Reading and Writing Files

In **Chapter 2** we learnt how to store our data in variables, and how to view the content of a variable using the `print ()` function.

However, in Biology we often need to read in large amounts of text that is already stored in a file. E.g. a vcf, bam, csv or a plain text file (like Notepad).

We also usually need to write the information we need into a new file and not the screen. The screen is also known as Standard Out (StdOut)

Python has some built-in functions and methods to help us accomplish this

Remember in the previous section, we had a DNA sequence and we created a variable to store it in.

```
my_dna = "ATTCCATCAAGCTGATCAGGTTATCCATCTAGATCATNNATAAAGTACTGGGCATGCAA"
```

Now let's pretend that the new DNA sequence we have to work with is so long, that someone cannot read us the DNA sequence, for fear of making a mistake.

So they stored the DNA sequence in a file called **dna.txt**

How do we access this DNA?

The open ( ) function

This function takes up to two arguments

```
open (file, mode)
```

The `file_name` takes just the **file name** if the input file and script are in the same directory

Or `open("path_of_directory/file", mode)` , if the script and input file are in separate locations

There are many different modes, but we will predominantly work with about four

```
1. open (file, "r")            : read mode (default)

2. open (file, "w")            : write mode

3. open (file, "a")            : append mode

4. open (file, "w+")           : read and write mode
```

In [1]:
```
# Please note that for this to work with using JUST open(),
# the file you want to read in,
# needs to be stored in the same folder as this script

open_dna_file = open("dna.txt", "r")
```

In [2]:
```
# you do NOT need to use the "r",
# because the default mode is the read mode

open_dna_file = open("dna.txt")
```

In [3]:
```
# What happens when you try to print it?

print (open_dna_file)

# When we used the open() method, we created a file object
# The print () function does not read a file object
```

```
<_io.TextIOWrapper name='dna.txt' mode='r' encoding='UTF-8'>
```

> Although it's by default on read mode, your computer has not actually read it yet

> It simply means that it's **ready to be read**

> Think of it as opening a plastic file from your office, but you never looked into the file to read it.

> Or like going to a doctor's office and the secretary just took out the existing files of all the patients who had appointments for that day

> So now let's also read the data in the opened file

The .read ( ) method

In [3]:
```
open_dna_file = open("dna.txt")

read_dna_in_file = open_dna_file.read()

# now we should be able to print

print(read_dna_in_file)
```

```
ATGGCAATAACCCCCCGTTTCTACTTCTAGAGGAGAAAAGTATTGACATGAGCGCTCCCGGCACAAGGGCCAAAGAAGTCTCCAATTTCTTATTTCCGAATGACATG
CGTCTCCTTGCGGGTAAATCACCGACCGCAATTCATAGAAGCCTGGGGGAACAGATAGGTCTAATTAGCTTAAGAGAGTAAATCCTGGGATCATTCAGTAGTAACCA
TAAACTTACGCTGGGGCTTCTTCGGCGGATTTTTACAGTTACCAACCAGGAGATTTGAAGTAAATCAGTTGAGGATTTAGCCGCGCTATCCGGTAATCTCCAAATTA
AAACATACCGTTCCATGAAGGCTAGAATTACTTACCGGCCTTTTCCATGCCTGCGCTATACCCCCCCACTCTCCCGCTTATCCGTCCGAGCGGAGGCAGTGCGATCC
TCCGTTAAGATATTCTTACGTGTGACGTAGCTATGTATTTTGCAGAGCTGGCGAACGCGTTGAACACTTCACAGATGGTAGGGATTCGGGTAAAGGGCGTATAATTG
GGGACTAACATAGGCGTAGACTACGATGGCGCCAACTCAATCGCAGCTCGAGCGCCCTGAATAACGTACTCATCTCAACTCATTCTCGGCAATCTACCGAGCGACTC
GATTATCAACGGCTGTCTAGCAGTTCTAATCTTTTGCCAGCATCGTAATAGCCTCCAAGAGATTGATGATAGCTATCGGCACAGAACTGAGACGGCGCCGATGGATA
GCGGACTTTCGGTCAACCACAATTCCCCACGGGACAGGTCCTGCGGTGCGCATCACTCTGAATGTACAAGCAACCCAAGTGGGCCGAGCCTGGACTCAGCTGGTTCC
TGCGTGAGCTCGAGACTCGGGATGACAGCTCTTTAAACATAGAGCGGGGGCGTCGAACGGTCGAGAAAGTCATAGTACCTCGGGTACCAACTTACTCAGGTTATTGC
TTGAAGCTGTACTATTTTAGGGGGGGAGCGCTGAAGGTCTCTTCTTCTCATGACTGAACTCGCGAGGGTCGTGAAGTCGGTTCCTTCAATGGTTAAAAAACAAAGGC
TTACTGTGCGCAGAGGAACGCCCATCTAGCGGCTGGCGTCTTGAATGCTCGGTCCCCTTTGTCATTCCGGATTAATCCATTTCCCTCATTCACGAGCTTGCGAAGTC
TACATTGGTATATGAATGCGACCTAGAAGAGGGCGCTTAAAATTGGCAGTGGTTGATGCTCTAAACTCCATTTGGTTTACTCGTGCATCACCGCGATAGGCTGACAA
AGGTTTAACATTGAATAGCAAGGCACTTCCGGTCTCAATGAACGGCCGGGAAAGGTACGCGCGCGGTATGGGAGGATCAAGGGGCCAATAGAGAGGCTCCTCTCTCA
CTCGCTAGGAGGCAAATGTAAAACAATGGTTACTGCATCGATACATAAAACATGTCCATCGGTTGCCCAAAGTGTTAAGTGTCTATCACCCCTAGGGCCGTTTCCCG
CATATAAACGCCAGGTTGTATCCGCATTTGATGCTACCGTGGATGAGTCTGCGTCGAGCGCGCCGCACGAATGTTGCAATGTATTGCATGAGTAGGGTTGACTAAGA
GCCGTTAGATGCGTCGCTGTACTAATAGTTGTCGACAGACCGTCGAGATTAGAAAATGGTACCAGCATTTTCGGAGGTTCTCTAACTAGTATGGATTGCGGTGTCTT
CACTGTGCTGCGGCTACCCATCGCCTGAAATCCAGCTGGTGTCAAGCCATCCCCTCTCCGGGACGCCGCATGTAGTGAAACATATACGTTGCACGGGTTCACCGCGG
TCCGTTCTGAGTCGACCAAGGACACAATCGAGCTCCGATCCGTACCCTCGACAAACTTGTACCCGACCCCCGGAGCTTGCCAGCTCCTCGGGTATCATGGAGCCTGT
GGTTCATCGCGTCCGATATCAAACTTCGTCATGATAAAGTCCCCCCCTCGGGAGTACCAGAGAAGATGACTACTGAGTTGTGCGAT
```

Also note that there are other methods for reading in a file

But this can be better demonstrated by using another dna file

So let's read in another dna file.....

