Coding and Software

Best Practices

"To make your work easy and your life infinitely better."

Grant Kendrick Parker
grant.parker@mavs.uta.edu

IceCube Bootcamp 2022

Thanks to Dr. Alex Olivas for much of the inspiration and content behind these slides.

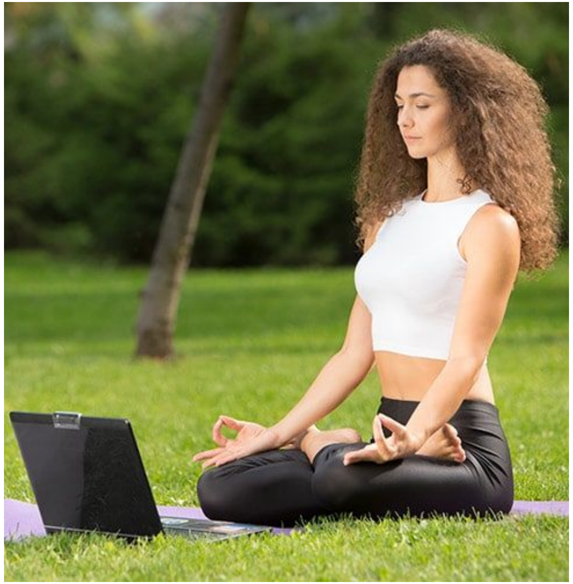
my program: *works perfectly*

me: *cleans up the code*

also my program:

Introduction and Introspection

- **Things to know going in:**
  - Code is ***frustrating***, embrace the chaos!.
  - Everyone codes ***differently*** (part of the frustration).
  - Find your coding zen, everyone is constantly learning.
- **What are “best practices”?:**
  - Confession: I am not a “software person”
  - I also didn’t know much code when I started.
  - These are good things!
  - “Best practices” are techniques for making your code the most ***readable*** and ***runnable*** for other people.

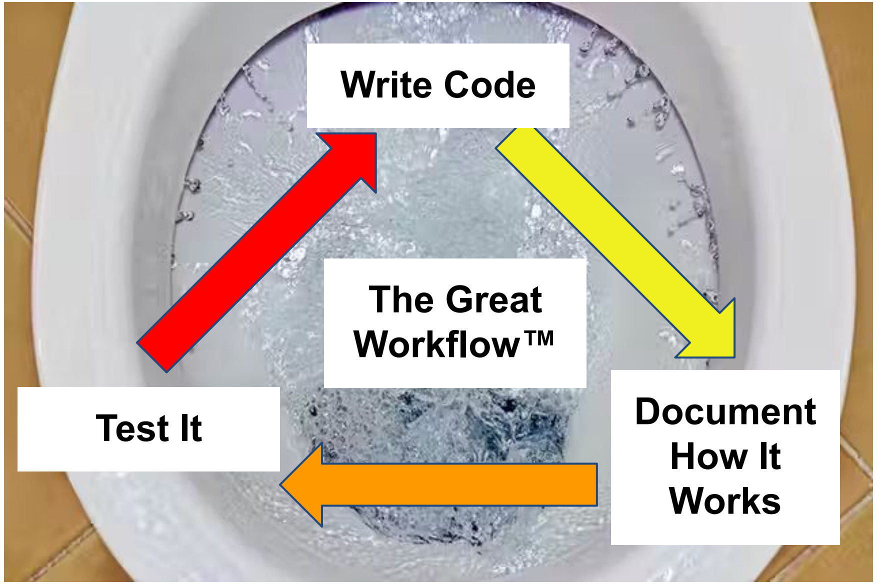


Embrace the chaos of code, find your peace.

Talk Outline

The Cycle of Code and Software Work:

- **Code**– The thing that does the task you specify.
- **Documentation**– The thing that tells a new person (or expert) how your code works (***WRITE A LOT AND IN SIMPLE TERMS***)
- **Tests**– The thing that makes sure your code works the way you say it does.
- **Workflow and Version Control**– How you do all of these and make your code better at the same time!



Sometimes it's a beautiful whirlpool. Other times, it's a toilet. Don't let your workflow be a toilet.

