Code re-use

- Re-use of existing code can potentially/significantly reduce development time for new projects
- Targetted code needs to be flexible enough to re-use
- Need to be able to find suitable code to re-use (possibly our own, or possibly written/maintained by others)
- Need to be able to trust the code we want to re-use
- Need to address maintenance of both our own code and the re-used code

When is re-use worth it

- When re-using code, it might not do exactly what we want, so often need to customize it or wrap it in other code to tweak its behaviour to give desired results

- Need to balance the time savings from using pre-existing code against costs to incorporate it in project (factoring in testing, maintenance, safety, etc)

- How much can we trust original source of code (both for code integrity at start and for long term maintenance etc)

Where do we find re-usable code

- Existing libraries provided with our language tools
- Trusted third party extensions/libraries
- Our own/in-house extensions/libraries
- Searching for suitable code often challenging – relies on decent documentation/catalogs of what's available
- Impacted by licensing/rights issues

Writing re-usable code

- Write the code to be stable, robust, well commented

- Test the code thoroughly

- Write the code to be generic (e.g. a function to get integer in range m to n with prompts taken as parameters, instead of a function that is hard-wired to get an age in range 18-65)

- Write ADTs to work off flexible data types (e.g. generic stack instead of a stack of ints or stack of floats)

- Add documentation on the re-usable code to a suitable searchable catalog so you/others can find it in the future

Code libraries

- Where permitted, make use of existing code libraries
- Know the common libraries for your language, and how to search them
- e.g. for C++ be familiar with (at least) the C standard libraries and the C++ standard template libraries (STL)
- Find good reference sources (e.g. cppreference.com, www.cplusplus.com, etc)