Consolidation

The following are some things to think about, find answers to, and generally bear in mind, especially when applying the things from this course to your real research.

Answers (or at least hints) are in a second document where you found this one

01 Background

- What is HPC?
- What are the most important things to remember when using computing for research?
- What is an OS? What are you running? What processor do you have?

02 Notebook to Script

- What are notebooks? How are they different from scripts?
- Can notebooks have any unwelcome consequences?
- Can you do anything in a notebook that you can't do in a script? What about vice versa?
- What changes might you make when moving from a notebook to a script?
- How can you keep a library of code for tasks you do often?
- What habits do you have that might lead to buggy or poorly maintainable code? How can you remedy them?

03 Clusters, Queues and Modules

- Which of the following are untrue? Which might get you a very upset email from a system administrator?

    - You're allowed to do anything on a cluster - if you shouldn't do it, it wont be possible
    - Clusters are great for all your small, short jobs so your laptop doesn't get warm
    - Documentation is mostly pointless and you don't need to read it
    - Running on a cluster is simple once you know how
    - Queues are designed to slow down your workflow and annoy you
    - There's no such thing as fair sharing of resources - you should just get to use as much as you want
    - Special resource nodes are a great way to jump the queue
    - If you can arrange your work to run when systems aren't busy, or exploit underused resources, you can get resources faster
- What sorts of parallelism are there? How can you run a program across multiple compute nodes (machines)? How can you run a bunch of copies of a code with different inputs?
- What should you do if you need a new module installed for your work?
- What queueing systems are in use on systems you might use? Can you use the same scripts between them (if more than one)?
- How can you keep track of when your jobs run and whether they succeed or fail?
- (Humour) What might be a sys-admin's favourite email subject/bug title?

- 04 - Skills - Checkpoints
    - What things are important to keep in mind when checkpointing code?
    - Complete this - if you can't continue from a checkpoint it is .....
    - What sorts of problems might need "exact" continuation? What sorts only need statistical correctness?
    - How can you automate checkpoint-and-continue workflows using the job scheduler? Is this a fair workflow?

- 05 - Skills - Containers
  - What are containers for? Why are they worthwhile?
  - Where can you find pre-built containers?
  - What is provisioning?

- 06 - Skills - Numba
  - What is compiling for? What does JIT mean?
  - What can the jit do for your code?
  - What tricks are there for paralellism in numba?

- 07 - Skills - High Performance Libraries
  - What reasons are there to use libraries?
  - What problems can they cause?
  - How can you make your life easier if you have to change out a library for a different one in future?

- 08 - Skills - GPUs
  - What is a GPU? Why are they good for parallel computing?
  - What sort of problems are suited for GPUs?
  - Are there ways to seamlessly "roll-over" between using a GPU and not using one?
  - Suppose I have a lot of passes to do over a small amount of data. What is likely to be the bottleneck in running this on a GPU?