Hitchhiker's Guide to Debugging

Codemash 2013
Carol Nichols and Jake Goulding

1. DON'T PANIC.
   - Changing code out of panic is likely to make things worse.
   - Debugging is a scientific process of making hypotheses and testing them.

2. Narrow down the probability of the location of the bug from an infinite number of places to one.
   - Prove that code is or is not part of causing the problem by trying to reproduce the bug with or without parts of the code.

3. Love stack traces, even though they're as horrible as Vogon poetry.
   - Pick out the resulting error message and the portions of the stack trace that refer to your code -- this will often be a great first step in narrowing down a problem.

4. We're on a spaceship! Use debuggers.
   - Most languages have ways to be able to explore the current environment while your code is running in a particular state.

5. Tests can be useful tools during and after the debugging process.
   - Be like the whale -- use tests to probe your surroundings and assert your assumptions.
   - DON'T be like the petunias -- write tests to prevent regressions and saying "oh no, not again".

6. Write your own Hitchhiker's Guide. Take notes for your fellow debuggers or for future you!
   - These can be in the form of a text file, a wiki page, an email, or whatever works for you.
   - Sometimes the act of writing up notes can help you figure out the problem whether someone else ever sees your notes or not.

Further Reading

- Debug It! by Paul Butcher (is.gd/debug_it)
- Nelson Elhage's Made of Bugs blog post about keeping a lab notebook (is.gd/lab_notebook)
- Douglas Adams' Hitchhiker's Guide to the Galaxy (wherever fine books are sold)

Do you have questions or feedback? Did you find these techniques useful? We'd love to hear from you!

- @carols10cents and @JakeGoulding