Hands-On Twenty - Programs
Transcription, Translation, and
GC Content

```python
# 1) transcription_coding.py
# Author: Sami Khuri
# Last updated: January 14, 2016
# Purpose: To convert DNA coding strand (non-template strand)
#          to its RNA equivalent
# Program uses: for loop, if/else conditional construct,
#          string concatenation with "+" and escape sequence "\t"

def transcribe(string):
    """Converts a given DNA coding strand into its corresponding RNA sequence"""

    rna = ""
    for base in string:
        if base == "T":
            rna += "U"   # rna = rna + "U"
        else:
            rna += base  # rna = rna + base
    return rna
# end of transcribe()

coding_strand = "AACCTTGGGGTTCCAA"
rna = transcribe(coding_strand)

print "Coding Strand:", "\t", coding_strand
print "RNA:", "\t\t", rna

&&&&&&&&&&&&&&&&&&&&&&&
```

```python
# translation.py
# Author: Sami Khuri
# Last updated: January 14, 2016
# Purpose: To perform translation of a coding sequence (CDS).
#          A CDS starts with the start codon: ATG, and ends with a
#          stop codon: TAA, TAG, or TGA, and its length is a multiple of three.
# Program uses dictionary, slicing a list, and built-in python function range()

def translate(cds):
    """The input sequence is a DNA coding sequence (CDS)"""

    geneticode = {
    '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',
    }
    prot = ""
    for i in range(0,len(cds),3):
        codon = cds[i:i+3]   # slice is i,1+1,i+2
        prot = prot + geneticode[codon]
    return prot

cds = "ATGTATCCTACACCCATAATTGA"
print "CDS is", cds
print "Translation of CDS is", translate(cds)

&&&&&&&&&&&&&&&&&&&&&&&
```

```
# 3) draft_gc_content.py
# Purpose: A function that computes the percentage of G's and C's in a DNA sequence
#

def gc_count(seq):
    """Compute the percentage of G and C in a DNA sequence"""

    total_length = XXXXXXXXXX
    gc_count = XXXXXXXXXX
    # for each base pair in the string,
    for bp in seq:
    # next, if the bp is a G or a C,
        if XXXXXXXXXX:
        # increment the count of gc
            XXXXXXXXXX

    # divide the gc_count by the total_count
    XXXXXXXXXX = (XXXXXXXXXX(gc_count) / total_length)*XXXXXXXXXX
    return(gc_content)

dna = "GCAACCGGTTACGCAAACAG"
print dna
# Print the answer
print 'GC content:', gc_count(dna)

&&&&&&&&&&&&&&&&&&&&&&
```

```python
# 4) gc_content_NCBI.py
# Author: Sami Khuri
# Last updated: February 15, 2016
# Purpose: A function that computes the percentage of G's and C's in a DNA sequence
#          read from a file
# Program uses: input file, for loop, and
#               python built-in functions: lower() and float()

def gc_count(seq):
    """Compute the percentage of G and C in a DNA sequence"""
    g_count = 0
    a_count = 0
    c_count = 0
    t_count = 0

    for line in seq:
        # convert bases to lower case
        line = line.lower()
        # for each base pair in the string,
        for bp in line:
        # next, if the bp is a G,
            if bp == 'g':
                g_count = g_count + 1
            if bp == 'c':
                c_count = c_count + 1
            if bp == 'a':
                a_count = a_count + 1
            if bp == 't':
                t_count = t_count + 1

    # divide the gc_count by the total_count
    gc_content = (float(g_count + c_count) / float(a_count + c_count + g_count + t_count)) *100
    return(gc_content)

cds = open("BRCA1.txt", "r")
print 'GC content:', gc_count(cds), "%"
```