But let's pretend that this file is not stored in the same folder as this script....

This file is called **other_dna.txt**

And it is stored on your **Desktop**, while this script is in your **Dropbox** folder

How will we read in this file?

```
In [4]: open_new_dna_file  = open("other_dna.txt")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-4-3a76f55873f2> in <module>
----> 1 open_new_dna_file  = open("other_dna.txt")

FileNotFoundError: [Errno 2] No such file or directory: 'other_dna.txt'
```

By looking at the error message, can you see why this didn't work?

> The script is looking for the file, in the same folder where this script is located.

> But this script and the input file, called "other_dna.txt", are in two different directories

> If you don't know in which directory you currently are,

> You can use the `pwd` (print work directory) command

```
In [5]: pwd ## NB!!!! Your OUTPUT will obviously be the path to where YOUR Jupyter notebook is running from
```

```
Out[5]: '/home/tracey/Dropbox/Python/Python_for_Biologists'
```

> So can you see that this notebook is running from my Dropbox folder

> But as I previously stated, the **other_dna.txt** file of interest is on my Desktop

> So now you have three choices

1. You can either bring the **other_dna.txt** file to the same folder as your script

2. Or you can change your directory...
So that this script runs from where your input file, **other_dna.txt**, is located
To do this, you use the `cd` command (change directory)

3. Or you can add the directory ahead of the file name, when you call the `open()` function.
E.g. if your input file is in your Desktop folder, then
`open ("/home/user/Desktop/input_file")`

In [6]:
```
# Let's try option no 2.
# for some reason the "cd" or "pwd" do not always work, but adding the percentage helps


## NB!!!! This should obviously be the path to YOUR DESKTOP
%cd /home/tracey/Desktop
open_new_dna_file  = open("other_dna.txt") # No error now
```

```
/home/tracey/Desktop
```

In [7]:
```
# But, let's change it back, because I actually want to remain in the Dropbox folder

%cd /home/tracey/Dropbox/Python/Python_for_Biologists ## NB!!! Your path will NOT be the same as mine
```

```
/home/tracey/Dropbox/Python/Python_for_Biologists
```

In [8]:
```
# Now let's try option no 3.
# Start by telling the script where the file is,
# by using the file path of your input file


## NB!!!! This should obviously be the path to YOUR DESKTOP
open_new_dna_file  = open("/home/tracey/Desktop/other_dna.txt")
# I personally prefer this option
```

In [9]:
```
# Now let's read it and see what is inside by printing it

## NB!!!! This should obviously be the path to YOUR DESKTOP

open_new_dna_file  = open("/home/tracey/Desktop/other_dna.txt")
read_new_dna_file = open_new_dna_file.read()
print (read_new_dna_file)

# Do you notice something different about this output?
```

```
ATGGCAATAACCCCCCGTTTCTACTTCTAGAGGAGAAAAGTATTGACAT
GAGCGCTCCCGGCACAAGGGCCAAAGAAGTCTCCAATTTCTTATTTCCG
AATGACATGCGTCTCCTTGCGGGTAAATCACCGACCGCAATTCATAGAA
GCCTGGGGGAACAGATAGGTCTAATTAGCTTAAGAGAGTAAATCCTGGG
ATCATTCAGTAGTAACCATAAACTTACGCTGGGGCTTCTTCGGCGGATT
TTTACAGTTACCAACCAGGAGATTTGAAGTAAATCAGTTGAGGATTTAG
CCGCGCTATCCGGTAATCTCCAAATTAAAACATACCGTTCCATGAAGGC
TAGAATTACTTACCGGCCTTTTCCATGCCTGCGCTATACCCCCCCACTC
TCCCGCTTATCCGTCCGAGCGGAGGCAGTGCGATCCTCCGTTAAGATAT
TCTTACGTGTGACGTAGCTATGTATTTTGCAGAGCTGGCGAACGCGTTG
AACACTTCACAGATGGTAGGGATTCGGGTAAAGGGCGTATAATTGGGGA
CTAACATAGGCGTAGACTACGATGGCGCCAACTCAATCGCAGCTCGAGC
GCCCTGAATAACGTACTCATCTCAACTCATTCTCGGCAATCTACCGAGC
GACTCGATTATCAACGGCTGTCTAGCAGTTCTAATCTTTTGCCAGCATC
GTAATAGCCTCCAAGAGATTGATGATAGCTATCGGCACAGAACTGAGAC
GGCGCCGATGGATAGCGGACTTTCGGTCAACCACAATTCCCCACGGGAC
AGGTCCTGCGGTGCGCATCACTCTGAATGTACAAGCAACCCAAGTGGGC
CGAGCCTGGACTCAGCTGGTTCCTGCGTGAGCTCGAGACTCGGGATGAC
AGCTCTTTAAACATAGAGCGGGGGCGTCGAACGGTCGAGAAAGTCATAG
TACCTCGGGTACCAACTTACTCAGGTTATTGCTTGAAGCTGTACTATTT
TAGGGGGGGAGCGCTGAAGGTCTCTTCTTCTCATGACTGAACTCGCGAG
GGTCGTGAAGTCGGTTCCTTCAATGGTTAAAAAACAAAGGCTTACTGTG
CGCAGAGGAACGCCCATCTAGCGGCTGGCGTCTTGAATGCTCGGTCCCC
TTTGTCATTCCGGATTAATCCATTTCCCTCATTCACGAGCTTGCGAAGT
CTACATTGGTATATGAATGCGACCTAGAAGAGGGCGCTTAAAATTGGCA
GTGGTTGATGCTCTAAACTCCATTTGGTTTACTCGTGCATCACCGCGAT
AGGCTGACAAAGGTTTAACATTGAATAGCAAGGCACTTCCGGTCTCAAT
GAACGGCCGGGAAAGGTACGCGCGCGGTATGGGAGGATCAAGGGGCCAA
TAGAGAGGCTCCTCTCTCACTCGCTAGGAGGCAAATGTAAAACAATGGT
TACTGCATCGATACATAAAACATGTCCATCGGTTGCCCAAAGTGTTAAG
```

```
TGTCTATCACCCCTAGGGCCGTTTCCCGCATATAAACGCCAGGTTGTAT
CCGCATTTGATGCTACCGTGGATGAGTCTGCGTCGAGCGCGCCGCACGA
ATGTTGCAATGTATTGCATGAGTAGGGTTGACTAAGAGCCGTTAGATGC
GTCGCTGTACTAATAGTTGTCGACAGACCGTCGAGATTAGAAAATGGTA
CCAGCATTTTCGGAGGTTCTCTAACTAGTATGGATTGCGGTGTCTTCAC
TGTGCTGCGGCTACCCATCGCCTGAAATCCAGCTGGTGTCAAGCCATCC
CCTCTCCGGGACGCCGCATGTAGTGAAACATATACGTTGCACGGGTTCA
CCGCGGTCCGTTCTGAGTCGACCAAGGACACAATCGAGCTCCGATCCGT
ACCCTCGACAAACTTGTACCCGACCCCCGGAGCTTGCCAGCTCCTCGGG
TATCATGGAGCCTGTGGTTCATCGCGTCCGATATCAAACTTCGTCATGA
TAAAGTCCCCCCCTCGGGAGTACCAGAGAAGATGACTACTGAGTTGTGC
GAT
```

