Reading and writing from files

In [4]:
```
### open function and file objects
f = open("/Users/pmagwene/Downloads/covid-ref.fsa")
```

In [ ]:
```
### reading everything from a file as a single string
```

In [5]:
```
s = f.read()
```

In [7]:
```
len(s)
```

Out[7]:
```
30429
```

In [ ]:
```
## reading a file line by line
```

In [16]:
```
f2 = open("/Users/pmagwene/Downloads/covid-ref.fsa")
```

In [17]:
```
lines = [line for line in f2.readlines()]
```

In [ ]:
```

```

In [20]:
```
lines[1]
```

Out[20]:
```
'ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATCTGTTCTCTAAA\n'
```

In [ ]:
```
### reading a file line by line
```

Write a function to parse a FASTA file

Introducing the FASTA file format for sequence data

The most ubiquitous file format used to represent nucleotide and protein sequence data is the FASTA format. Wikipedia has a good overview of the FASTA format (https://en.wikipedia.org/wiki/FASTA_format). We'll illustrate this format with an example -- the COVID-19 reference genome, which can be found on Genbank (https://www.ncbi.nlm.nih.gov/genbank/) via this link (https://www.ncbi.nlm.nih.gov/nuccore/NC_045512).

Summary of FASTA format:

- Each file can hold one or more sequence records
- The beginning of each record is delimited by a line called a header, which has a `>` character at the beginning, followed by the name associated with that record (and an optional description). For example `>seq1 Involved in...` would indicate the beginning of a record with the name `seq1` and the description "Involved in...".
- On or more sequence lines follow header lines. These lines are usually wrapped to have length <=80 characters but this is not required.

In [8]:
```python
def parse_FASTA(fname):
    record_dict = {}
    f = open(fname, 'r')

    recname = ""            # will hold names of records
    seq = ""                # will hold seq strings
    active_record = False  # indicates whether we are currently working on building a record

    for line in f.readlines():

        line = line.strip()  # strip any whitespace at beginning/end of line

        if line == "":       # empty line
            continue         # go to next iteration of for loop


        if line[0] == ">":                 # are we dealing with a new record?
            if active_record:              # did we already have an active record?
                record_dict[recname] = seq # if so, add to old active record to the dict so we can
                                           # begin a new one

            recname = line[1:].split()[0]  # name of new record
            seq = ""                       # reset variable holding the string
            active_record = True           # set flag to indicate we now have an active record
            continue                       # go to the next iteration of for loop, as there's noth

        seq += line

    if active_record:                 # if we've exhausted all the lines, we might still have an act
        record_dict[recname] = seq  # if so, add it to the dict

    return record_dict
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Using our parse_FASTA function

Download the following files to your computer:

- `covid-ref.fsa` (https://github.com/bio208fs-class/bio208fs-lecture/raw/master/data/covid-ref.fsa)
- `covid-S-and-E.fsa` (https://github.com/bio208fs-class/bio208fs-lecture/raw/master/data/covid-S-and-E.fsa)

Assignment 01

Using codeblocks, solve the following problems:

a) Show how to use the `parse_FASTA` function to read the file `covid-S-and-E.fsa` file. (1 pt)

In [9]:

```python
# code for 1a
x = parse_FASTA("/Users/cleve/OneDrive/Documents/Notes/Notebooks Linked Materials/Duke University/
print(x)
```

