When things go wrong (debugging and profiling)
Announcements

- AI Quiz on Autolab
  - Due Wednesday Night

- PA 0 (setup Git) on Autolab
  - Due 1 week from today
  - See Piazza for common problems

- PA 1 (Parsing CSV files in Scala) on Autolab
  - Due 2 weeks from today
  - Submissions open a week from today (or maybe sooner, up to you...)
  - Start Early!
Github + Autolab

- PA 0 allows us to connect your Autolab and Github accounts
- Help us open PA 1 early!
  - PA 1 will open once %90 of you (from both A and B) submit PA 0
  - Once PA 1 opens, allow 24 hours after submitting PA 0 to submit PA 1

https://xkcd.com/2565/
Notes on Submissions

- **Github Classroom**
  - You will get an invite link for each individual project
  - Following the link will create a GitHub Git repository for the project with template code

- **Edit the repository code according to project specific instructions**
  - Make sure to commit and push frequently
  - Create a new submission in Autolab when ready

- **Requirements**
  - Make sure you are using Scala 2.13.x
  - Don’t add any outside packages
Being a good computer scientist does not mean getting things 100% right all of the time. Things WILL go wrong.

A good computer scientist knows how to solve problems, and how to recover when things go wrong.
Things WILL go wrong...often

Being a good computer scientist does not mean getting things 100% right all of the time. Things WILL go wrong.

A good computer scientist knows how to solve problems, and how to recover when things go wrong.

Let's talk about some useful tools for recovering...
The REPL (read - eval - print loop)

- **From IntelliJ:** Ctrl+Shift+D
  - Highlight a line and press Ctrl+Shift+X to execute
  - Copy+paste a line and press Ctrl+Enter to execute

- **From the command line:** scala
  - Paste or type commands to run them
  - Type :help to get a list of additional commands

- **From SBT:** console
Unit Testing

- Break the big problem into smaller problems
  - Test each small solution before combining them

- Useful for debugging
  - Sanity check each step in a large process to make sure it works
  - Separate the UI from the tests

- Useful way to encode your assumptions, constraints, etc
  - Automatic reminder if your assumptions change
  - Also acts as self-documentation
Unit Testing

- Break the big problem into smaller problems
  - Test individual solutions before combining them

  If you’re building a boat, you aren’t going to build the entire thing then just throw it in the water and hope it floats...you would test throughout the whole process.

- Useful way to encode your assumptions, constraints, etc
  - Automatic reminder if your assumptions change
  - Also acts as self-documentation

The same logic applies to your coding projects!
Basic Debugging
class HelloWorldTest extends AnyFlatSpec {
   "HelloWorld.doThings()" should "return 5" in {
      assert(HelloWorld.doThings() == 5)
   }
   it should "not return 10" in {
      assert(HelloWorld.doThings() != 10)
   }
   "HelloWorld.x" should "have type Float" in {
      assert(HelloWorld.x.isInstanceOf[Float])
   }
   "Register(0).addToValue" should "return the input value" in {
      val reg = Register(0)
      for (i <- 1 to 10000) { assert(reg.addToValue(i) == i) }
   }
}
class HelloWorldTest extends AnyFlatSpec {
  "HelloWorld.doThings()" should "return 5" in {
    assert(HelloWorld.doThings() == 5)
  }
  it should "not return 10" in {
    assert(HelloWorld.doThings() != 10)
  }
  "HelloWorld.x" should "have type Float" in {
    assert(HelloWorld.x.isInstanceOf[Float])
  }
  "Register(0).addToValue" should "return the input value" in {
    val reg = Register(0)
    for (i <- 1 to 10000) { assert(reg.addToValue(i) == i) }
  }
}
Live Demo

ScalaTest
Profiling

● IntelliJ -> Profilers

● SBT -> HProf
  ○ https://docs.oracle.com/javase/8/docs/technotes/samples/hprof.html
Profiling

- IntelliJ -> Profilers

- SBT -> HProf
  - [https://docs.oracle.com/javase/8/docs/technotes/samples/hprof.html](https://docs.oracle.com/javase/8/docs/technotes/samples/hprof.html)

```scala
fork in run := true
javaOptions in run += "-agentlib:hprof=cpu=samples,depth=10"
```
Profiling

- IntelliJ -> Profilers

- SBT -> HProf
  - [https://docs.oracle.com/javase/8/docs/technotes/samples/hprof.html](https://docs.oracle.com/javase/8/docs/technotes/samples/hprof.html)

```scala
fork in run := true
javaOptions in run += "-agentlib:hprof=cpu=samples,depth=10"
```

Load HProf  
Sample CPU Usage  
Stack Trace Depth
Live Demo

Profiling with HProf
TRACE 300207:

scala.collection.StrictOptimizedLinearSeqOps.drop(LinearSeq.scala:261)
scala.collection.StrictOptimizedLinearSeqOps.drop$(LinearSeq.scala:257)
scala.collection.immutable.List.drop(List.scala:79)
scala.collection.immutable.List.drop(List.scala:79)

...
## HProf Traces

JAVA PROFILE 1.0.1, created Fri Sep 3 02:24:46 2021

Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.

Redistribution and use in source and binary forms, with or without

... 

### TRACE 300207

- `scala.collection.StrictOptimizedLinearSeqOps.drop(LinearSeq.scala:261)`
- `scala.collection.StrictOptimizedLinearSeqOps.drop$(LinearSeq.scala:257)`
- `scala.collection.immutable.List.drop(List.scala:79)`
- `scala.collection.immutable.List.drop(List.scala:79)`

...

### CPU SAMPLES BEGIN (total = 185) Fri Sep 3 02:24:48 2021

<table>
<thead>
<tr>
<th>rank</th>
<th>self</th>
<th>accum</th>
<th>count</th>
<th>trace</th>
<th>method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44.86%</td>
<td>44.86%</td>
<td>83</td>
<td>300207</td>
<td><code>scala.collection.StrictOptimizedLinearSeqOps.drop</code></td>
</tr>
<tr>
<td>2</td>
<td>35.14%</td>
<td>80.00%</td>
<td>65</td>
<td>300210</td>
<td><code>scala.collection.immutable.$colon$colon.tail</code></td>
</tr>
<tr>
<td>3</td>
<td>5.95%</td>
<td>85.95%</td>
<td>11</td>
<td>300071</td>
<td><code>java.lang.ClassLoader.defineClass1</code></td>
</tr>
<tr>
<td>4</td>
<td>2.16%</td>
<td>88.11%</td>
<td>4</td>
<td>300209</td>
<td><code>scala.collection.immutable.Range.foreach$mVc$sp</code></td>
</tr>
</tbody>
</table>