Writing Good Research Software

Peter Hill

Physics Coding Club

- Weekly, one hour informal seminar or two hour practical session
- Open to everyone!
- Upcoming topics:
  - Hands on intro to version control
  - Hands on testing
  - Containers
  - Cloud Computing
  - Project structure

https://physicscodingclub.github.io/

What is Research Software?

- Software used to generate, process or analyse results
- Might be enormous 100k lines of complex simulation code
- Might be 100 lines for pulling data off of instrument
- Might be 10 lines to plot results
- Could even be a spreadsheet!

Why is Research Software important?

From a 2014 survey:

- 90% of researchers use research software
- 70% said they couldn't do their research without it

=> Most modern research is impossible without some form of software!

Why is **good** Research Software important?

- It’s important to be able to trust the results are correct
- If the results are *possibly* wrong, then the research is suspect
- If the research is suspect, it’s not science!

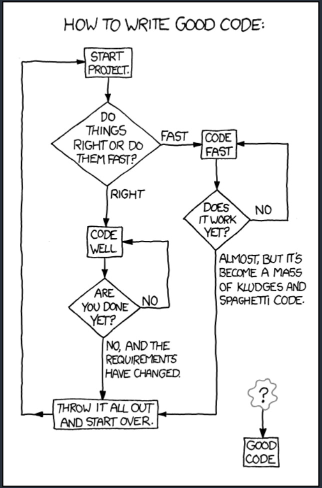
What goes into good research software?

- “Trustable correctness”
- Easy to use
- Easy to maintain
- Portable
- Efficient
- Under version control

Sustainability

“The capacity for software to endure” - Daniel S. Katz

How to write good software


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

xkcd #844

Why use tests?

All software contains bugs!

- Tpyos
- Maths errors
  - Wrong equations used?
- Logic errors
  - Code doesn't do what you think it does
- Edge-case errors
  - Unforeseen sets of circumstances

Not just a theoretical concern!

Bugs have real world consequences:

- Software on Mars Climate Observer mixed up metric and imperial measurements
- Bug in Therac-25 radiation therapy machine delivered "massive overdoses of radiation"
- F-22 Raptor aircraft crashed due to software "malfunction"
- Citigroup fined £5m for mistaking real data for test data for 15 years

Tests are vital

- Tests let you prove that the software is correct
- Tests provide trust that the research is correct
- Tests give you confidence to make changes to the code
- Often faster to write tests first

If it doesn’t have tests, it’s wrong!

Testing software

Types of tests

- **Static analysis**: checking the source code
- **Runtime testing**: checking program is still in a “good state”
- **Unit testing**: checking individual parts of the code are correct
- **System/integrated testing**: checking the program as a whole is correct

Runtime testing

- Most runtime checking can be thought of as either a precondition or a postcondition

Preconditions

- Things that must be true of function inputs:

```python
def square_root(number):
    if number < 0:
        raise ValueError("Can't take square root of negative number")
    ...
```

Runtime testing

Postconditions

- Things that must be true of function results:

```python
def square_root(number):
    ...
    if result < 0:
        raise ValueError("Result of square root was negative somehow")

    return result
```

Runtime testing

Advantages

- Always there!
- Catch errors and unexpected edge cases early, before they escalate

Disadvantages

- Have to decide if we can carry on or if should we just give up
- Can slow things down

Unit testing

- Catch bugs as soon as possible
  - Preferably during development!
- Test individual components
- Test range of inputs
- If the building blocks are correct, the whole thing is more likely to be correct
- Good test coverage helps you make changes

Unit testing

An example

```
def test_square_root_4():
    assert square_root(4) == 2

def test_square_root_minus4():
    with raises(ValueError):
        square_root(-4)
```

Unit testing

Comparison to known values

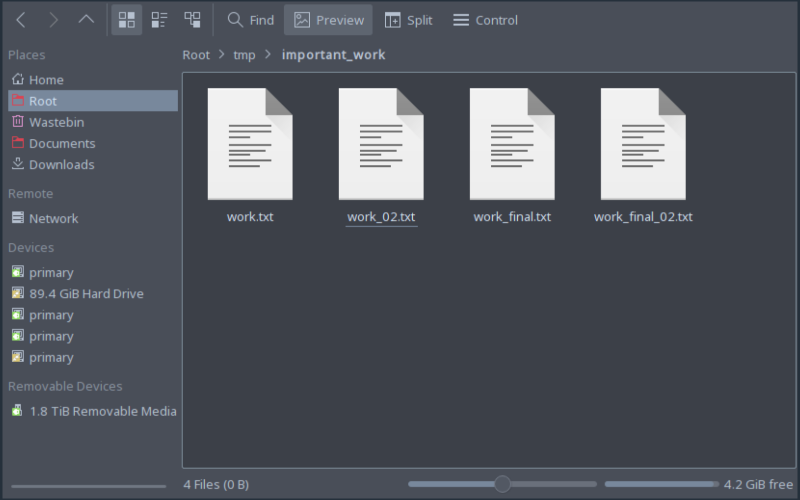
- Best to compare to some analytic or exact result
  - Might be a simpler problem
  - Might be a carefully constructed problem
- Can you compare to a “known good” result?
  - From a different implementation?
  - different algorithm?
  - different software?
  - previous version?
- Can be difficult with scientific software, we might not know the correct answer!

Unit testing

Logic checking

- Does the answer make physical/mathematical sense?
  - Does the molecule have positive mass? Is it travelling slower than the speed of light?
  - Does the particle move in the correct direction under these forces?
  - Is the inverse of the inverse the identity transform?
- Does the answer make programmatic sense?
  - Is the average of a set of numbers less than the maximum value?
  - Does appending a value increase the size of the container?

What is version control?



Is_this_version_control_meme.jpg

What is version control?

- Version control systems record changes to a file/set of files over time
  - Not just software! This talk is under version control
  - Allows you revert files back to a previous state, compare changes over time, see who last modified something, etc.
- Instead of keeping multiple copies of the same file, normally just store the *differences* (“diffs”) between versions of the files

Why is version control important?

- Tracking versions
  - Know instantly which is the latest version
  - Roll back to previous versions
  - See history of project/file/line
  - Find out when bugs were introduced
  - Maintain/compare different versions
- Coordination between developers
  - Easier to keep track of when changes are made
  - Easier to work on separate features
  - Easier to merge distinct changes from separate developers
  - Easier to resolve conflicts on same features
  - Tracking who made what changes

If it's not under version control, it doesn't exist!

Documentation

- Good documentation makes software easier to use and easier to maintain
- Documentation is for you in six months!
- Everyone hates writing documentation; everyone hates missing documentation
- What counts as documentation?
    - A “README” file
    - An instruction manual
    - A reference manual
    - Code comments
    - Names!
    - Version control commits

Naming things

- Readability counts!
- Help reduce cognitive load required to understand
- Don't needlessly abbreviate
- Don't just type up maths

Example

```
def calcf(r, p):
    ...

def calculate_force(position, momentum):
    ...
```

Resources

- Physics Coding Club!
  - https://physicscodingclub.github.io/
- Research Computing Training and Support
  - https://wiki.york.ac.uk/display/RCTS/Research+Computing+Training+and+Support
- Research Software Engineering Association
  - https://rse.ac.uk/events/rse-webinar-series/
- Software Sustainability Institute
  - https://software.ac.uk/
- Working Effectively with Legacy Code by Michael C. Feathers
- Some material in this talk adapted from https://chryswoods.com/talks