following code...

```
dna.write('BRCA1_subseq.fna',
          'fragment sequence of BRCA1.fna',
          seq,100,300)
```

should produce the following `BRCA1_subseq.fna` file.

```
>fragment sequence of BRCA1.fna
GAGTCCCGGGAAAGGGACAGGGGGCCCAAGTGATGCTCTGGGGTACTGGCGTGGGAGAGTGGATTTCCGAAGCTGACAGA
TGGGTATTCTTTGACGGGGGGTAGGGGCGGAACCTGAGAGGCGTAAGGCGTTGTGAACCCTGGGGAGGGGGCAGTTTGT
AGGTCGCGAGGGAAGCGCTGAGGATCAGGAAGGGGGCACTG
```

5 *matches* = **find**(*sequence*, *sequence_to_find*)

Find a sequence within another sequence and record the indices where they occurred.

The *sequence* and *sequence_to_find* arguments should be linear arrays of type `list` containing values of type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

The returned *matches* should be a linear array of type `list`, which contains values of type `int`. It should contain the indices where the matches were found. This should include overlapping occurrences.

The following code...

```
seq = list('AAAGTTAAATAATAAATAGGTGAA')
seq_to_find = list('AAA')
matches = dna.find(seq,seq_to_find)
print(matches)
```

should produce the following output to the console:

```
[0, 6, 13]
```

6 *new_sequence* = **add**(*sequence*, *sequence_to_add*, *index*)

Add a sequence into an existing sequence at a specified index.

The *sequence* and *sequence_to_add* arguments should be linear arrays of type `list` containing values of type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

The *index* argument should be of type `int`. *sequence_to_add* should be placed before *index* within *sequence*. 1 If *index* specifies a location beyond *sequence*, then add *sequence_to_add* to the end of *sequence*.

The returned *new_sequence* should be the updated sequence.

The following code...

```
line = '';
seq = list('AAAGTTAAATAATAAATAGGTGAA')
print(line.join(seq))
seq = dna.add(seq,list('NNNNN'),5)
print(line.join(seq))
seq = dna.delete(seq,6,4)
print(line.join(seq))
seq = dna.replace(seq,list('HHH'),5, 1)
print(line.join(seq))
```

should produce the following output to the console:

```
AAAGTTAAATAATAAATAGGTGAA
AAAGTNNNNNTAAATAATAAATAGGTGAA
AAAGTNTAAATAATAAATAGGTGAA
AAAGTHHHTAAATAATAAATAGGTGAA
```

new_sequence = **delete**(*sequence*, *index*, *number_of_codes*)

Delete a subsequence from a sequence as specified by a starting index and the number of codes to delete.

The *sequence* argument should be a linear array of type `list` that contains values of type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

The *index* argument should specify the first index, from which to delete nucleic acid codes. The *number_of_codes* should specify the number of nucleic acid codes to delete from the *index*. 1 If the *index* and *number_of_codes* specify a subsequence that exceeds the length of the *sequence*, then delete nucleic acid codes until the end of that sequence. 2 If the *index* specifies a sequence index that is out of bounds, then do not delete anything.

The returned *new_sequence* should be the updated sequence.

new_sequence = **replace**(*sequence*, *sequence_to_add*, *index*, *number_of_codes*)

Replace a section of a sequence with a new subsequence.

The *sequence* and *sequence_to_add* arguments should be linear arrays of type `list` containing values of type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

The *index* argument should specify the first index, from which to delete nucleic acid codes. *sequence_to_add* should also be placed before *index* within *sequence*. The *number_of_codes* argument should specify the number of nucleic acid codes to delete from the *index*. 1 If the *index* and *number_of_codes* specify a subsequence that exceeds the length of the *sequence*, then delete nucleic acid codes until the end of that sequence. 2 If the *index* specifies a sequence index that is out of bounds, then add *sequence_to_add* to the end of the sequence.

The returned *new_sequence* should be the updated sequence.

protein_sequence, table = **dna2protein**(*dna_sequence*)

Convert the DNA sequence to its corresponding protein sequence.

The *dna_sequence* argument should be a linear array of type `list` that contains values of type `str`. Each value in the list should be one of the 17 nucleic acid codes listed in the FNA format section (Section 3).

The returned *protein_sequence* should be a linear array of type `list` that contains values of the type `str`. Each value in the list should be a single character of the protein codes specified in `dna2protein.csv` or '?' if the 3-letter nucleic acid code could not be deciphered. If there are remaining nucleic acid codes that are not read, do not return a character into *protein_sequence*.

The returned *table* should be a dictionary of type `dict`, which contains the 3-letter nucleic acid code and its corresponding protein symbol. This table should be extracted from the `dna2protein.csv` file. In addition to the `.csv` value, include a 3-letter nucleic acid code entry '`???`', which maps to the character '?'.

The following code...

```
dna_seq = list('AAAGTTAAATAATAAATAGGTGAA')
pro_seq, table = dna.dna2protein(dna_seq)
print(pro_seq)
print(table)
```

should produce the following output to the console:

```
{'TTT': 'F', 'TTC': 'F', 'TTA': 'L', 'TTG': 'L', 'CTT': 'L', 'CTC': 'L', 'CTA':
'L', 'CTG': 'L', 'ATT': 'I', 'ATC': 'I', 'ATA': 'I', 'ATG': 'M', 'GTT': 'V',
'GTC': 'V', 'GTA': 'V', 'GTG': 'V', 'TCT': 'S', 'TCC': 'S', 'TCA': 'S', 'TCG':
'S', 'CCT': 'P', 'CCC': 'P', 'CCA': 'P', 'CCG': 'P', 'ACT': 'T', 'ACC': 'T',
'ACA': 'T', 'ACG': 'T', 'GCT': 'A', 'GCC': 'A', 'GCA': 'A', 'GCG': 'A', 'TAT':
'Y', 'TAC': 'Y', 'TAA': 'X', 'TAG': 'X', 'CAT': 'H', 'CAC': 'H', 'CAA': 'Q',
'CAG': 'Q', 'AAT': 'N', 'AAC': 'N', 'AAA': 'K', 'AAG': 'K', 'GAT': 'D', 'GAC':
'D', 'GAA': 'E', 'GAG': 'E', 'TGT': 'C', 'TGC': 'C', 'TGA': 'X', 'TGG': 'W',
'CGT': 'R', 'CGC': 'R', 'CGA': 'R', 'CGG': 'R', 'AGT': 'S', 'AGC': 'S', 'AGA':
'R', 'AGG': 'R', 'GGT': 'G', 'GGC': 'G', 'GGA': 'G', 'GGG': 'G', '???': '?'}
```

5. Coding rules

Do not declare any variables in the global space of `dna.py`.

Do not import any module in `dna.py`.

In the same folder, place the following files:

- `dna.py` (your code)
- `dna2protein.csv`
- your `.fna` files

However, only submit your `dna.py` file to us.

6. Marking criteria

We will mark your submitted `dna.py` code according to the following categories:

(1) Implementation and evidence of coding knowledge (majority of your marks)

(2) Coding efficiency

(3) Coding style and commenting

We will use `import dna` near the top of our script to test your code. We will run several test conditions against each of your functions. This includes function parameters and `.fna` files that have not been provided to you.