This file seems to be narrower and split over multiple lines

But this is exactly the same DNA sequence as in the **dna.txt** file

So let's print the length to see if it's the same?

```
In [10]: len(read_dna_in_file)
Out[10]: 2013
```

```
In [11]: len(read_new_dna_file)
Out[11]: 2054
```

The new file is longer because the DNA sequence was split over multiple lines....

And in Python even new lines and blank spaces are counted as characters

Let's demonstrate this with a simple string variable

In [12]:
```
my_characters = "countme"
print(my_characters)
len(my_characters)
```

```
countme
```

Out[12]:
```
7
```

In [13]:
```
my_characters = "count me"
print(my_characters)
len(my_characters)
```

```
count me
```

Out[13]:
```
8
```

In [14]:
```
my_characters = "count me\n\n\n"
print(my_characters)
len(my_characters)
# there are three extra lines after the "count me"...
# and the length of this variable
```

```
count me



```

Out[14]:
```
11
```

Now that we are aware that there are spaces that we might not be able to see

You may want to remove these spaces

Because they may lead to incorrect inferences

The methods:

```
.strip ( )    (strip empty space from both sides)
.rstrip ( )   (strip empty space from the right side)
.lstrip ( )   (strip empty space from the left sides)
```

In [15]:
```
# For now, let's go back to our original "dna.txt" file

# I usually just use .strip().....
# because then I know it will take care of empty space on the right and left side
open_dna_file = open("dna.txt")

read_dna_in_file = open_dna_file.read()

len(read_dna_in_file)
```

Out[15]:
```
2013
```

In [16]:
```
open_dna_file = open("dna.txt")

read_dna_in_file = open_dna_file.read()

stripped_read_dna_in_file = read_dna_in_file.strip()

len(stripped_read_dna_in_file)


# leaving it empty strips it off any space
# whether its a blank space, a tab or a new line
# So it looks as if there was at least one space, possibly at the end of the file
```

Out[16]:
```
2012
```

> **NOTE:** You may also strip on other characters

In [17]:
```
my_sequence = "CCGCGCTATCCGGTAATCTCCAAATTAAAA"
strip_As_off_sequence = my_sequence.rstrip("AAAA")

print(my_sequence)
print(strip_As_off_sequence)
```

```
CCGCGCTATCCGGTAATCTCCAAATTAAAA
CCGCGCTATCCGGTAATCTCCAAATT
```

> ***NOTE!! Python has more than one read method

The methods:

`.read ( )`

`.readline( )`

`.readlines( )`

Let's understand the difference

In [18]:
```
# when we use ".read ()", the system reads this as one complete string entity


## NB!!!! This should obviously be the path to YOUR DESKTOP
open_new_dna_file  = open("/home/tracey/Desktop/other_dna.txt")
read_new_dna_file = open_new_dna_file.read()
print (read_new_dna_file)
```

```
ATGGCAATAACCCCCCGTTTCTACTTCTAGAGGAGAAAAGTATTGACAT
GAGCGCTCCCGGCACAAGGGCCAAAGAAGTCTCCAATTTCTTATTTCCG
AATGACATGCGTCTCCTTGCGGGTAAATCACCGACCGCAATTCATAGAA
GCCTGGGGGAACAGATAGGTCTAATTAGCTTAAGAGAGTAAATCCTGGG
ATCATTCAGTAGTAACCATAAACTTACGCTGGGGCTTCTTCGGCGGATT
TTTACAGTTACCAACCAGGAGATTTGAAGTAAATCAGTTGAGGATTTAG
CCGCGCTATCCGGTAATCTCCAAATTAAAACATACCGTTCCATGAAGGC
TAGAATTACTTACCGGCCTTTTCCATGCCTGCGCTATACCCCCCCACTC
TCCCGCTTATCCGTCCGAGCGGAGGCAGTGCGATCCTCCGTTAAGATAT
TCTTACGTGTGACGTAGCTATGTATTTTGCAGAGCTGGCGAACGCGTTG
AACACTTCACAGATGGTAGGGATTCGGGTAAAGGGCGTATAATTGGGGA
CTAACATAGGCGTAGACTACGATGGCGCCAACTCAATCGCAGCTCGAGC
GCCCTGAATAACGTACTCATCTCAACTCATTCTCGGCAATCTACCGAGC
GACTCGATTATCAACGGCTGTCTAGCAGTTCTAATCTTTTGCCAGCATC
GTAATAGCCTCCAAGAGATTGATGATAGCTATCGGCACAGAACTGAGAC
GGCGCCGATGGATAGCGGACTTTCGGTCAACCACAATTCCCCACGGGAC
AGGTCCTGCGGTGCGCATCACTCTGAATGTACAAGCAACCCAAGTGGGC
CGAGCCTGGACTCAGCTGGTTCCTGCGTGAGCTCGAGACTCGGGATGAC
AGCTCTTTAAACATAGAGCGGGGGCGTCGAACGGTCGAGAAAGTCATAG
TACCTCGGGTACCAACTTACTCAGGTTATTGCTTGAAGCTGTACTATTT
TAGGGGGGGAGCGCTGAAGGTCTCTTCTTCTCATGACTGAACTCGCGAG
GGTCGTGAAGTCGGTTCCTTCAATGGTTAAAAAACAAAGGCTTACTGTG
CGCAGAGGAACGCCCATCTAGCGGCTGGCGTCTTGAATGCTCGGTCCCC
TTTGTCATTCCGGATTAATCCATTTCCCTCATTCACGAGCTTGCGAAGT
CTACATTGGTATATGAATGCGACCTAGAAGAGGGCGCTTAAAATTGGCA
GTGGTTGATGCTCTAAACTCCATTTGGTTTACTCGTGCATCACCGCGAT
AGGCTGACAAAGGTTTAACATTGAATAGCAAGGCACTTCCGGTCTCAAT
GAACGGCCGGGAAAGGTACGCGCGCGGTATGGGAGGATCAAGGGGCCAA
TAGAGAGGCTCCTCTCTCACTCGCTAGGAGGCAAATGTAAAACAATGGT
TACTGCATCGATACATAAAACATGTCCATCGGTTGCCCAAAGTGTTAAG
TGTCTATCACCCCTAGGGCCGTTTCCCGCATATAAACGCCAGGTTGTAT
CCGCATTTGATGCTACCGTGGATGAGTCTGCGTCGAGCGCGCCGCACGA
```

```
ATGTTGCAATGTATTGCATGAGTAGGGTTGACTAAGAGCCGTTAGATGC
GTCGCTGTACTAATAGTTGTCGACAGACCGTCGAGATTAGAAAATGGTA
CCAGCATTTTCGGAGGTTCTCTAACTAGTATGGATTGCGGTGTCTTCAC
TGTGCTGCGGCTACCCATCGCCTGAAATCCAGCTGGTGTCAAGCCATCC
CCTCTCCGGGACGCCGCATGTAGTGAAACATATACGTTGCACGGGTTCA
CCGCGGTCCGTTCTGAGTCGACCAAGGACACAATCGAGCTCCGATCCGT
ACCCTCGACAAACTTGTACCCGACCCCCGGAGCTTGCCAGCTCCTCGGG
TATCATGGAGCCTGTGGTTCATCGCGTCCGATATCAAACTTCGTCATGA
TAAAGTCCCCCCCTCGGGAGTACCAGAGAAGATGACTACTGAGTTGTGC
GAT
```