```
{'YP_009724390.1': 'MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSSVLHSTQDLFLPFFSNVTWFHAIHVSGTNGTK
RFDNPVLPFNDGVYFASTEKSNIIRGWIFGTTLDSKTQSLLIVNNATNVVIKVCEFQFCNDPFLGVYYHKNNKSWMESEFRVYSSANNCTFEYVSQP
FLMDLEGKQGNFKNLREFVFKNIDGYFKIYSKHTPINLVRDLPQGFSALEPLVDLPIGINITRFQTLLALHRSYLTPGDSSSGWTAGAAAYYVGYLQ
PRTFLLKYNENGTITDAVDCALDPLSETKCTLKSFTVEKGIYQTSNFRVQPTESIVRFPNITNLCPFGEVFNATRFASVYAWNRKRISNCVADYSVL
YNSASFSTFKCYGVSPTKLNDLCFTNVYADSFVIRGDEVRQIAPGQTGKIADYNYKLPDDFTGCVIAWNSNNLDSKVGGNYNYLYRLFRKSNLKPFE
RDISTEIYQAGSTPCNGVEGFNCYFPLQSYGFQPTNGVGYQPYRVVVLSFELLHAPATVCGPKKSTNLVKNKCVNFNFNGLTGTGVLTESNKKFLPF
QQFGRDIADTTDAVRDPQTLEILDITPCSFGGVSVITPGTNTSNQVAVLYQDVNCTEVPVAIHADQLTPTWRVYSTGSNVFQTRAGCLIGAEHVNNS
YECDIPIGAGICASYQTQTNSPRRARSVASQSIIAYTMSLGAENSVAYSNNSIAIPTNFTISVTTEILPVSMTKTSVDCTMYICGDSTECSNLLLQY
GSFCTQLNRALTGIAVEQDKNTQEVFAQVKQIYKTPPIKDFGGFNFSQILPDPSKPSKRSFIEDLLFNKVTLADAGFIKQYGDCLGDIAARDLICAQ
KFNGLTVLPPLLTDEMIAQYTSALLAGTITSGWTFGAGAALQIPFAMQMAYRFNGIGVTQNVLYENQKLIANQFNSAIGKIQDSLSSTASALGKLQD
VVNQNAQALNTLVKQLSSNFGAISSVLNDILSRLDKVEAEVQIDRLITGRLQSLQTYVTQQLIRAAEIRASANLAATKMSECVLGQSKRVDFCGKGY
HLMSFPQSAPHGVVFLHVTYVPAQEKNFTTAPAICHDGKAHFPREGVFVSNGTHWFVTQRNFYEPQIITTDNTFVSGNCDVVIGIVNNTVYDPLQPE
LDSFKEELDKYFKNHTSPDVDLGDISGINASVVNIQKEIDRLNEVAKNLNESLIDLQELGKYEQYIKWPWYIWLGFIAGLIAIVMVTIMLCCMTSCC
SCLKGCCSCGSCCKFDEDDSEPVLKGVKLHYT', 'YP_009724392.1': 'MYSFVSEETGTLIVNSVLLFLAFVVFLLVTLAILTALRLCAYC
CNIVNVSLVKPSFYVVYSRVKNLNSSRVPDLLV'}
```

b) Show how to use list comprehension to get the gene names of each of the records in `covid-S-and-E.fsa` (1 pt)

In [25]:

```python
# code for 1b
gene_name = [print(key) for key in x]
```

```
YP_009724390.1
YP_009724392.1
```

c) Show how to get the sequence corresponding to the gene with the name `YP_009724390.1` (1 pt)

In [211]:

```python
# code for 1c
print(x['YP_009724390.1'])
```

```
MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSSVLHSTQDLFLPFFSNVTWFHAIHVSGTNGTKRFDNPVLPFNDGVYFASTEK
SNIIRGWIFGTTLDSKTQSLLIVNNATNVVIKVCEFQFCNDPFLGVYYHKNNKSWMESEFRVYSSANNCTFEYVSQPFLMDLEGKQGNFKNLREFVF
KNIDGYFKIYSKHTPINLVRDLPQGFSALEPLVDLPIGINITRFQTLLALHRSYLTPGDSSSGWTAGAAAYYVGYLQPRTFLLKYNENGTITDAVDC
ALDPLSETKCTLKSFTVEKGIYQTSNFRVQPTESIVRFPNITNLCPFGEVFNATRFASVYAWNRKRISNCVADYSVLYNSASFSTFKCYGVSPTKLN
DLCFTNVYADSFVIRGDEVRQIAPGQTGKIADYNYKLPDDFTGCVIAWNSNNLDSKVGGNYNYLYRLFRKSNLKPFERDISTEIYQAGSTPCNGVEG
FNCYFPLQSYGFQPTNGVGYQPYRVVVLSFELLHAPATVCGPKKSTNLVKNKCVNFNFNGLTGTGVLTESNKKFLPFQQFGRDIADTTDAVRDPQTL
EILDITPCSFGGVSVITPGTNTSNQVAVLYQDVNCTEVPVAIHADQLTPTWRVYSTGSNVFQTRAGCLIGAEHVNNSYECDIPIGAGICASYQTQTN
SPRRARSVASQSIIAYTMSLGAENSVAYSNNSIAIPTNFTISVTTEILPVSMTKTSVDCTMYICGDSTECSNLLLQYGSFCTQLNRALTGIAVEQDK
NTQEVFAQVKQIYKTPPIKDFGGFNFSQILPDPSKPSKRSFIEDLLFNKVTLADAGFIKQYGDCLGDIAARDLICAQKFNGLTVLPPLLTDEMIAQY
TSALLAGTITSGWTFGAGAALQIPFAMQMAYRFNGIGVTQNVLYENQKLIANQFNSAIGKIQDSLSSTASALGKLQDVVNQNAQALNTLVKQLSSNF
GAISSVLNDILSRLDKVEAEVQIDRLITGRLQSLQTYVTQQLIRAAEIRASANLAATKMSECVLGQSKRVDFCGKGYHLMSFPQSAPHGVVFLHVTY
VPAQEKNFTTAPAICHDGKAHFPREGVFVSNGTHWFVTQRNFYEPQIITTDNTFVSGNCDVVIGIVNNTVYDPLQPELDSFKEELDKYFKNHTSPDV
DLGDISGINASVVNIQKEIDRLNEVAKNLNESLIDLQELGKYEQYIKWPWYIWLGFIAGLIAIVMVTIMLCCMTSCCSCLKGCCSCGSCCKFDEDDS
EPVLKGVKLHYT
```

d) Show how to use a single for-loop to create two lists containing the name of each protein in `covid-S-and-E.fsa` and a corresponding list giving the length of each of those proteins (2 pts)

In [212]:
```
# code for 1d
a = []
b= []
for i in x:
    a.append(i)
    b.append(len(x[i]))
print(a)
print(b)
```

```
['YP_009724390.1', 'YP_009724392.1']
[1273, 75]
```

Assignment 02

Using codeblocks, solve the following problems:

a) Refer to the GenBank record page for the COVID-19 reference genome (https://www.ncbi.nlm.nih.gov/nuccore/NC_045512). What are the genome coordinates for the coding sequence for the "S" (spike) gene? What are the genome coordinates for the coding sequence of the "E" (envelope) gene? Assign the start and stop coordinates for each of these genes to variables with appropriate names (1 pt)

In [213]:
```
# code for 2a
s_gene_start = 21563
s_gene_stop = 25384
e_gene_start = 26245
e_gene_stop = 26472
```

b) Using the DNA nucleotide sequence you loaded from the `covid-ref.fsa` file, show how to retrieve the nucleotide sequences corresponding to the "S" and "E" genes. Remember that Python strings are 0-indexed, whereas GenBank using 1-index coordinates (3 pts)

In [214]:
```
# code for 2b
y = parse_FASTA("/Users/cleve/OneDrive/Documents/Notes/Notebooks Linked Materials/Duke University/
print(y[key][s_gene_start-1:s_gene_stop-1])
print()
print(y[key][e_gene_start-1:e_gene_stop-1])
```

