THREE THINGS TO LEARN THE HARD WAY

@thejessleigh | #devtogether

My name is Jess Leigh Unrein, and my pronouns are ve/ver/vis. Right now I work as a Data Engineer with ActiveCampaign, but this month is special to me because I started Dev Bootcamp almost exactly five years ago. It feels both like it was yesterday and like it was a lifetime ago. I don't know how many of you here are bootcamps students, but bootcamp programs involve cramming an overwhelming amount of information in your head over a very short time, and you don't even get a certificate or a credential when you walk out. All you have is whatever you managed to retain over 9 or 12 or 16 grueling weeks, and an urgent need to find a job. It feels impossible when you graduate from bootcamp that you only barely have enough knowledge to get started. You just learned so much! I imagine that it's similar for college graduates. Getting your first job in any field is nerve wracking and disorienting because you have no idea just how much you don't know.

SET YOURSELF UP FOR ✨ SUCCESS ✨ (BY MAKING YOUR LIFE HARDER)

@thejessleigh | #devtogether

One of the first things you'll learn when you start your new job is that you'll be working on very few problems that haven't been solved before, and there's a tool for *almost* anything you could ever want.

It's tempting to get bogged down in learning a new suite of tools and chasing the new cutting edge technology that will supposedly solve your problems. Please don't do that. Even if you've used some of these fancy tools in the past at a bootcamp or in school, you should try to go without them at your first dev job. Every time you start a new job you're going to have to learn a different set of tools, processes, philosophies, and best practices. You're setting yourself up for future success if you don't get too attached to any particular toolchain and instead focus on the fundamentals.

COMMAND LINE GIT

@thejessleigh | #devtogether

Some places use Mercurial, but it's likely that during your time as a developer you will use git as the primary version control software for your code. Github does have a GUI. It's perfectly possible that the Github GUI is brilliant, but in five years I've only met a handful of people who use it. If you ask your coworkers to help you, and if you plan on pairing, the git GUI will bewilder them, and they will not be able to help you.

Besides, understanding how git works and why you're doing what you're doing will absolutely prove to be a valuable skill in the workplace. It's a common joke on the internet that you only need to memorize a few git commands, and that will carry you through. It's true that you *can* live life this way, and many people do. But if you're the one person on your team who can help debug a botched rebase, or who knows off the top of your head how to specifically select pick useful commits from a stale development branch, you'll be a valuable resource for your team going forward. Your peers will be in awe of you. They might even call you a wizard.

Side note, stop calling people wizards. Tech isn't magic. Anything that anyone on your team does is learnable and achievable. Terms like wizard or the portmanteau "automagically" tend to elevate certain people or processes in a way that makes them feel unapproachable and unobtainable. It perpetuates this useless myth of "talent" that obscures the hard work it takes to actually get good at something, and scares people off from achieving. Soap box moment over.

DITCH YOUR ORM
WHEN DEVELOPING LOCALLY

@thejessleigh | #devtogether

Object relational mappers are a neat bit of business that take information from a database table and turn that data into an Object that you can work with programmatically in your language of choice. Each language has at least a couple different ORMs, and they are incredibly useful for when you need to take information out of a database, keep track of that state, transform it in some way, and then save those changes as a new state. My preferred ORM is Python's SQLAlchemy. It has cool history functions that can tell you what has happened to the object within a particular database session.

But unless you're using some fancy features like keeping track of an object's history within a session, or testing out a database trigger, don't use your ORM in local development for the first couple months of working on a new project. Try to do everything in raw SQL first, and then rewrite things using your ORM before putting up a pull request for your team to review. I know there are probably a lot of people who disagree with me on this, but I think this accomplishes a few things.

1. **Stretch your SQL muscles**
2. **Get comfortable in your db shell**
3. **Get familiar with your data**

@thejessleigh | #devtogether

This is going to slow you down at first, especially if you're used to working with an ORM. But working these skillsets is going to help you debug queries in the future, and get you more acquainted with your codebase.

It's really embarrassing for me to admit now, but at my first dev job I was *terrified* of actually jumping into my database shell and running queries and mucking around. I still have a tiny residual fear of going into production databases, even if I know I'm only going to perform read only operations. My SQL has always been pretty okay. I was, for some reason, comfortable writing SQL queries and executing them with Python, but accessing the database directly felt dangerous. And there were a ton of MySQL or Postgres specific commands that I should have learned earlier. I really wish I'd bitten the bullet and played around with the database earlier.I would have wasted a lot less time writing weird Python scripts to avoid interacting with the db directly.

RUN EARLY. RUN OFTEN. CHERISH YOUR ERROR MESSAGES.

(JUST MAKE SURE YOU'RE NOT CONNECTED TO THE PROD DATABASE WHEN YOU DO SO 😳)

@thejessleigh | #devtogether

When I was in bootcamp one of the most common problems I saw people run into was the fear of running code and having it error out. I think, especially for people who change careers and are used to other workflows, we're not trained to expect and accept error messages. So running your code and seeing it scream at you - sometimes in big red text on your black and green terminal - that something errored out can feel like an indictment of you or your work.

Really, it's just a normal part of the development process. Run your code when you know it will fail. Put a bunch of print statements in. If you're feeling fancy, use a debugger like `pry` in Ruby or `ipdb` in Python. It took me a while to retrain my brain into understanding that when code gives you an error, you haven't failed. It's your code providing you with valuable information for improving it.

I mean, ideally you don't want a bunch of errors happening in production, and you shouldn't just be running test scripts willy nilly if it touches a production database. But if you're developing locally or in a sandbox environment, take advantage of that sandbox and play around. You're not going to break your computer and you're not going to wreck the code irrecoverably. If you learn the hard way and intentionally make mistakes with the intention of learning, you're well on your way to a long and successful career in tech.

THANK YOU!

@thejessleigh | #devtogether