@ConradIrwin

Read

Eval

Print

Loop

ONE LINE OF RUBY!

```ruby
def repl
  loop{ puts eval $stdin.readline }
end
```

12,000 LINES OF RUBY!

```
require 'pry'
binding.pry
```

REPL-DRIVEN DEVELOPMENT

- Don't write code first.
- Don't write tests first.
- Use the REPL first.

LOGICAL AWESOME

My Codes Are Perfect

P(BUG)

- When you write code: ~0.8
- Github production: ~0.000000008*

* Github exception-% this week is 0.0008%,
assume 1000 method-calls per request.

100,000,000 TIMES BETTER?

- Run the code → 1000 times better?
- Read the code → 100 times better?
- Understand the code → 1000 times?

100,000,000 TIMES BETTER?

- Run the code → takes seconds?
- Read the code → takes minutes?
- Understand the code → takes days?

PRY

- Easy to run the code!
- Easy to read the code!
- Helps to understand the code...*

* Though I usually git clone if I need this.

DEMO

RECAP

- `ls` shows methods.
- `$` (`show-source`) shows code.
- `?` (`show-doc`) shows documentation.

DOWNSIDES OF REPL

- Doesn't keep a record.
- You have to re-test lots.
- Hard to test large apps.

SOLUTION: TESTS

1. Find solution in REPL.
2. Record findings in tests.
3. Write code that works first time :).

TESTS HELP

- When code-base is large.
- With API design.
- Document assumptions.

DOWNSIDES OF TESTS?

- Slow to write.
- Slow to run.
- Go out of date.

PRY CAN HELP!

```
gem install pry-plus
```

```
require 'pry-rescue/minitest'
require 'pry-rescue/rspec'
```

DEMO

RECAP

- try-again
- break
- step / next
- play
- edit

EVEN WITH TESTING

- Bugs slip through.
- Become harder to catch.
- Require longer to debug.

bugsnag

- Catches exceptions in production.
- Captures data to help you debug.
- Notifies you intelligently.

FIXING BUGS

- Reproduce locally.
- Find problem.
- Fix it.

DEMO

RECAP

- binding.pry
- cd
- up / down
- whereami
- help

USE A REPL

- Saves time
- Improves quality
- More fun!

USE PRY!

- Featureful
- Robust
- Friendly

@ConradIrwin