```
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGCATACACTAATTCTTTCA
CACGTGGTGTTTATTACCCTGACAAAGTTTTCAGATCCTCAGTTTTACATTCAACTCAGGACTTGTTCTTACCTTTCTTTTCCAATGTTACTTGGTT
CCATGCTATACATGTCTCTGGGACCAATGGTACTAAGAGGTTTGATAACCCTGTCCTACCATTTAATGATGGTGTTTATTTTGCTTCCACTGAGAAG
TCTAACATAATAAGAGGCTGGATTTTTGGTACTACTTTAGATTCGAAGACCCAGTCCCTACTTATTGTTAATAACGCTACTAATGTTGTTATTAAAG
TCTGTGAATTTCAATTTTGTAATGATCCATTTTTGGGTGTTTATTACCACAAAAACAACAAAAGTTGGATGGAAAGTGAGTTCAGAGTTTATTCTAG
TGCGAATAATTGCACTTTTGAATATGTCTCTCAGCCTTTTCTTATGGACCTTGAAGGAAAACAGGGTAATTTCAAAAATCTTAGGGAATTTGTGTTT
AAGAATATTGATGGTTATTTTAAAATATATTCTAAGCACACGCCTATTAATTTAGTGCGTGATCTCCCTCAGGGTTTTTCGGCTTTAGAACCATTGG
TAGATTTGCCAATAGGTATTAACATCACTAGGTTTCAAACTTTACTTGCTTTACATAGAAGTTATTTGACTCCTGGTGATTCTTCTTCAGGTTGGAC
AGCTGGTGCTGCAGCTTATTATGTGGGTTATCTTCAACCTAGGACTTTTCTATTAAAATATAATGAAAATGGAACCATTACAGATGCTGTAGACTGT
GCACTTGACCCTCTCTCAGAAACAAAGTGTACGTTGAAATCCTTCACTGTAGAAAAAGGAATCTATCAAACTTCTAACTTTAGAGTCCAACCAACAG
AATCTATTGTTAGATTTCCTAATATTACAAACTTGTGCCCTTTTGGTGAAGTTTTTAACGCCACCAGATTTGCATCTGTTTATGCTTGGAACAGGAA
GAGAATCAGCAACTGTGTTGCTGATTATTCTGTCCTATATAATTCCGCATCATTTTCCACTTTTAAGTGTTATGGAGTGTCTCCTACTAAATTAAAT
GATCTCTGCTTTACTAATGTCTATGCAGATTCATTTGTAATTAGAGGTGATGAAGTCAGACAAATCGCTCCAGGGCAAACTGGAAAGATTGCTGATT
ATAATTATAAATTACCAGATGATTTTACAGGCTGCGTTATAGCTTGGAATTCTAACAATCTTGATTCTAAGGTTGGTGGTAATTATAATTACCTGTA
TAGATTGTTTAGGAAGTCTAATCTCAAACCTTTTGAGAGAGATATTTCAACTGAAATCTATCAGGCCGGTAGCACACCTTGTAATGGTGTTGAAGGT
TTTAATTGTTACTTTCCTTTACAATCATATGGTTTCCAACCCACTAATGGTGTTGGTTACCAACCATACAGAGTAGTAGTACTTTCTTTTGAACTTC
TACATGCACCAGCAACTGTTTGTGGACCTAAAAAGTCTACTAATTTGGTTAAAAACAAATGTGTCAATTTCAACTTCAATGGTTTAACAGGCACAGG
TGTTCTTACTGAGTCTAACAAAAAGTTTCTGCCTTTCCAACAATTTGGCAGAGACATTGCTGACACTACTGATGCTGTCCGTGATCCACAGACACTT
GAGATTCTTGACATTACACCATGTTCTTTTGGTGGTGTCAGTGTTATAACACCAGGAACAAATACTTCTAACCAGGTTGCTGTTCTTTATCAGGATG
TTAACTGCACAGAAGTCCCTGTTGCTATTCATGCAGATCAACTTACTCCTACTTGGCGTGTTTATTCTACAGGTTCTAATGTTTTTCAAACACGTGC
AGGCTGTTTAATAGGGGCTGAACATGTCAACAACTCATATGAGTGTGACATACCCATTGGTGCAGGTATATGCGCTAGTTATCAGACTCAGACTAAT
TCTCCTCGGCGGGCACGTAGTGTAGCTAGTCAATCCATCATTGCCTACACTATGTCACTTGGTGCAGAAAATTCAGTTGCTTACTCTAATAACTCTA
TTGCCATACCCACAAATTTTACTATTAGTGTTACCACAGAAATTCTACCAGTGTCTATGACCAAGACATCAGTAGATTGTACAATGTACATTTGTGG
TGATTCAACTGAATGCAGCAATCTTTTGTTGCAATATGGCAGTTTTTGTACACAATTAAACCGTGCTTTAACTGGAATAGCTGTTGAACAAGACAAA
AACACCCAAGAAGTTTTTGCACAAGTCAAACAAATTTACAAAACACCACCAATTAAAGATTTTGGTGGTTTTAATTTTTCACAAATATTACCAGATC
CATCAAAACCAAGCAAGAGGTCATTTATTGAAGATCTACTTTTCAACAAAGTGACACTTGCAGATGCTGGCTTCATCAAACAATATGGTGATTGCCT
TGGTGATATTGCTGCTAGAGACCTCATTTGTGCACAAAAGTTTAACGGCCTTACTGTTTTGCCACCTTTGCTCACAGATGAAATGATTGCTCAATAC
ACTTCTGCACTGTTAGCGGGTACAATCACTTCTGGTTGGACCTTTGGTGCAGGTGCTGCATTACAAATACCATTTGCTATGCAAATGGCTTATAGGT
TTAATGGTATTGGAGTTACACAGAATGTTCTCTATGAGAACCAAAAATTGATTGCCAACCAATTTAATAGTGCTATTGGCAAAATTCAAGACTCACT
TTCTTCCACAGCAAGTGCACTTGGAAAACTTCAAGATGTGGTCAACCAAAATGCACAAGCTTTAAACACGCTTGTTAAACAACTTAGCTCCAATTTT
GGTGCAATTTCAAGTGTTTTAAATGATATCCTTTCACGTCTTGACAAAGTTGAGGCTGAAGTGCAAATTGATAGGTTGATCACAGGCAGACTTCAAA
GTTTGCAGACATATGTGACTCAACAATTAATTAGAGCTGCAGAAATCAGAGCTTCTGCTAATCTTGCTGCTACTAAAATGTCAGAGTGTGTACTTGG
ACAATCAAAAAGAGTTGATTTTTGTGGAAAGGGCTATCATCTTATGTCCTTCCCTCAGTCAGCACCTCATGGTGTAGTCTTCTTGCATGTGACTTAT
GTCCCTGCACAAGAAAAGAACTTCACAACTGCTCCTGCCATTTGTCATGATGGAAAAGCACACTTTCCTCGTGAAGGTGTCTTTGTTTCAAATGGCA
CACACTGGTTTGTAACACAAAGGAATTTTTATGAACCACAAATCATTACTACAGACAACACATTTGTGTCTGGTAACTGTGATGTTGTAATAGGAAT
TGTCAACAACACAGTTTATGATCCTTTGCAACCTGAATTAGACTCATTCAAGGAGGAGTTAGATAAATATTTTAAGAATCATACATCACCAGATGTT
GATTTAGGTGACATCTCTGGCATTAATGCTTCAGTTGTAAACATTCAAAAAGAAATTGACCGCCTCAATGAGGTTGCCAAGAATTTAAATGAATCTC
TCATCGATCTCCAAGAACTTGGAAAGTATGAGCAGTATATAAAATGGCCATGGTACATTTGGCTAGGTTTTATAGCTGGCTTGATTGCCATAGTAAT
GGTGACAATTATGCTTTGCTGTATGACCAGTTGCTGTAGTTGTCTCAAGGGCTGTTGTTCTTGTGGATCCTGCTGCAAATTTGATGAAGACGACTCT
GAGCCAGTGCTCAAAGGAGTCAAATTACATTACACATA