Picking Your Code

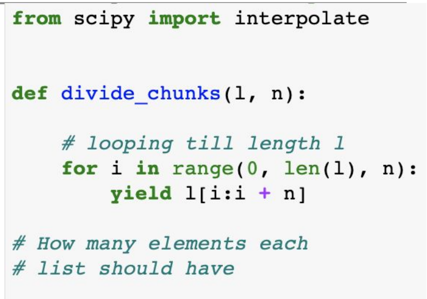
Pros and Cons of Different Languages:

- In IceCube, our software is generally written as a combination of two languages: **Python** and **C++.**

- **Python**:
  - Reads a lot like normal english, you just have to learn the “grammar”.
  - ***Slower*** and ***less powerful*** than C++ (only noticeable for *big calculations*)

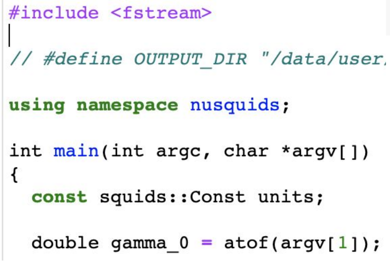
- **C++**:
  - Much ***harder to read, more elements and structure*** to learn and keep in mind.
  - Suitable for big and complex calculations, let’s you optimize memory usage and machine capabilities.

Python



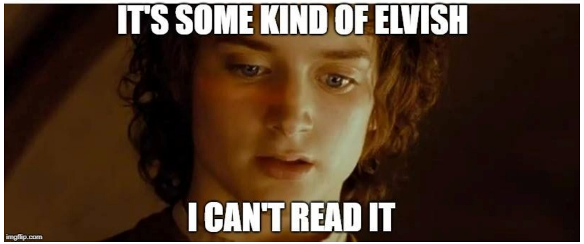
```python
from scipy import interpolate

def divide_chunks(l, n):

    # looping till length l
    for i in range(0, len(l), n):
        yield l[i:i + n]

# How many elements each
# list should have
```

C++



```cpp
#include <fstream>

// #define OUTPUT_DIR "/data/user

using namespace nusquids;

int main(int argc, char *argv[])
{
  const squids::Const units;

  double gamma_0 = atof(argv[1]);
```

When you trying to look at the code you wrote a month ago



Make Your Code Good

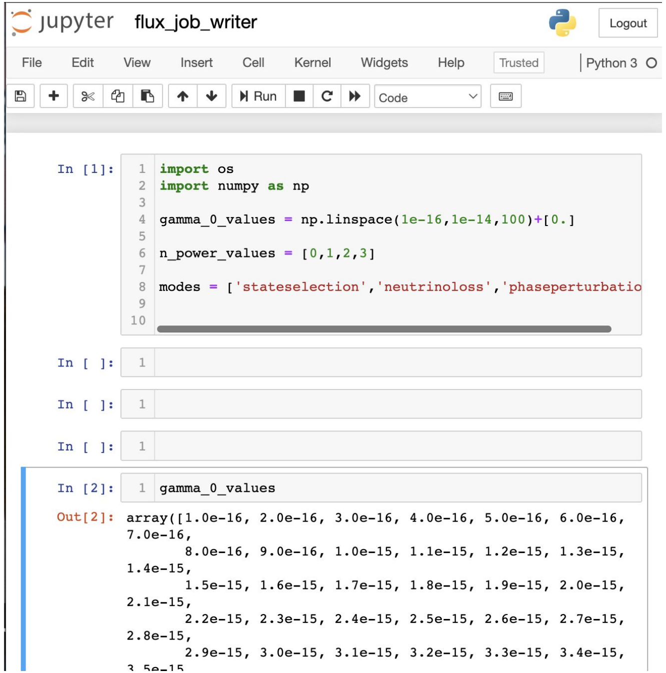
Keys to Efficient Code

- Often, IceCube software is a **combination of Python and C++**: Be wary *and* aware.

- Keep your code ***modular***:
  - A function should perform a single purpose with minimal operations.
  - Easier and faster to fix smaller parts.
  - Extend to big picture: **Good software does *ONE* thing *REALLY WELL*– *not EVERYTHING really BADLY*.**

- Write your code with an ***editor***– PyCharm, Emacs, Jupyter (iPython) Notebooks, JupyterHub.

