Goal of Program

This program is to do the following:

1. Extract the DNA sequence from a file and put into a string
2. Check for validity of the string (any character other than ACTG and its lowercase)
3. Compile the complementary strand
   - A-T & C-G base pair
   - Flip the strand from (3'-5') to (5'-3') or vice versa

1. Extract the DNA sequence

There are two files, 1 file has a valid DNA sequence and the other has invalid DNA sequence. Both are read into different strings.

```
with open('valid.txt', 'r') as file:
    valid_list = file.read().replace('\n', '') #if there is a newline,
ignore it.
with open('invalid.txt', 'r') as file:
    invalid_list = file.read().replace('\n', '')
```

2. Validity check

The validity check is created in a function (started out with def). To check the string, pass the string as the parameter to the function. The function will output whether the DNA string is valid or not.

`findall` is a function in Regular Expression (re) module, to find all matches in the mentioned regular expression inside `("[...]")` In this case, `"[^ACTGactg]"` expression means that we want to get any character that is not ACTG and actg. Then, we want to find all of these occurrences in the `str` string. The `findall` function will return a touple of string found, and we are assigning the touple to `match` variable.

```
import re
### Check for validity of string
def validity(str):
    match = re.findall("[^ACTGactg]", str) #find any character that
does not have ACTG or actg
    if match: #any touple detected
        print( str + "\tInvalid nucleotide detected")
    else:  #no touple detected
        print(str + "\t\tDNA sequence is good")
validity (valid_list)
validity (invalid_list)
```

```
ATGCTAGCTAGCTAG       DNA sequence is good
ATCGXATG*CTG#ATCGT    Invalid nucleotide detected
```

Compiling the complementary strand

Base pairing

In order to base pair, we need a loop that iterate at every character in a long list of string - `for c in str`. At every character, consider which nucleotide is it (A, T, C , or G). If the character is an Adenine (A), we want to attach a Thymine (T) to the `result` string.

Flipping strands

To flip the strand in reverse order, instead of using another function, we could append the complementary nucleotide at the beginning of the `result` string.

```
def compile_seq (str):
    result = "" #create an empty string to store the complementary
sequence
    for c in str:
        if c == 'A' or c == 'a':
            result = 'T' + result #adding the responding nucleotide at
the beginning of the result string
        if c == 'T' or c == 't':
            result = 'A' + result
        if c == 'C' or c == 'c':
            result = 'G' + result
        if c == 'G' or c== 'g':
            result = 'C' + result
    print('Given sequence: \t' + str + '\n' + 'Result sequence: \t' +
result)
compile_seq (valid_list) #calling the function to compile the
complementary sequence on the valid list

Given sequence: ATGCTAGCTAGCTAG
Result sequence:      CTAGCTAGCTAGCAT
```