ATGTACTCATTCGTTTCGGAAGAGACAGGTACGTTAATAGTTAATAGCGTACTTCTTTTTCTTGCTTTCGTGGTATTCTTGCTAGTTACACTAGCCA
TCCTTACTGCGCTTCGATTGTGTGCGTACTGCTGCAATATTGTTAACGTGAGTCTTGTAAAACCTTCTTTTTACGTTTACTCTCGTGTTAAAAATCT
GAATTCTTCTAGAGTTCCTGATCTTCTGGTCTA
```

Assignment 03

a) Write a translation function, `translate` , that takes as an input a string representing a DNA coding sequence and returns a string representing the corresponding protein sequence (5 pts)

In [216]:
```python
# code for 3a
def translate(seq):
    protein = ""
    table = {
        'ATA':'I', 'ATC':'I', 'ATT':'I', 'ATG':'M',
        'ACA':'T', 'ACC':'T', 'ACG':'T', 'ACT':'T',
        'AAC':'N', 'AAT':'N', 'AAA':'K', 'AAG':'K',
        'AGC':'S', 'AGT':'S', 'AGA':'R', 'AGG':'R',
        'CTA':'L', 'CTC':'L', 'CTG':'L', 'CTT':'L',
        'CCA':'P', 'CCC':'P', 'CCG':'P', 'CCT':'P',
        'CAC':'H', 'CAT':'H', 'CAA':'Q', 'CAG':'Q',
        'CGA':'R', 'CGC':'R', 'CGG':'R', 'CGT':'R',
        'GTA':'V', 'GTC':'V', 'GTG':'V', 'GTT':'V',
        'GCA':'A', 'GCC':'A', 'GCG':'A', 'GCT':'A',
        'GAC':'D', 'GAT':'D', 'GAA':'E', 'GAG':'E',
        'GGA':'G', 'GGC':'G', 'GGG':'G', 'GGT':'G',
        'TCA':'S', 'TCC':'S', 'TCG':'S', 'TCT':'S',
        'TTC':'F', 'TTT':'F', 'TTA':'L', 'TTG':'L',
        'TAC':'Y', 'TAT':'Y', 'TAA':'_', 'TAG':'_',
        'TGC':'C', 'TGT':'C', 'TGA':'_', 'TGG':'W',
    }
    while len(seq)%3 != 0:
            seq = seq[:-1]
    if len(seq)%3 == 0:
        for i in range(0, len(seq), 3):
            codon = seq[i:i + 3]
            protein += table[codon]
    return protein
```

b) Test your `translate` function by applying it to the coding sequence of the "S" and "E" genes, and comparing your results to the protein sequences from the `covid-S-and-E.fsa` file provided above.

In [217]:
```python
print(translate(y[key][s_gene_start-1:s_gene_stop-1]))
print()
print(translate(y[key][e_gene_start-1:e_gene_stop-1]))
```