In [19]:

```
# When we use ".readline ( )"
# It reads only the fist line


## NB!!!! This should obviously be the path to YOUR DESKTOP
open_new_dna_file  = open("/home/tracey/Desktop/other_dna.txt")
readline_new_dna_file = open_new_dna_file.readline()
print (readline_new_dna_file)
```

```
ATGGCAATAACCCCCCGTTTCTACTTCTAGAGGAGAAAAGTATTGACAT
```

In [21]:
```python
# The ".readlines ( )"
# Splits the file after every new line "\n", and makes a list out of the file
# where every new line is an item in the list
# We will learn more about lists in the upcoming chapters


## NB!!!! This should obviously be the path to YOUR DESKTOP
open_new_dna_file  = open("/home/tracey/Desktop/other_dna.txt")
readlines_new_dna_file = open_new_dna_file.readlines()
print (readlines_new_dna_file)
```

```
['ATGGCAATAACCCCCCGTTTCTACTTCTAGAGGAGAAAAGTATTGACAT\n', 'GAGCGCTCCCGGCACAAGGGCCAAAGAAGTCTCCAATTTCTTATTTCCG
\n', 'AATGACATGCGTCTCCTTGCGGGTAAATCACCGACCGCAATTCATAGAA\n', 'GCCTGGGGGAACAGATAGGTCTAATTAGCTTAAGAGAGTAAATCCT
GGG\n', 'ATCATTCAGTAGTAACCATAAACTTACGCTGGGGCTTCTTCGGCGGATT\n', 'TTTACAGTTACCAACCAGGAGATTTGAAGTAAATCAGTTGAGG
ATTTAG\n', 'CCGCGCTATCCGGTAATCTCCAAATTAAAACATACCGTTCCATGAAGGC\n', 'TAGAATTACTTACCGGCCTTTTCCATGCCTGCGCTATACC
CCCCCACTC\n', 'TCCCGCTTATCCGTCCGAGCGGAGGCAGTGCGATCCTCCGTTAAGATAT\n', 'TCTTACGTGTGACGTAGCTATGTATTTTGCAGAGCTG
GCGAACGCGTTG\n', 'AACACTTCACAGATGGTAGGGATTCGGGTAAAGGGCGTATAATTGGGGA\n', 'CTAACATAGGCGTAGACTACGATGGCGCCAACTC
AATCGCAGCTCGAGC\n', 'GCCCTGAATAACGTACTCATCTCAACTCATTCTCGGCAATCTACCGAGC\n', 'GACTCGATTATCAACGGCTGTCTAGCAGTTC
TAATCTTTTGCCAGCATC\n', 'GTAATAGCCTCCAAGAGATTGATGATAGCTATCGGCACAGAACTGAGAC\n', 'GGCGCCGATGGATAGCGGACTTTCGGTC
AACCACAATTCCCCACGGGAC\n', 'AGGTCCTGCGGTGCGCATCACTCTGAATGTACAAGCAACCCAAGTGGGC\n', 'CGAGCCTGGACTCAGCTGGTTCCTG
CGTGAGCTCGAGACTCGGGATGAC\n', 'AGCTCTTTAAACATAGAGCGGGGGCGTCGAACGGTCGAGAAAGTCATAG\n', 'TACCTCGGGTACCAACTTACTC
AGGTTATTGCTTGAAGCTGTACTATTT\n', 'TAGGGGGGGAGCGCTGAAGGTCTCTTCTTCTCATGACTGAACTCGCGAG\n', 'GGTCGTGAAGTCGGTTCCT
TCAATGGTTAAAAAACAAAGGCTTACTGTG\n', 'CGCAGAGGAACGCCCATCTAGCGGCTGGCGTCTTGAATGCTCGGTCCCC\n', 'TTTGTCATTCCGGATT
AATCCATTTCCCTCATTCACGAGCTTGCGAAGT\n', 'CTACATTGGTATATGAATGCGACCTAGAAGAGGGCGCTTAAAATTGGCA\n', 'GTGGTTGATGCTC
TAAACTCCATTTGGTTTACTCGTGCATCACCGCGAT\n', 'AGGCTGACAAAGGTTTAACATTGAATAGCAAGGCACTTCCGGTCTCAAT\n', 'GAACGGCCGG
GAAAGGTACGCGCGCGGTATGGGAGGATCAAGGGGCCAA\n', 'TAGAGAGGCTCCTCTCTCACTCGCTAGGAGGCAAATGTAAAACAATGGT\n', 'TACTGCA
TCGATACATAAAACATGTCCATCGGTTGCCCAAAGTGTTAAG\n', 'TGTCTATCACCCCTAGGGCCGTTTCCCGCATATAAACGCCAGGTTGTAT\n', 'CCGC
ATTTGATGCTACCGTGGATGAGTCTGCGTCGAGCGCGCCGCACGA\n', 'ATGTTGCAATGTATTGCATGAGTAGGGTTGACTAAGAGCCGTTAGATGC\n', 'G
TCGCTGTACTAATAGTTGTCGACAGACCGTCGAGATTAGAAAATGGTA\n', 'CCAGCATTTTCGGAGGTTCTCTAACTAGTATGGATTGCGGTGTCTTCAC\n',
'TGTGCTGCGGCTACCCATCGCCTGAAATCCAGCTGGTGTCAAGCCATCC\n', 'CCTCTCCGGGACGCCGCATGTAGTGAAACATATACGTTGCACGGGTTCA
\n', 'CCGCGGTCCGTTCTGAGTCGACCAAGGACACAATCGAGCTCCGATCCGT\n', 'ACCCTCGACAAACTTGTACCCGACCCCCGGAGCTTGCCAGCTCCTC
GGG\n', 'TATCATGGAGCCTGTGGTTCATCGCGTCCGATATCAAACTTCGTCATGA\n', 'TAAAGTCCCCCCCTCGGGAGTACCAGAGAAGATGACTACTGAG
TTGTGC\n', 'GAT\n']
```