- Write your code with ***consistency***; use a **style guide** and **style checker (linter)**:
  - **Guides: PEP8, Google Employee Guide, C++.**
  - **Linters: C++ (here and here), Python (here and here)**



In [1]:

```
import os
import numpy as np

gamma_0_values = np.linspace(1e-16,1e-14,100)+[0.]

n_power_values = [0,1,2,3]

modes = ['stateselection','neutrinoloss','phaseperturbatio
```

In [ ]:

In [ ]:

In [ ]:

In [2]:

```
gamma_0_values
```

Out[2]:

```
array([1.0e-16, 2.0e-16, 3.0e-16, 4.0e-16, 5.0e-16, 6.0e-16,
7.0e-16,
       8.0e-16, 9.0e-16, 1.0e-15, 1.1e-15, 1.2e-15, 1.3e-15,
1.4e-15,
       1.5e-15, 1.6e-15, 1.7e-15, 1.8e-15, 1.9e-15, 2.0e-15,
2.1e-15,
       2.2e-15, 2.3e-15, 2.4e-15, 2.5e-15, 2.6e-15, 2.7e-15,
2.8e-15,
       2.9e-15, 3.0e-15, 3.1e-15, 3.2e-15, 3.3e-15, 3.4e-15,
3.5e-15
```

Documentation

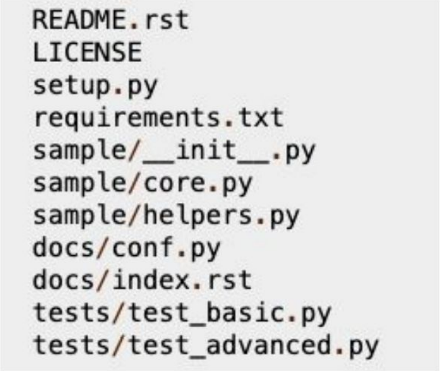
<u>Structure of A Code Project</u>

- Organized code is good code. The right figure demonstrates the basic universal structure.

- **README**: The document every new user will read before using new code.
  - Describe the general purpose of the code
  - Explain how to install and compile on a local machine.
  - <u>Write clear README's with lots of description, do not skimp on this step.</u>

- **docs Repository:** Often documentation for a new project can be generated by running a script. This has many advantages, so consider learning this practice.

- **LICENSE:** Whether you make a project on your own or contribute a feature to a project, code can be released for public use. <u>*Keep your contact information updated with IceCube, you may be reached for licensing purposes*</u>**.**



```
README.rst
LICENSE
setup.py
requirements.txt
sample/__init__.py
sample/core.py
sample/helpers.py
docs/conf.py
docs/index.rst
tests/test_basic.py
tests/test_advanced.py
```

Documentation

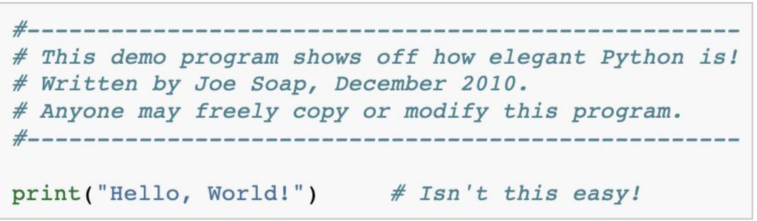
How to Document Your Code

- Two main ways to document: comments and tech notes.

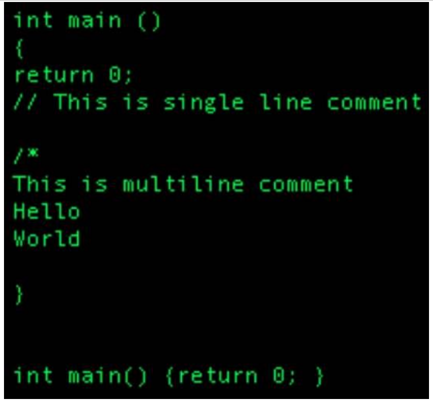
- **Comments**: Lines in your code where you explain what's happening.
  - Comment ***everywhere*** and ***a lot.***
  - Use comments to explain what the code does at each step.
  - Docstrings: Comments inside functions that specify the function's purpose.

- **Tech note:** A paper that describes the full details of a software suite.
  - Explain motivation, include relevant mathematics, physics, plots.
  - Show examples for all/common scenarios.
  - List classes and functions with definitions.
  - Get a paper out of it?