```
MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSSVLHSTQDLFLPFFSNVTWFHAIHVSGTNGTKRFDNPVLPFNDGVYFASTEK
SNIIRGWIFGTTLDSKTQSLLIVNNATNVVIKVCEFQFCNDPFLGVYYHKNNKSWMESEFRVYSSANNCTFEYVSQPFLMDLEGKQGNFKNLREFVF
KNIDGYFKIYSKHTPINLVRDLPQGFSALEPLVDLPIGINITRFQTLLALHRSYLTPGDSSSGWTAGAAAYYVGYLQPRTFLLKYNENGTITDAVDC
ALDPLSETKCTLKSFTVEKGIYQTSNFRVQPTESIVRFPNITNLCPFGEVFNATRFASVYAWNRKRISNCVADYSVLYNSASFSTFKCYGVSPTKLN
DLCFTNVYADSFVIRGDEVRQIAPGQTGKIADYNYKLPDDFTGCVIAWNSNNLDSKVGGNYNYLYRLFRKSNLKPFERDISTEIYQAGSTPCNGVEG
FNCYFPLQSYGFQPTNGVGYQPYRVVVLSFELLHAPATVCGPKKSTNLVKNKCVNFNFNGLTGTGVLTESNKKFLPFQQFGRDIADTTDAVRDPQTL
EILDITPCSFGGVSVITPGTNTSNQVAVLYQDVNCTEVPVAIHADQLTPTWRVYSTGSNVFQTRAGCLIGAEHVNNSYECDIPIGAGICASYQTQTN
SPRRARSVASQSIIAYTMSLGAENSVAYSNNSIAIPTNFTISVTTEILPVSMTKTSVDCTMYICGDSTECSNLLLQYGSFCTQLNRALTGIAVEQDK
NTQEVFAQVKQIYKTPPIKDFGGFNFSQILPDPSKPSKRSFIEDLLFNKVTLADAGFIKQYGDCLGDIAARDLICAQKFNGLTVLPPLLTDEMIAQY
TSALLAGTITSGWTFGAGAALQIPFAMQMAYRFNGIGVTQNVLYENQKLIANQFNSAIGKIQDSLSSTASALGKLQDVVNQNAQALNTLVKQLSSNF
GAISSVLNDILSRLDKVEAEVQIDRLITGRLQSLQTYVTQQLIRAAEIRASANLAATKMSECVLGQSKRVDFCGKGYHLMSFPQSAPHGVVFLHVTY
VPAQEKNFTTAPAICHDGKAHFPREGVFVSNGTHWFVTQRNFYEPQIITTDNTFVSGNCDVVIGIVNNTVYDPLQPELDSFKEELDKYFKNHTSPDV
DLGDISGINASVVNIQKEIDRLNEVAKNLNESLIDLQELGKYEQYIKWPWYIWLGFIAGLIAIVMVTIMLCCMTSCCSCLKGCCSCGSCCKFDEDDS
EPVLKGVKLHYT

