```
'''
Below is the dictionary of codons-to-amino acids. When a codon is found in a DNA strand,
we traverse through the dictionary until we find the codon, and then return the amino acid
that accompanies it.
'''

AminoAcids = {"TTT": "Phe", "TTC": "Phe", "TTA": "Leu", "TTG": "Leu",
              "TCT": "Ser", "TCC": "Ser", "TCA": "Ser", "TCG": "Ser",
              "TAT": "Tyr", "TAC": "Tyr", "TAA": "END", "TAG": "END",
              "TGT": "Cys", "TGC": "Cys", "TGA": "END", "TGG": "Trp",
              "CTT": "Leu", "CTC": "Leu", "CTA": "Leu", "CTG": "Leu",
              "CCT": "Pro", "CCC": "Pro", "CCA": "Pro", "CCG": "Pro",
              "CAT": "His", "CAC": "His", "CAA": "Gin", "CAG": "Gin",
              "CGT": "Arg", "CGC": "Arg", "CGA": "Arg", "CGG": "Arg",
              "ATT": "Ile", "ATC": "Ile", "ATA": "Ile", "ATG": "Met",
              "ACT": "Thr", "ACC": "Thr", "ACA": "Thr", "ACG": "Thr",
              "AAT": "Asn", "AAC": "Asn", "AAA": "Lys", "AAG": "Lys",
              "AGT": "Ser", "AGC": "Ser", "AGA": "Arg", "AGG": "Arg",
              "GTT": "Val", "GTC": "Val", "GTA": "Val", "GTG": "Val",
              "GCT": "Ala", "GCC": "Ala", "GCA": "Ala", "GCG": "Ala",
              "GAT": "Asp", "GAC": "Asp", "GAA": "Glu", "GAG": "Glu",
              "GGT": "Gly", "GGC": "Gly", "GGA": "Gly", "GGG": "Gly"}

welcome = '''
Hello and welcome to the DNA-Amino Acid converter 9000. What this program does is take an
inputted
strand of DNA and prints the first chain of amino acids. You will be prompted to enter the
strand of DNA that you wish to convert. A correct input would be one that contains only A's, T's,
C's, and G's and
with no spaces in between (meaning "ACTG ATCG" is not a valid entry). It does not matter
whether the letters are
all lowercase or uppercase. If an invalid string is input, an error will be printed and the program
will not run.
'''

print(welcome)
dna = input("Please enter a valid DNA strand to find the first chain of amino acids.")

# Checks to make sure that the length of the DNA sequence contains at least 3 letters
length = len(dna)
if length < 3:
    print("Invalid input. DNA strand does not have 3 or more bases.")
    exit()
```

```
# Checks to make sure that only letters are in the input; no numbers, symbols, or spaces
if not dna.isalpha():
    print("Invalid input. DNA contains numbers or spaces, not just A's, T's, G's, or C's.")
    print("Please input a valid DNA strand as per the rules stated in the welcome line.")
    exit()

# Converts DNA sequence to all capital letters, to work with the dictionary
dna = dna.upper()

started = False             # This is a check for whether the first ATG base has been found
startofTriple = 0           # Index used as the beginning of the current triple being looked at
endofTriple = 3             # Index used as the end of the current triple being looked at
chain = []                  # Contains the amino acids found and is printed at the end

# Iterates through the DNA sequence, going through the process of grabbing 3 bases, getting
the corresponding amino acid
# from the AminoAcids library, and then appends it to the list which will be printed at the end.
while endofTriple <= length:
    triple = dna[startofTriple:endofTriple]
    if triple in AminoAcids:
        if AminoAcids[triple] == "Met":                     # This case is when the first ATG is
found, which appends "Met" to the amino acid list
            if not started:
                started = True
                chain.append("Met")
                startofTriple = startofTriple + 3
                endofTriple = endofTriple + 3
            else:
                chain.append("Met")
                startofTriple = startofTriple + 3
                endofTriple = endofTriple + 3
        elif AminoAcids[triple] == "END":                   # This case is when one of the END
codons are reached
            if started:
                started = False
                break
            elif endofTriple == length:                     # Special case when the last three bases
is am END codon but no ATG has been found to start the amino acid chain
                print("Invalid input. DNA sequence does not contain starting codon ATG")
                exit()
        elif started:                                       # This case is simply adding the codon's amino
acid to the output list after ATG has been found
```

```
            chain.append(AminoAcids[triple])
            startofTriple = startofTriple + 3
            endofTriple = endofTriple + 3
        else:                                   # This case is when a valid codon is read but
ATG has not been read yet
            if endofTriple == length:
                print("Invalid input. DNA sequence does not contain the starting codon ATG")
                exit()
            startofTriple = startofTriple + 1
            endofTriple = endofTriple + 1
    else:
        print("Invalid input. DNA contains other letters other than A T G and C.")
        print("Please input a valid DNA strand as per the rules stated in the welcome line.")
        exit()

print("The first amino acid chain of the given DNA sequence is: ")
print(chain)


'''
THESE ARE TEST CASES FOR THE PROGRAM. WHEN YOU TYPE IN THIS STRING OF
DNA, THE OUTPUT SHOULD MATCH WHAT IS TO
THE RIGHT OF THE ARROW.

ATG --> Met
ATGTTGTCG --> Met Leu Ser
CCAATGGAG --> Met Glu
ATGCGCCACTAA --> Met Arg His
ATGCGCCACTAACGC --> Met Arg His
ATGCGCCACTAAATG --> Met Arg His
ATGGCCACTAACGCCGC --> Met Arg His

DHAHDHAH --> Error
AT --> Error
ATGHWE --> Error
FE WEW --> Error
TAG --> Error
CIG --> Error
21121 --> Error
EE#DV --> Error
'''
```