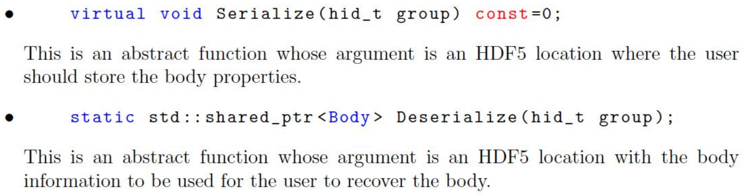
- **DOCUMENTATION SAVES LIVES, DO IT**



```
#-----------------------------------------------------------
# This demo program shows off how elegant Python is!
# Written by Joe Soap, December 2010.
# Anyone may freely copy or modify this program.
#-----------------------------------------------------------

print("Hello, World!")      # Isn't this easy!
```



```
int main ()
{
return 0;
// This is single line comment

/*
This is multiline comment
Hello
World

}


int main() {return 0; }
```



- `virtual void Serialize(hid_t group) const=0;`

  This is an abstract function whose argument is an HDF5 location where the user should store the body properties.

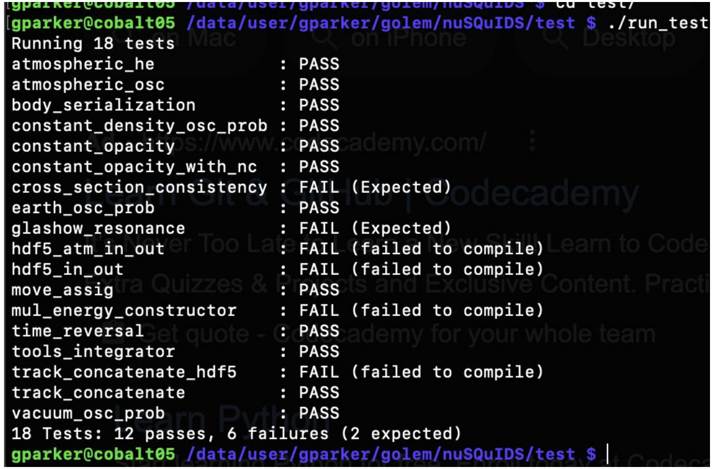
- `static std::shared_ptr<Body> Deserialize(hid_t group);`

  This is an abstract function whose argument is an HDF5 location with the body information to be used for the user to recover the body.

Brief Comment on Testing

Testing Standards

- Tests verify that all code functions operate successfully and as designed.
- Usually, tests are run collectively by calling a **single script** that produces verbal output.
- Tests should **sample all minimal examples** and the full range of classes + functions of the suite.
- Describe your tests in your docs.
- Test suites help write and organize this. Example: python unittest
- **MAKE SURE TESTS WORK BEFORE RELEASING NEW VERSIONS OR MERGING (next slides).**



```
gparker@cobalt05 /data/user/gparker/golem/nuSQuIDS $ cd test/
gparker@cobalt05 /data/user/gparker/golem/nuSQuIDS/test $ ./run_test
Running 18 tests
atmospheric_he               : PASS
atmospheric_osc              : PASS
body_serialization           : PASS
constant_density_osc_prob    : PASS
constant_opacity             : PASS
constant_opacity_with_nc     : PASS
cross_section_consistency    : FAIL (Expected)
earth_osc_prob               : PASS
glashow_resonance            : FAIL (Expected)
hdf5_atm_in_out              : FAIL (failed to compile)
hdf5_in_out                  : FAIL (failed to compile)
move_assig                   : PASS
mul_energy_constructor       : FAIL (failed to compile)
time_reversal                : PASS
tools_integrator             : PASS
track_concatenate_hdf5       : FAIL (failed to compile)
track_concatenate            : PASS
vacuum_osc_prob              : PASS
18 Tests: 12 passes, 6 failures (2 expected)
gparker@cobalt05 /data/user/gparker/golem/nuSQuIDS/test $ |
```

Workflow

How Does GitHub Work?

- GitHub has many advantages:
  - Multiple people can contribute to the code.
  - Excellent for version control and feature management.

- GitHub structure can be hard to learn– study hard and practice, it will pay off. [GitHub Guide]