MYSFVSEETGTLIVNSVLLFLAFVVFLLVTLAILTALRLCAYCCNIVNVSLVKPSFYVYSRVKNLNSSRVPDLLV
```

Assigment 04

The protein coding regions of many genes are encoded not as single continuous blocks of the genome, but instead in regions called "exons" that are separated by non-coding regions called "introns". Following transcription, intronic sequences are "spliced out" of messenger RNA (mRNA) by a protein complex called the "Sliceosome". The end product of this splicing process is the sequence that will actually be translated by ribosomes.

a) Write a splicing function that takes as input two arguments:

1. a string representing the genomic DNA sequence of the gene
2. a list of list (or tuples), where each sublist (tuple) contains a pair of numerical (integer) coordinates giving the start and stop coordinates (1-indexed, relative to the beginning of the sequence) of the exons of the gene.

The output should be a string representing to the spliced DNA sequence of the gene (i.e. the exons concatenated in the correct order) (5 pts)

In [220]:
```
# code for 4a
def splice(seq, exons):
    gene = ""
    exon_temp = ""
    for j,k in exons:
        exon_temp = seq[j - 1: k -1]
        gene = gene + exon_temp
    return gene
```

```
ATGC
```

b) test your splice function by looking up the exon information for the yeast gene ACT1 (https://www.yeastgenome.org/locus/S000001855) at the Saccharomyces Genome Database (https://yeastgenome.org) and using that information to splice the file `ACT1-genomic.fsa` (https://github.com/bio208fs-class/bio208fs-lecture/raw/master/data/ACT1-genomic.fsa) and then comparing your result to the spliced version of ACT1 in the file `ACT1-coding.fsa` (https://github.com/bio208fs-class/bio208fs-lecture/raw/master/data/ACT1-coding.fsa) (2 pts)

In [225]:
```
# code for 4b

#print(splice(,))

x = parse_FASTA("/Users/cleve/OneDrive/Documents/Notes/Notebooks Linked Materials/Duke University/
seq = x['ACT1']
exons = [(1,10),(320,1437)]
print(splice(seq, exons))
```

```
ATGGATTCTAGGTTGCTGCTTTGGTTATTGATAACGGTTCTGGTATGTGTAAAGCCGGTTTTGCCGGTGACGACGCTCCTCGTGCTGTCTTCCCATC
TATCGTCGGTAGACCAAGACACCAAGGTATCATGGTCGGTATGGGTCAAAAAGACTCCTACGTTGGTGATGAAGCTCAATCCAAGAGAGGTATCTTG
ACTTTACGTTACCCAATTGAACACGGTATTGTCACCAACTGGGACGATATGGAAAAGATCTGGCATCATACCTTCTACAACGAATTGAGAGTTGCCC
CAGAAGAACACCCTGTTCTTTTGACTGAAGCTCCAATGAACCCTAAATCAAACAGAGAAAAGATGACTCAAATTATGTTTGAAACTTTCAACGTTCC
AGCCTTCTACGTTTCCATCCAAGCCGTTTTGTCCTTGTACTCTTCCGGTAGAACTACTGGTATTGTTTTGGATTCCGGTGATGGTGTTACTCACGTC
GTTCCAATTTACGCTGGTTTCTCTCTACCTCACGCCATTTTGAGAATCGATTTGGCCGGTAGAGATTTGACTGACTACTTGATGAAGATCTTGAGTG
AACGTGGTTACTCTTTCTCCACCACTGCTGAAAGAGAAATTGTCCGTGACATCAAGGAAAAACTATGTTACGTCGCCTTGGACTTCGAACAAGAAAT
GCAAACCGCTGCTCAATCTTCTTCAATTGAAAAATCCTACGAACTTCCAGATGGTCAAGTCATCACTATTGGTAACGAAAGATTCAGAGCCCCAGAA
GCTTTGTTCCATCCTTCTGTTTTGGGTTTGGAATCTGCCGGTATTGACCAAACTACTTACAACTCCATCATGAAGTGTGATGTCGATGTCCGTAAGG
AATTATACGGTAACATCGTTATGTCCGGTGGTACCACCATGTTCCCAGGTATTGCCGAAAGAATGCAAAAGGAAATCACCGCTTTGGCTCCATCTTC
CATGAAGGTCAAGATCATTGCTCCTCCAGAAAGAAAGTACTCCGTCTGGATTGGTGGTTCTATCTTGGCTTCTTTGACTACCTTCCAACAAATGTGG
ATCTCAAAACAAGAATACGACGAAAGTGGTCCATCTATCGTTCACCACAAGTGTTTCTA
```

In [ ]:
```
```