```
# GeneLink test code.

dataDir = "../data"

# Aminio acid code letter to name
def a2n(a = 'F'):
   if a == 'F':
      return "phenylalanine"
   elif a == 'L':
      return "leucine"
   elif a == 'I':
      return "isoleucine"
   elif a == 'M':
      return "methionine"
   elif a == 'V':
      return "valine"
   elif a == 'S':
      return "serine"
   elif a == 'P':
      return "proline"
   elif a == 'T':
      return "threonine"
   elif a == 'A':
      return "alanine"
   elif a == 'Y':
      return "tyrosine"
   elif a == 'H':
      return "histidine"
   elif a == 'Q':
      return "glutamine"
   elif a == 'N':
      return "asparagine"
   elif a == 'K':
      return "lysine"
   elif a == 'D':
      return "aspartic"
   elif a == 'E':
      return "glutamic Acid"
   elif a == 'C':
      return "cysteine"
   elif a == 'W':
      return "tryptophan"
   elif a == 'R':
      return "arginine"
   elif a == 'S':
      return "serine"
   elif a == 'G':
      return "glycine"

# Transform a codon to an amino acid code letter.  Return '' for stop codons.
def c2a(c = 'TTT'):
    if c in ['TTT', 'TTC']:
        return 'F'
    elif c in ['TTA', 'TTG', 'CTT', 'CTC', 'CTA', 'CTG']:
        return 'L'
    elif c in ['ATT', 'ATC', 'ATA', 'ATG']:
        return 'I'
    elif c in ['GTT', 'GTC', 'GTA', 'GTG']:
        return 'V'
    elif c in ['TCT', 'TCC', 'TCA', 'TCG']:
        return 'S'
    elif c in ['CCT', 'CCC', 'CCA', 'CCG']:
        return 'P'
    elif c in ['ACT', 'ACC', 'ACA', 'ACG']:
        return 'T'
    elif c in ['GCT', 'GCC', 'GCA', 'GCG']:
        return 'A'
    elif c in ['TAT', 'TAC']:
        return 'Y'
    elif c in ['CAT', 'CAC']:
        return 'H'
    elif c in ['CAA', 'CAG']:
        return 'Q'
    elif c in ['AAT', 'AAC']:
        return 'N'
    elif c in ['AAA', 'AAG']:
        return 'K'
    elif c in['GAT', 'GAC']:
        return 'D'
    elif c in ['GAA', 'GAG']:
        return 'E'
    elif c in ['TGT', 'TGC']:
        return 'C'
    elif c in ['TGG']:
        return 'W'
    elif c in ['CGT', 'CGC', 'CGA', 'CGG', 'AGA', 'AGG']:
        return 'R'
    elif c in ['AGT', 'AGC']:
        return 'S'
    elif c in ['GGT', 'GGC', 'GGA', 'GGG']:
        return 'G'
    elif c in ['TAA', 'TAG']:     # Stop
        return ''
    elif c in ['TGA']:            # Stop
        return ''


# Transform a DNA sequence into a sequence of amino acids.
# Ignore base pairs that are not between start and stop codons.
def seq2c(seq, verbose = False):
    WAITING_FOR_START    = 0
    READING_AMINO_ACIDS = 1
    n = len(seq)
    if n < 9:
       return
    amino_acids = ''
    triple      = ''
    state       = WAITING_FOR_START
    start_index = 0
    i           = 0
    while i < len(seq):
       #print (i, state, triple)
       if state == WAITING_FOR_START:
          triple = seq[i:(i+3)]        # read three base pairs
          if triple == 'ATG':
             start_index = i
             i = i + 3
             state = READING_AMINO_ACIDS
          else:
             i = i + 1
       elif state == READING_AMINO_ACIDS:
          codon = seq[i:(i+3)]
          aa = c2a(codon)
          i = i + 3
          if len(aa) > 0:
             amino_acids = amino_acids + aa
          else:                        # read a stop
             if (verbose):
                print ("Start/Stop: ", start_index, i-3)
             state = WAITING_FOR_START
    return amino_acids
```

```python
# Read a DNA sequence from a FASTA formatted file.  The first
# line must be a comment line as it is ignored.
def readseq(filename):
    path = dataDir + "/" + filename
    seq = ""
    f = open(path, 'r')
    line = f.readline()
    for line in f:
        seq = seq + line.rstrip()
    f.close()
    return seq
```