Combining Scripts, Modules and Objects

Aims

- Have an idea of when to use
  - scripts only
  - modules
  - Objects
- Have an idea of how they relate to each other

When do I use a each?

- When you want to develop, there is no reason not to only use scripts if that is what you are comfortable with, but let's look at why you want to consider all the three options.

Script?

- The backbone has to be a script in some format.
- Important features:
  - a) Name - make the name relevant
  - b) Keep the code flow readable.
  - c) Use functions instead of copy and paste
  - d) Only keep code which cannot be used elsewhere, in the script
  - e) any interactivity is best kept to the script

Module?

- If you are writing lots of functions, then you should consider breaking these out into a module.
- Important features:
  - a) Name - make the name of the module(s) relevant to the functions they provide.
  - b) Export the functions
  - c) Make the functions as 'black box' as possible - i.e. *find_sequence* rather than *find_codon*

Object?

- Functions need to be very specific? Trying to represent data? then you should consider making an object.
- Important features:
  - a) Name - it represent the object
  - b) Methods should be relevant to the object
  - c) Make your attributes relevant – i.e. Sequencer attributes that might be relevant, *flowcell, camera, reagents* rather than *contents, equipment*

Example – Produce a Cup of Tea

- Script: make_tea
- This is a process. So we want the script to go from start to end
  - 1) get kettle
  - 2) fill kettle with water
  - 3) boil water
  - 4) get cup
  - 5) get teabag
  - 6) put teabag in cup

Example – Produce a Cup of Tea

- Script: make_tea
  - 7) add boiling water to cup
  - 8) allow to brew
  - 9) remove teabag from cup
  - 10) add sugar
  - 11) add milk
  - 12) add lemon
  - 13) drink
- This is what it needs to do.

Example – Produce a Cup of Tea

- Modules: TeaHelper
- Are there any processes which are repeated, and could be put into a module
- adding something to something else?

Example – Produce a Cup of Tea

- Objects: Kettle, Cup, Teabag
- What do each of them have/do
  - Kettle
    - filled with water, needs to boil water,
    - needs to be able to be poured
  - Cup
    - able to have a teabag, water, milk, sugar,
    - able to remove a teabag, be drunk
  - Teabag
    - can be fresh/spent

Example – Produce a Cup of Tea

- So we can see options to utilise all three,
- although in practice, we probably wouldn't write a module just to provide the adding function, unless we could see benefits outside of making a cuppa.

Summary

- This short section was really just to give you an idea of what you want to think about when developing a new perl program.
- There are plenty of further options, and other things you can do.
- Perls motto of 'There is more than one way to do it' is as applicable here, as when actually writing your code.

Summary

- Think about what you want to achieve first, you can solve almost all of your problems really sensibly.
- What will help you focus more on that though are tests, and that is what we shall now look at.