List and strings can both be sliced using `[ : ]`

So, by using `.readlines( )` , you can for example skip reading the first 2 lines/elements

using the following

```
In [22]: ## NB!!!! This should obviously be the path to YOUR DESKTOP
         open_new_dna_file  = open("/home/tracey/Desktop/other_dna.txt")
         readlines_new_dna_file = open_new_dna_file.readlines()
         skip_2_lines = readlines_new_dna_file[2:] # you skipped elements 0 and element 1
         print(skip_2_lines)
```

```
['AATGACATGCGTCTCCTTGCGGGTAAATCACCGACCGCAATTCATAGAA\n', 'GCCTGGGGGAACAGATAGGTCTAATTAGCTTAAGAGAGTAAATCCTGGG
\n', 'ATCATTCAGTAGTAACCATAAACTTACGCTGGGGCTTCTTCGGCGGATT\n', 'TTTACAGTTACCAACCAGGAGATTTGAAGTAAATCAGTTGAGGATT
TAG\n', 'CCGCGCTATCCGGTAATCTCCAAATTAAAACATACCGTTCCATGAAGGC\n', 'TAGAATTACTTACCGGCCTTTTCCATGCCTGCGCTATACCCCC
CCACTC\n', 'TCCCGCTTATCCGTCCGAGCGGAGGCAGTGCGATCCTCCGTTAAGATAT\n', 'TCTTACGTGTGACGTAGCTATGTATTTTGCAGAGCTGGCG
AACGCGTTG\n', 'AACACTTCACAGATGGTAGGGATTCGGGTAAAGGGCGTATAATTGGGGA\n', 'CTAACATAGGCGTAGACTACGATGGCGCCAACTCAAT
CGCAGCTCGAGC\n', 'GCCCTGAATAACGTACTCATCTCAACTCATTCTCGGCAATCTACCGAGC\n', 'GACTCGATTATCAACGGCTGTCTAGCAGTTCTAA
TCTTTTGCCAGCATC\n', 'GTAATAGCCTCCAAGAGATTGATGATAGCTATCGGCACAGAACTGAGAC\n', 'GGCGCCGATGGATAGCGGACTTTCGGTCAAC
CACAATTCCCCACGGGAC\n', 'AGGTCCTGCGGTGCGCATCACTCTGAATGTACAAGCAACCCAAGTGGGC\n', 'CGAGCCTGGACTCAGCTGGTTCCTGCGT
GAGCTCGAGACTCGGGATGAC\n', 'AGCTCTTTAAACATAGAGCGGGGGCGTCGAACGGTCGAGAAAGTCATAG\n', 'TACCTCGGGTACCAACTTACTCAGG
TTATTGCTTGAAGCTGTACTATTT\n', 'TAGGGGGGGAGCGCTGAAGGTCTCTTCTTCTCATGACTGAACTCGCGAG\n', 'GGTCGTGAAGTCGGTTCCTTCA
ATGGTTAAAAAACAAAGGCTTACTGTG\n', 'CGCAGAGGAACGCCCATCTAGCGGCTGGCGTCTTGAATGCTCGGTCCCC\n', 'TTTGTCATTCCGGATTAAT
CCATTTCCCTCATTCACGAGCTTGCGAAGT\n', 'CTACATTGGTATATGAATGCGACCTAGAAGAGGGCGCTTAAAATTGGCA\n', 'GTGGTTGATGCTCTAA
ACTCCATTTGGTTTACTCGTGCATCACCGCGAT\n', 'AGGCTGACAAAGGTTTAACATTGAATAGCAAGGCACTTCCGGTCTCAAT\n', 'GAACGGCCGGGAA
AGGTACGCGCGCGGTATGGGAGGATCAAGGGGCCAA\n', 'TAGAGAGGCTCCTCTCTCACTCGCTAGGAGGCAAATGTAAAACAATGGT\n', 'TACTGCATCG
ATACATAAAACATGTCCATCGGTTGCCCAAAGTGTTAAG\n', 'TGTCTATCACCCCTAGGGCCGTTTCCCGCATATAAACGCCAGGTTGTAT\n', 'CCGCATT
TGATGCTACCGTGGATGAGTCTGCGTCGAGCGCGCCGCACGA\n', 'ATGTTGCAATGTATTGCATGAGTAGGGTTGACTAAGAGCCGTTAGATGC\n', 'GTCG
CTGTACTAATAGTTGTCGACAGACCGTCGAGATTAGAAAATGGTA\n', 'CCAGCATTTTCGGAGGTTCTCTAACTAGTATGGATTGCGGTGTCTTCAC\n', 'T
GTGCTGCGGCTACCCATCGCCTGAAATCCAGCTGGTGTCAAGCCATCC\n', 'CCTCTCCGGGACGCCGCATGTAGTGAAACATATACGTTGCACGGGTTCA\n',
'CCGCGGTCCGTTCTGAGTCGACCAAGGACACAATCGAGCTCCGATCCGT\n', 'ACCCTCGACAAACTTGTACCCGACCCCCGGAGCTTGCCAGCTCCTCGGG
\n', 'TATCATGGAGCCTGTGGTTCATCGCGTCCGATATCAAACTTCGTCATGA\n', 'TAAAGTCCCCCCCTCGGGAGTACCAGAGAAGATGACTACTGAGTTG
TGC\n', 'GAT\n']
```

In [23]:
```
# or you can read
## NB!!!! This should obviously be the path to YOUR DESKTOP
open_new_dna_file  = open("/home/tracey/Desktop/other_dna.txt")
readlines_new_dna_file = open_new_dna_file.readline()
skip_2_lines = readlines_new_dna_file[2:] # you skipped elements 0 and element 1
print(skip_2_lines)
```

```
GGCAATAACCCCCCGTTTCTACTTCTAGAGGAGAAAAGTATTGACAT
```

One thing that we haven't covered thus far

Is that you should always close your file

after you've read in what you needed using the **method**:

```
input_file.close()
```

In [24]:
```
## NB!!!! This should obviously be the path to YOUR DESKTOP

open_new_dna_file  = open("/home/tracey/Desktop/other_dna.txt")
readlines_new_dna_file = open_new_dna_file.readline()
skip_2_lines = readlines_new_dna_file[2:] # you skipped elements 0 and element 1
print(skip_2_lines)
open_new_dna_file.close()
```

```
GGCAATAACCCCCCGTTTCTACTTCTAGAGGAGAAAAGTATTGACAT
```

