.

Think of DNA as a 1D string of charactets A, C, G and T. acronym for (Adenine, cytosine, guanine, and thymine). 4 nucleotides used to construct DNA

Each 3 characters sequence of nucleotides makes 1 AMINO ACID

ATA - I

ATG - M

CAA - Q

TCT - S

TCG - W ¶

Refer to the dictionary called table below

the keys are codons or nucleotide triples.

And their values correspond to common one-letter symbols

used for the different amino acids.

In [2]:
```
# codon: 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',
}
```

Q1) use the dictionary above to look up the codon that corresponds to GTA

In [3]:
```
#Ans1
#Simply finds the value associated with key GTA
table['GTA']
```

Out[3]: 'V'

the read_seq method READS IN THE TWO text files and stores them as one big string.

In [4]:
```
seq = ''
def read_seq(inputfile):
    with open(inputfile,'r')as f:
        seq = f.read()# seq is one LONG STRING
        seq = seq.replace('\n','')
        seq = seq.replace('\r','')
        return seq



prt =  read_seq('protein.txt')
dna =  read_seq('dna.txt')
```

the translate function returns a string of AMINO ACIDS

Translate a string containing a nucleotide seq into a string containing the corrresponding sequence of amino acides. Nucleotides are translated in triplets using the table dictionary; each amino acide 4 is encoded with a string of length

The input to our program is going to be a DNA sequence We then read this sequence three letters at a time,translate each triplet to a single letter that stands for a specific amino acid, and then proceed to the next set of three letters.

complete the translate method below that takes DNA sequence and returns the protein sequence

In [18]:
```
# complete the translate method below that takes in a DNA sequence and
# returns the protein sequence
def translate(seq):
    dnaSeq=seq
    protein = ''
    #creates a counter so I can interate by threes while still keeping a
ccurate indexes
    x=0
    #iterate through DNA sequence
    for i in range(0, (len(dnaSeq)//3)):
        #store groups of three
        miniSeq=''
        #find the three letters and puts them into miniSeq
        for substring in range(0,3):
            miniSeq+=dnaSeq[x]
            x+=1
        #Finds the Codein using the dictionary and assigns it to protein
        protein+=table[miniSeq]
    return protein#returns protein sequence

translate(dna[20:32])
```

Out[18]:
```
'MSTH'
```

Q3) what is the output of translate(dna[20:938]) - display the output.

In [19]:
```
# Q3) Answer
translate(dna[20:938])
```

Out[19]:
```
'MSTHDTSLKTTEEVAFQIILLCQFGVGTFANVFLFVYNFSPISTGSKQRPRQVILRHMAVANALTLFLTI
FPNNMMTFAPIIPQTDLKCKLEFFTRLVARSTNLCSTCVLSIHQFVTLVPVNSGKGILRASVTNMASYSCY
SCWFFSVLNNIYIPIKVTGPQLTDNNNNSKSKLFCSTSDFSVGIVFLRFAHDATFMSIMVWTSVSMVLLLH
RHCQRMQYIFTLNQDPRGQAETTATHTILMLVVTFVGFYLLSLICIIFYTYFIYSHHSLRHCNDILVSGFP
TISPLLLTFRDPKGPCSVFFNC_'
```

Q4) What is the length of translate(dna[20:938])?

In [20]:
```
#Ans:
len(translate(dna[20:938]))
```

Out[20]:
```
306
```

Q5) How many neuclotides are there in translate(dna[20:938])

In [28]:
```
#Ans:
#Split each character in the amino sequence into a dictionary with Count
er
from collections import Counter
#x=amino sequence
x=translate(dna[20:938])
#y=dictionary of characters in sequence
y=Counter(x)
#prints the length of the full sequence minus the amount of '_' in the s
equence using the dictionary I made
print((len(x)-y['_']))
```

```
305
```

Q6) compare the length of the protein sequence - prt to length of the length of translate(dna[20:938]) ? what is the output?

In [71]:
```
#Ans:
prt=''
#opens the dna sequence and puts the whole sequence into prt
with open('dna.txt','r') as file:
    prt+=file.read().replace('\n', ' ')
#prints the difference between the length of dna and the length of the a
mino acids sequence
if len(prt)>len(translate(dna[20:938])):
    print('prt is longer than dna[20:938] by '+ str(len(prt)-len(transla
te(dna[20:938]))))
elif len(prt)>len(translate(dna[20:938])):
    print('prt is shorter than dna[20:938] by '+ str(len(translate(dna[2
0:938]))-len(prt)))
else:
    print('They are equal')
```

```
prt is longer than dna[20:938] by 867
```

Paper part of test below

In [38]:
```
import numpy as np
feature = np.arange(6,21)
```

In [41]:
```
x=np.random.rand()
```

In [10]:
```
x=np.random.uniform((-2,2))
x
```

Out[10]:
```
array([-0.52876867,  1.25640606])
```

In [3]:
```
x=np.random.randint(50,101,6)
```

Out[3]:
```
array([65, 97, 85, 66, 74, 53])
```

In [54]:
```
np.zeros(10)
```

Out[54]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [55]:
```
np.ones(10)
```

Out[55]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [4]:
```
np.full((1,10),5.0)
```

Out[4]:
```
array([[5., 5., 5., 5., 5., 5., 5., 5., 5., 5.]])
```

In [57]:
```
np.arange(10,51,2)
```

Out[57]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40,
42,
       44, 46, 48, 50])
```

In [58]:
```
np.arange(0,9).reshape(3,3)
```

Out[58]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [60]:
```
np.eye(3)
```

Out[60]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [61]:
```
np.random.randn(25)
```

Out[61]:
```
array([ 2.00639570e+00, -1.13135529e+00,  2.36512784e+00,  1.45850037e+
00,
       -1.01236814e+00,  2.48180744e+00, -1.28826025e+00,  1.14981570e+
00,
       -1.99805911e+00,  1.14021501e+00,  6.65431354e-01,  3.22071886e-
04,
       -4.32326155e-01, -1.97164640e+00, -1.72566128e+00,  8.26246158e-
03,
        8.01572355e-01, -5.88881404e-01,  2.08007968e-01, -2.51720413e-
01,
        6.98073794e-01,  1.61252582e+00,  1.25943960e+00,  9.14643238e-
01,
       -1.11381234e+00])
```

In [6]:
```
np.arange(0,1,20)
```

Out[6]:
```
array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

In [66]:
```
x=np.random.randn(100).std()
```

In [67]:
```
arr.argmin()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-67-78131e590238> in <module>
----> 1 arr.argmin()

NameError: name 'arr' is not defined
```

In [68]:
```
arr.dtype
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-68-99e7aa6f566b> in <module>
----> 1 arr.dtype

NameError: name 'arr' is not defined
```

In [2]:
```
import numpy as np
```

In [ ]:
```

```