Introductions:

Who am I, My creds
Course people to introduce themselves and what they would like to get out of this course, any Perl Background
What this course in intending to teach
- What Perl is
- how to use it
- Some tips to producing good code
- The modern way of writing perl
- Looking at real code

A brief explanation:

Larry produced Perl1 by ripping apart Awk in 1987
Most commonly used version 5.8.8 but 5.8.x is approx 10 years old
Now entering a yearly release cycle, starting with 5.12 this year (5.10.1 previous stable version), 5.14 is this year

Went through a period of being the language for bioinformatics, web development and systems administration. There is plenty
of support for all of these, and it has been a major language on site.

Excellent support for text parsing/mining, many other languages see the regex for perl as to be the one to emulate

Another great feature of Perl is that once you know a bit, you can do a lot without knowing everything.

Running a perl program:

hello_world.pl

Introduce how a script is constructed and runs
Produce a hello world script

Errors (p190, p178)
Show errors by making a script which doesn't run

Variables: (p21)

Numbers and Maths functions (p31)
Integer is a number
doing math functions (p90)
++
0 is false
Strings (p23)
any sequence of characters quoted
numbers can be strings
concatenation (p90)
empty string is false
Scalars (p51), Arrays (p54), Hashes (p60)

- How to assign/declare
- Benefits
- my/our
- undef (p32)
  - it is false
  - warnings if you try to do any string or number operations on it
- References
  - What is a reference?
    - like a copy of the original
    - pointer to the memory location that contains the information
  - Dereferencing
    - It's how we retrieve the information in the reference
  - Multilevel Data Structures (p82)
    - Create a table, headed and not
  - What are the benefits of using them?
    - Can reduce memory compared to copying
    - If you modify via the reference, you modify the original
  - Drawbacks?
    - If you modify via the reference, you modify the original

IO:

- Inputs:
  - @ARGV
    - Command line arguments
  - STDIN (p193)
    - Interaction with the program as it runs
  - DATA
    - Data input found at the end of a file
    - Useful for keeping fixed data with the code that will use it
- Outputs:
  - STDOUT
    - The default place where output is sent to
  - STDERR
    - The default place where error and warning messages are sent to

Regular Expressions: (p131)

- The syntax
  - my $sequence =~ m/aaa/ixms;
- Match
  - find a codon in some sequence
- Substitute
  - convert one codon to another
- Transliterate
  - get the other strand of some sequence

Decision Making

- Control Flow (p34)

if/else
while
operators && || //
eq/ne ==/!=
my $value = x ? y : z;

File operations: (p193)

File::Slurp
Reading from a file
Writing to a file

Can be done without File::Slurp, how to 'open'/'close'

Functions: (p93)

What is a function?
keywords are functions - they do something i.e. print, open, readFile, writeFile
A piece of code which does a job
Benefits
No need to repeat the same code multiple times - it's reusable
Difference from method - none
Scope
A function doesn't generally know much from outside of it, or affects anything outside of it
Takes input parameters
An array
Returns outputs
An array
Writing a function
Count the number of A's in a Sequence
Extending a function
Count the number of a requested base in a Sequence
Functions can call functions
count_a, count_c, count_g, count_t, count_n
This example is a bit of overkill, but principle is shown

Modules: (p21)

What is a module?
.pm file
How does it differ from a script
script 'use's it
reusable code between scripts

```
package My::Module;
1;
```

```
use My::Module;
```

Sourcing a local lib
use lib qw{lib /some/path/to/lib};
PERL5LIB

Writing a module
Transfer our Sequence Base counter to a module
sequence_counter.pm
Export that to the script
Default export
Optional export

CPAN (p15)
Someone has probably done it before you

Objects: (p148)

What is an object?
How does it differ from a Module
Store the data with the methods which manipulate it

package name becomes class

Moose - the simplest way to create an object, and the most modern!
use Moose;
Add an attribute to hold the sequence
has q{sequence} => (
isa q{Str}, is => q{rw},
);
Modify method to act on the stored sequence
Convert script to use this
Add in our function to locate a codon

Creating an object from scratch
bless a reference in a new method
creating attribute holders
getter
setter
accessor
You can see why use Moose; is better
write less to get as much, and more

The combination of scripts, modules and objects

How do we join a script, module and object together to make a coherent script
What part might be suited to which?
Why that might be?

Testing: (p184)

What is a test?
Something which can check you get expected outputs from given inputs
What is a test script?
A file of test assertions which typically would check a whole file of code
What are the benefits?

If code get changed in the future, we can ensure that no previous functionality changes
How do we run it?
prove
We'll write a test suite to check our sequence module

Test First, Test often
The paradigm of modern software development
Write a test for a new method on our sequence module object
Run the tests
Write the code
Run the tests

One liners:

Running a perl one-liner
Very useful for quickly seeing if something is going to do what you expect
perl -e 'print qq{hello world\n};'
Writing a perl one-liner
-l => perl -le 'print q{hello world};' # we have added a default to print a newline character on all prints, just like say
Don't do this if you expect it to be run more than once
Don't do this if you expect it to be run at all

If Extra time:

Lets look at some real production scripts!