Here it may not make as much sense as to why you may want to read in your lines as a list with each line being an item, but there are occasions where may want this

I usually use the `.read ( )`

And use a `for loop` to access the lines in the file that I wish to work with

But we haven't done loops yet

So we will see that later

Note There are other libraries that can be imported that helps us to better read in manipulate certain file types and the data in the file

Libraries can be imported e..g:

`import csv`

`import pandas`

`from Bio import SeqIO` .....(This is from the BioPython collection of Python tools)

Writing to files

In order to write content to a file, you need to take care of five things:

1. **Create/name** an output file
2. Instruct it to **open the file** for **writing**
3. Instruct it when it needs to start writing
4. Instruct it on **what** it should write
5. Close the file

Remember, you've only opened files for reading, thus far

These instructions actually pertain to reading files too. Even though you may not be aware that you followed all these steps

There are two ways to write:

And it will all depend on what you chose when you opened the file

1. Start writing your output from the top of the file

Even if it means you overwrite everything that was already in the file

```
open(output_file, "w")
```

2. Start writing on a new line, AFTER the last character in the file (append)

```
open(output_file, "a")
```

Let's open an output file and write

This is the end of the file, into the output file, using the append mode

But let's first see if there is already data in the output file

In [25]:
```
## NB!!!! This should obviously be the path to YOUR DESKTOP

outfile = open("/home/tracey/Desktop/outfile.txt", "a")
outfile.write("This is the end of the file")
outfile.close()


# note that we have to close the file
# Or you will notice, that you probably need to run this code twice
# Before you are able to see the line in the file
# Closing files after reading and writing, is GOOD PRACTICE
```

In [26]:
```
# Notice the difference between the append and the write mode

## NB!!!! This should obviously be the path to YOUR DESKTOP
outfile = open("/home/tracey/Desktop/outfile.txt", "w")
outfile.write("I will overwrite what is already in your file")
outfile.close()
```

In [27]:
```
# The previous output file, called outfile.txt", already existed
# But if your output file does not exist,
# you can create it, in the same way


## NB!!!! This should obviously be the path to YOUR DESKTOP
outfile = open("/home/tracey/Desktop/new_outfile.txt", "w")

outfile.write("This is the a new output file")

outfile.close()
```

Best Practice

It is very common to forget to close a file after we've opened it

By adding a **`with`** in front of the function

We **automatically close** the file

Instead of `open (file, 'mode')` , use

```
with open (file, 'mode')
```

In [28]:
```
# We can rewrite:

outfile = open("new_outfile.txt", "w")
outfile.write("This is the a new output file")
outfile.close()

# as

with open("new_outfile.txt", "w") as outfile:
    outfile.write("This is the a new output file")

# we don't need the line "outfile.close()"
# the "with" part, takes care of that

# Let's find the file called "new_outfile.txt" on my Desktop
```

What went wrong above?

Why is this file not on my Desktop?

In [29]:
```
# If I want it on my Desktop
# I need to specify this

# Or it will create an output file in the directory where this script is

## NB!!!! This should obviously be the path to YOUR DESKTOP
with open("/home/tracey/Desktop/new_outfile.txt", "w") as outfile:
    outfile.write("This is the a new output file")
```

In [32]:
```
# "with open(file, mode)" should also be used when opening a file for reading

with open("dna.txt") as open_dna_file:
    read_dna_in_file = open_dna_file.read().strip()

print(read_dna_in_file)
```

```
ATGGCAATAACCCCCCGTTTCTACTTCTAGAGGAGAAAAGTATTGACATGAGCGCTCCCGGCACAAGGGCCAAAGAAGTCTCCAATTTCTTATTTCCGAATGACATG
CGTCTCCTTGCGGGTAAATCACCGACCGCAATTCATAGAAGCCTGGGGGAACAGATAGGTCTAATTAGCTTAAGAGAGTAAATCCTGGGATCATTCAGTAGTAACCA
TAAACTTACGCTGGGGCTTCTTCGGCGGATTTTTACAGTTACCAACCAGGAGATTTGAAGTAAATCAGTTGAGGATTTAGCCGCGCTATCCGGTAATCTCCAAATTA
AAACATACCGTTCCATGAAGGCTAGAATTACTTACCGGCCTTTTCCATGCCTGCGCTATACCCCCCCACTCTCCCGCTTATCCGTCCGAGCGGAGGCAGTGCGATCC
TCCGTTAAGATATTCTTACGTGTGACGTAGCTATGTATTTTGCAGAGCTGGCGAACGCGTTGAACACTTCACAGATGGTAGGGATTCGGGTAAAGGGCGTATAATTG
GGGACTAACATAGGCGTAGACTACGATGGCGCCAACTCAATCGCAGCTCGAGCGCCCTGAATAACGTACTCATCTCAACTCATTCTCGGCAATCTACCGAGCGACTC
GATTATCAACGGCTGTCTAGCAGTTCTAATCTTTTGCCAGCATCGTAATAGCCTCCAAGAGATTGATGATAGCTATCGGCACAGAACTGAGACGGCGCCGATGGATA
GCGGACTTTCGGTCAACCACAATTCCCCACGGGACAGGTCCTGCGGTGCGCATCACTCTGAATGTACAAGCAACCCAAGTGGGCCGAGCCTGGACTCAGCTGGTTCC
TGCGTGAGCTCGAGACTCGGGATGACAGCTCTTTAAACATAGAGCGGGGGCGTCGAACGGTCGAGAAAGTCATAGTACCTCGGGTACCAACTTACTCAGGTTATTGC
TTGAAGCTGTACTATTTTAGGGGGGGAGCGCTGAAGGTCTCTTCTTCTCATGACTGAACTCGCGAGGGTCGTGAAGTCGGTTCCTTCAATGGTTAAAAAACAAAGGC
TTACTGTGCGCAGAGGAACGCCCATCTAGCGGCTGGCGTCTTGAATGCTCGGTCCCCTTTGTCATTCCGGATTAATCCATTTCCCTCATTCACGAGCTTGCGAAGTC
TACATTGGTATATGAATGCGACCTAGAAGAGGGCGCTTAAAATTGGCAGTGGTTGATGCTCTAAACTCCATTTGGTTTACTCGTGCATCACCGCGATAGGCTGACAA
AGGTTTAACATTGAATAGCAAGGCACTTCCGGTCTCAATGAACGGCCGGGAAAGGTACGCGCGCGGTATGGGAGGATCAAGGGGCCAATAGAGAGGCTCCTCTCTCA
CTCGCTAGGAGGCAAATGTAAAACAATGGTTACTGCATCGATACATAAAACATGTCCATCGGTTGCCCAAAGTGTTAAGTGTCTATCACCCCTAGGGCCGTTTCCCG
CATATAAACGCCAGGTTGTATCCGCATTTGATGCTACCGTGGATGAGTCTGCGTCGAGCGCGCCGCACGAATGTTGCAATGTATTGCATGAGTAGGGTTGACTAAGA
GCCGTTAGATGCGTCGCTGTACTAATAGTTGTCGACAGACCGTCGAGATTAGAAAATGGTACCAGCATTTTCGGAGGTTCTCTAACTAGTATGGATTGCGGTGTCTT
CACTGTGCTGCGGCTACCCATCGCCTGAAATCCAGCTGGTGTCAAGCCATCCCCTCTCCGGGACGCCGCATGTAGTGAAACATATACGTTGCACGGGTTCACCGCGG
TCCGTTCTGAGTCGACCAAGGACACAATCGAGCTCCGATCCGTACCCTCGACAAACTTGTACCCGACCCCCGGAGCTTGCCAGCTCCTCGGGTATCATGGAGCCTGT
GGTTCATCGCGTCCGATATCAAACTTCGTCATGATAAAGTCCCCCCCTCGGGAGTACCAGAGAAGATGACTACTGAGTTGTGCGAT
```