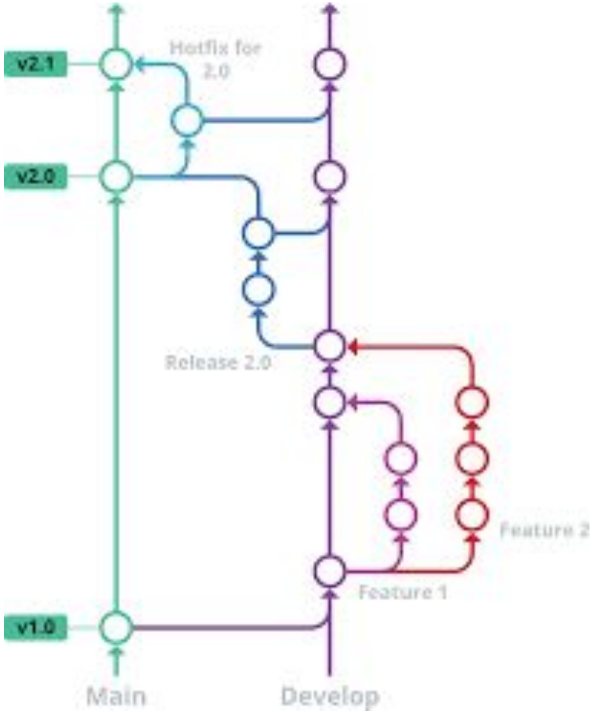
- An example workflow is diagrammed on the right.

- Scenario: You start a project that requires modifying some IceCube software. What are the steps?

  - **Sign into GitHub and find the repository.**

  - **Create a branch** (contact the repo owners or #icecube-it on Slack)

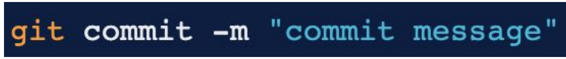
  - **Experiment→Test→Commit→Merge! (kinda)**



Tracking Your Edit History

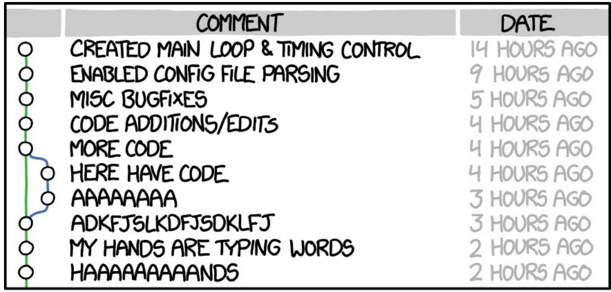
How to Make Commits Correctly

- The most important thing you'll do is make commits.
  - Commits upload your code changes to your branch (local machine→GitHub)
  - The "commit message" explains the update. **ALWAYS INCLUDE A COMMIT MESSAGE.**



```
git commit -m "commit message"
```

- Rules for commits:
  - Should be narrowly-focused: one update at a time. This means **update small** and **update often**. **Think of it as a lab notebook for your code: record everything!**
  - Do not combine stylistic/organization updates with functional updates.
  - Write **detailed** and **straightforward** commit messages.
  - Test that the update 1) works, 2) compiles against the main branch, and 3) that all tests pass.



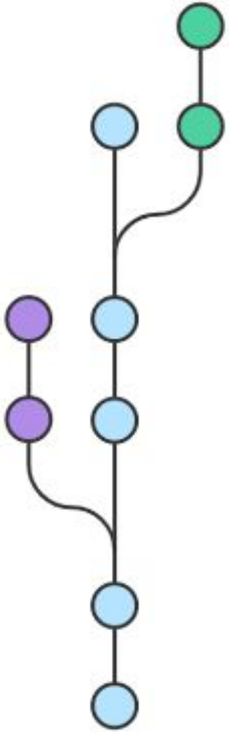
	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Prune Your Tree

The Purpose of Branches

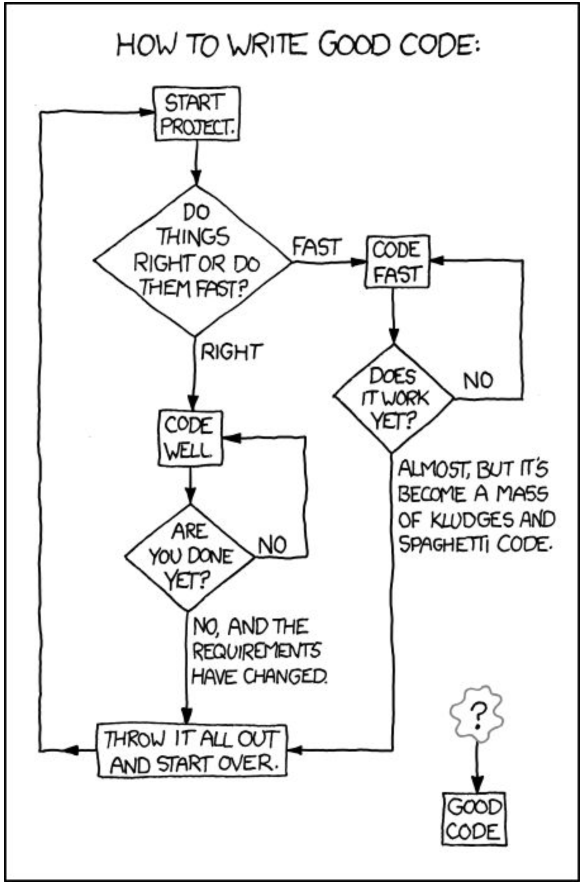
- In general, branches are used to experiment and add *features* to software that are integrated after thorough testing.
- In IceCube, we often use branches for ***analyses***.
- Name your branch after your name and analysis.
- Often, unless your work is to improve/add features to widely-used software, analyzers ***do not*** attempt to merge with the main branch (this varies between groups).
    - If you do your analysis right, your code is independent of the software infrastructure and you can go ahead and merge.



