Tutorial 2 - Lecture catch up, including finding things with The Unix Shell

`find`

- powerful search function
- searches for files, not inside files
- **when would this be useful?**

`$( )` extends the utility of `find`

If we wanted to use Unix to count the number of lines in each of the *.pdb* files in the `data-shell/molecules` directory. We could...

```
wc -l $(find . -type file)
```

Why does this work?

`grep`

- powerful search function
- searches inside files or any information passed via stdin
- by default considers entire lines containing matches

`grep` application

Copy and paste contents of *DNA.txt* from Lecture 3 directory on Sakai into `nano` and save as *DNA.txt*

Challenge: Building a simple gene finder - use Unix to count the number of start codons (ATG), and therefore potential genes, in the DNA sequence contained in *DNA.txt*

`sed`

`grep` is to find, as `sed` is to find and replace

The `sed` utility reads the specified files, or the *standard input* if no files are specified, modifying the input as specified by a list of **commands**. The input is then written to the *standard output*.

Each line of a file, except the newline character is copied into a **pattern space** where it is acted on by `sed` commands.

Two of the most common and straightforward uses of `sed` are:

- `sed 's/pattern/replacement/flags'`
  - common flags: N - replace the Nth occurrence, g - replace all occurrences, not just first (default)

- `sed 'y/string1/string2/'`
  - each character in string1 is replaced by the corresponding character in string2; this is a multi-version of `tr`

Challenge: Imagine we wanted to visualize all of our matches to the start codon *ATG* in *DNA.txt*. How would we use `sed` to do this?

Challenge: use `sed`, along with `tr` and `rev`, to create the reverse complement of the sequence contained in *DNA.txt*