In [37]:
```
# otherwise you should close it yourself

open_dna_file = open("dna.txt")

read_dna_in_file = open_dna_file.read()

open_dna_file.close()
```

EXERCISES

1. Splitting genomic DNA

a) Look in the chapter_3 folder, for a file called "genomic_dna.txt".

It contains the same piece of genomic DNA that we were using in the final exercise from chapter 2

b) Write a program that will split the genomic DNA into coding and non-coding parts, and write these sequences to two separate files.

Hint: use your solution to the last exercise from chapter 2 as a starting point.

In [38]:
```
###############
# Solution 1a #
###############


# NB!! Remember that YOUR "genomic_dna.txt" file will be in a different directory.
# So use the path where your file is
## NB!!!! This should obviously be the path to YOUR DESKTOP

# Locate all and create the necessary files
in_file =  open ("/home/tracey/Desktop/genomic_dna.txt")
output_coding = open("/home/tracey/Desktop/coding_file.txt", "w")
output_non_coding = open ("/home/tracey/Desktop/non_coding_file.txt", "w")

# read the input file
genomic_dna = in_file.read()


# get the introns and exons from my file
first_exon = genomic_dna[:62]
last_exon = genomic_dna[90:]
combined_exons = first_exon + last_exon
intron = (genomic_dna[62:90]).lower()

# write my output to their respective files
output_coding.write(combined_exons)
output_non_coding.write(intron)

# close the files
in_file.close()
output_coding.close()
output_non_coding.close()
```

The better solution would be this one

In [39]:
```
###############
# Solution 1b #
###############



# NB!! Remember that your genomic file will be in a different directory.
# So use the path where your file is


# Locate all and create the necessary files
## NB!!!! This should obviously be the path to YOUR DESKTOP
with open ("/home/tracey/Desktop/genomic_dna.txt") as in_file, \
    open ("/home/tracey/Desktop/coding_file.txt", "w") as output_coding, \
    open ("/home/tracey/Desktop/non_coding_file.txt", "w") as output_non_coding:


    # read the input file
    genomic_dna = in_file.read().strip()


    # get the introns and exons from my file
    first_exon = genomic_dna[:62]
    last_exon = genomic_dna[90:]
    combined_exons = first_exon + last_exon
    intron = (genomic_dna[62:90]).lower()

    # write my output to their respective files
    output_coding.write(combined_exons)
    output_non_coding.write(intron)


    # Note that you have to stick to the indentation
    # I opened all the files I needed at once, by separating them with commas
    # I used the escape character "\", so that I don't have to write it all in one line
    # Makes the code easier to read
    # Please note, that you will NOT always open a file for writing, at the beginning.
```

2. Writing a FASTA file

FASTA file format is a commonly-used DNA and protein sequence file format.

A single sequence in FASTA format looks like this:

>sequence_name

ATCGACTGATCGATCGTACGAT

Where sequence_name is a header that describes the sequence (the greater-than symbol indicates the start of the header line).

Often, the header contains an accession number that relates to the record for the sequence in a public sequence database.

A single FASTA file can contain multiple sequences, like this:

>sequence_one

ATCGATCGATCGATCGAT

>sequence_two

ACTAGCTAGCTAGCATCG

sequence_three

ACTGCATCGATCGTACCT

Write a program that will create a FASTA file for the following three sequences***

make sure that all sequences are in upper case and only contain the bases A, T, G and C

Sequence header	DNA sequence
ABC123	ATCGTACGATCGATCGATCGCTAGACGTATCG
DEF456	actgatcgacgatcgatcgatcacgact

Sequence header	DNA sequence
HIJ789	ACTGAC-ACTGT--ACTGTA----CATGTG

In [40]:
```
###############
# Solution 2 #
###############


# store the sequences in an appropriate variable name

ABC123 = "ATCGTACGATCGATCGATCGCTAGACGTATCG"
DEF456 = "actgatcgacgatcgatcgatcacgact".upper() # covert to upper case
HIJ789 = "ACTGAC-ACTGT--ACTGTA----CATGTG".replace("-","") # removing the "-"

ABC123_seq_name = ">ABC123"
DEF456_seq_name = ">DEF456"
HIJ789_seq_name = ">HIJ789"

ABC123_line = ABC123_seq_name + "\n" + ABC123 + "\n"
DEF456_line = DEF456_seq_name + "\n" + DEF456 + "\n"
HIJ789_line = HIJ789_seq_name + "\n" + HIJ789
# another way of writing it is
# ABC123_line = f">ABC123\n{ABC123}\n"


# print (HIJ789_line)


with open ("/home/tracey/Desktop/three_sequences.fasta", "w") as dna_out:

    dna_out.write(ABC123_line)
    dna_out.write(DEF456_line)
    dna_out.write(HIJ789_line)
```

3. Writing multiple FASTA files

Use the data from the previous exercise, but instead of creating a single FASTA file, create three new FASTA files – one per sequence. The names of the FASTA files

should be the same as the sequence header names, with the extension .fasta.