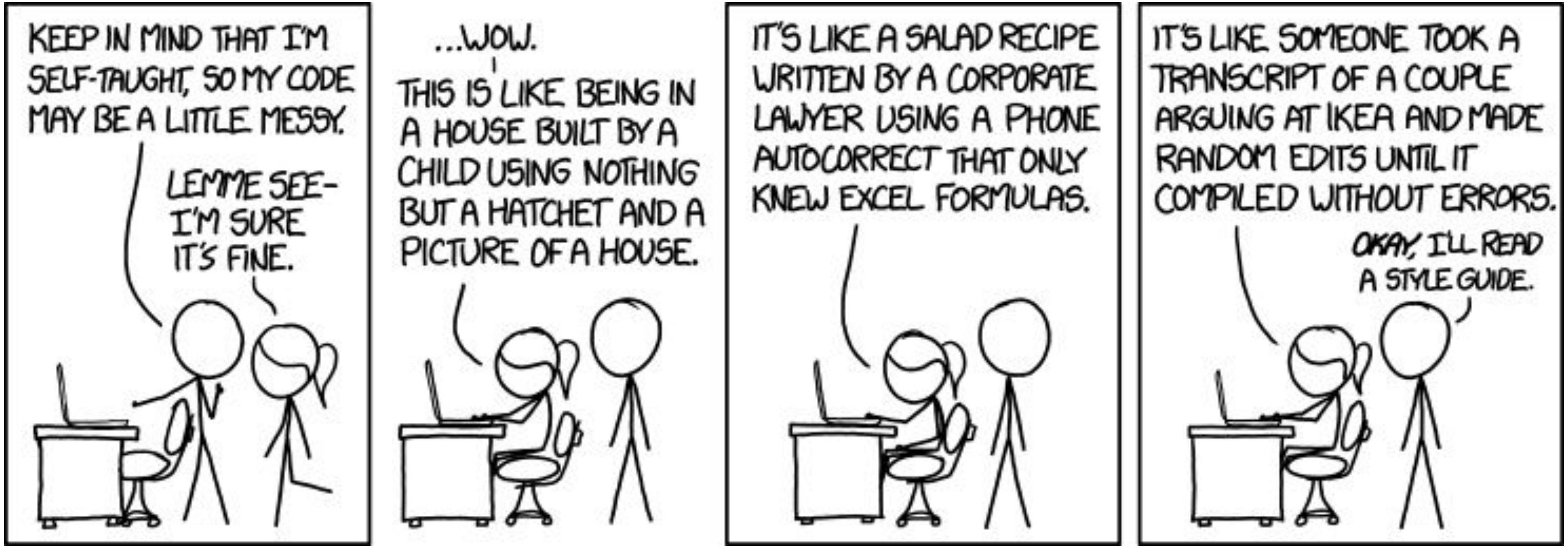
Last Bits of Advice

Things You Should Know

- Go to **#software** in the IceCube Slack for any questions/errors with code.
    - Also ask in your Working Group channels (do this first!)
- Go to **#icecube-it** for any hardware-related issues.
- When in doubt, ask!
- Learn and use GitHub now so you don't have to during a code review!
- Best coders learn both from practice and reading, try this free site: hackerrank.com
- Nobody knows everything, we collaborate to help each other learn and succeed.


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

Thank You


KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.
LEMME SEE– I'M SURE IT'S FINE.
...WOW.
THIS IS LIKE BEING IN A HOUSE BUILT BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.
IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.
IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.
OKAY, I'LL READ A STYLE GUIDE.