In [41]:
```
###############
# Solution 3a #
###############

# store the sequences in an appropriate variable name

ABC123 = "ATCGTACGATCGATCGATCGCTAGACGTATCG"
DEF456 = "actgatcgacgatcgatcgatcacgact".upper() # covert to upper case
HIJ789 = "ACTGAC-ACTGT--ACTGTA----CATGTG".replace("-","") # removing the "-"


# Note that I first printed them all, to see if they were correct, before I wrote them to the files.
#print(HIJ789)

ABC123_seq_name = ">ABC123"
DEF456_seq_name = ">DEF456"
HIJ789_seq_name = ">HIJ789"

ABC123_line = ABC123_seq_name + "\n" + ABC123 + "\n"
DEF456_line = DEF456_seq_name + "\n" + DEF456 + "\n"
HIJ789_line = HIJ789_seq_name + "\n" + HIJ789 + "\n"


# open three separate output files
# write the appropriate line to the correspondinf output file

## NB!!!! This should obviously be the path to YOUR DESKTOP
with open ("/home/tracey/Desktop/ABC123.fasta", "w") as ABC123_out, \
    open ("/home/tracey/Desktop/DEF456.fasta", "w") as DEF456_out, \
    open ("/home/tracey/Desktop/HIJ789.fasta", "w") as HIJ789_out:


    ABC123_out.write(ABC123_line)
    DEF456_out.write(DEF456_line)
    HIJ789_out.write(HIJ789_line)
```

Let's say that you find your desktop is becoming all cluttered from printing all these files

And you want to create a folder for the new output files

But you're not sure if the folder name already exists

You can import a library called `os` , and allow it to check if it exists

If it doesn't, you can tell it to create the directory

We will learn more about this in later chapters

In [42]:
```
###############
# Solution 3b #
###############


import os

ABC123 = "ATCGTACGATCGATCGATCGCTAGACGTATCG"
DEF456 = "actgatcgacgatcgatcgatcacgact".upper()
HIJ789 = "ACTGAC-ACTGT--ACTGTA----CATGTG".replace("-","")

ABC123_seq_name = ">ABC123"
DEF456_seq_name = ">DEF456"
HIJ789_seq_name = ">HIJ789"

ABC123_line = ABC123_seq_name + "\n" + ABC123 + "\n"
DEF456_line = DEF456_seq_name + "\n" + DEF456 + "\n"
HIJ789_line = HIJ789_seq_name + "\n" + HIJ789 + "\n"


# Note that I first printed them all, to see if they were correct, before I wrote them to the files.
#print(HIJ789)

## NB!!!! This should obviously be the path to YOUR DESKTOP
if not os.path.exists("/home/tracey/Desktop/Test_dir"):
    os.makedirs("/home/tracey/Desktop/Test_dir")

with open ("/home/tracey/Desktop/Test_dir/ABC123.fasta", "w") as ABC123_out, \
    open ("/home/tracey/Desktop/Test_dir/DEF456.fasta", "w") as DEF456_out, \
    open ("/home/tracey/Desktop/Test_dir/HIJ789.fasta", "w") as HIJ789_out:


    ABC123_out.write(ABC123_line)
    DEF456_out.write(DEF456_line)
    HIJ789_out.write(HIJ789_line)
```

ADDITIONAL Exercises

Additional Exercise 1

a) You have a file called **ls_orchid.fasta** on your **Desktop**

b) Read in the file

c) Check if the sequence "AAGGAT" is in the file

d) Store this in a variable called "answer"

e) Print the variable called "answer"

f) What was your answer?

g) Now make up a sequence with six characters made up of any combination of "A","T","C","G" and "N"

h) Store it in a variable name called "my_sequence"

I) Check if my_sequence is in your file

j) Print your variable called "my_sequence"

In [43]:
```
## NB!!!! This should obviously be the path to YOUR DESKTOP

with open("/home/tracey/Desktop/ls_orchid.fasta") as infile:
    read_infile = infile.read()

    #print(read_infile)
    answer = "AAGGAT" in read_infile
    print(answer)

    my_sequence = "AANTTT" in read_infile
    print(my_sequence)
```

```
True
False
```

Additional Exercise 2

a) I have a multifasta file on my desktop, called **ls_orchid.fasta**

b) Use BioPython to read in the file and show me all the sequence IDs

c) Print out a glimpse of the nucleotide sequences for each fasta sequence

d) Print out the length of each sequence

In [44]:
```
%cd /home/tracey/Desktop/

from Bio import SeqIO
for seq_record in SeqIO.parse("ls_orchid.fasta", "fasta"):
    print(seq_record.id)
    print(repr(seq_record.seq)) # if ou don't use the "repr", you get the full sequence
    print(len(seq_record))
```

```
/home/tracey/Desktop
gi|2765658|emb|Z78533.1|CIZ78533
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC')
740
gi|2765657|emb|Z78532.1|CCZ78532
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAG...GGC')
753
gi|2765656|emb|Z78531.1|CFZ78531
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGCAG...TAA')
748
gi|2765655|emb|Z78530.1|CMZ78530
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAAACAACAT...CAT')
744
gi|2765654|emb|Z78529.1|CLZ78529
Seq('ACGGCGAGCTGCCGAAGGACATTGTTGAGACAGCAGAATATACGATTGAGTGAA...AAA')
733
gi|2765652|emb|Z78527.1|CYZ78527
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...CCC')
718
```

Additional Exercise 3

a) Open a file called ***test_outfile.txt*** for writing

b) Write the following sentence into the file:
This is the message we want to change to upper case

c) Now open the file ***test_outfile.txt*** into which you just wrote your sentence

d) Read the contents of the file in uppercase

e) Store this in a variable name called **uppercase_sentence**

e) Print this variable to your screen

In [46]:
```
## NB!!!! This should obviously be the path to YOUR DESKTOP
with open("/home/tracey/Desktop/test_outfile.txt", "w") as outfile:
    outfile.write("This is the message we want to change to upper case")

with open("/home/tracey/Desktop/test_outfile.txt") as infile:

    uppercase_sentence= infile.read().upper()
    print(uppercase_sentence)
```

```
THIS IS THE MESSAGE WE WANT TO CHANGE TO UPPER CASE
```

Additional Exercise 4

a) Open the ***genomic_dna.txt*** file

b) Strip any white space from the file

c) Create a variable called "count_AT"

d) Count the number of As and Ts and add them together

e) Print the result to a file called "count_AT.txt"

f) Print ***The results were printed to an output file*** to your screen

In [47]:
```
# open your files

## NB!!!! This should obviously be the path to YOUR DESKTOP
with open("/home/tracey/Desktop/genomic_dna.txt") as infile,\
    open ("/home/tracey/Desktop/count_AT.txt", "w") as outfile:

    read_infile = infile.read().strip() # strip off empty spaces from both ends
    count_AT = str(read_infile.count("A") + read_infile.count("T"))

    outfile.write(count_AT)
    print("Results printed to a file")
```

```
Results printed to a file
```

In [48]:
```
# same as above

## NB!!!! This should obviously be the path to YOUR DESKTOP
infile = open("/home/tracey/Desktop/genomic_dna.txt")
outfile = open("/home/tracey/Desktop/count_AT.txt", "w")

read_infile = infile.read().strip()
count_AT = str(read_infile.count("A") + read_infile.count("T"))

outfile.write(count_AT)
print("Results printed to a file")

infile.close()
outfile.close()
```

```
Results printed to a file
```