goto;

GOTO
CHICAGO 2023

#GOTOchgo

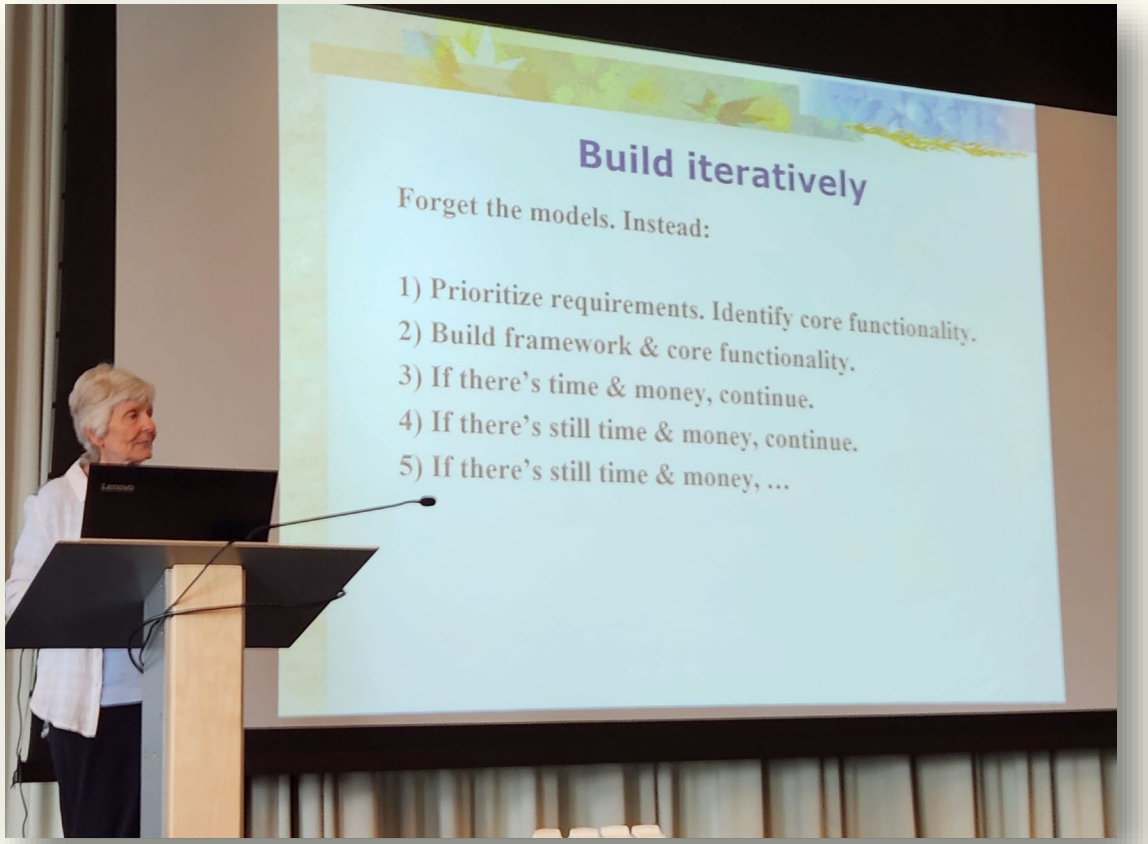
REPLs All The Way Up

Avdi Grimm

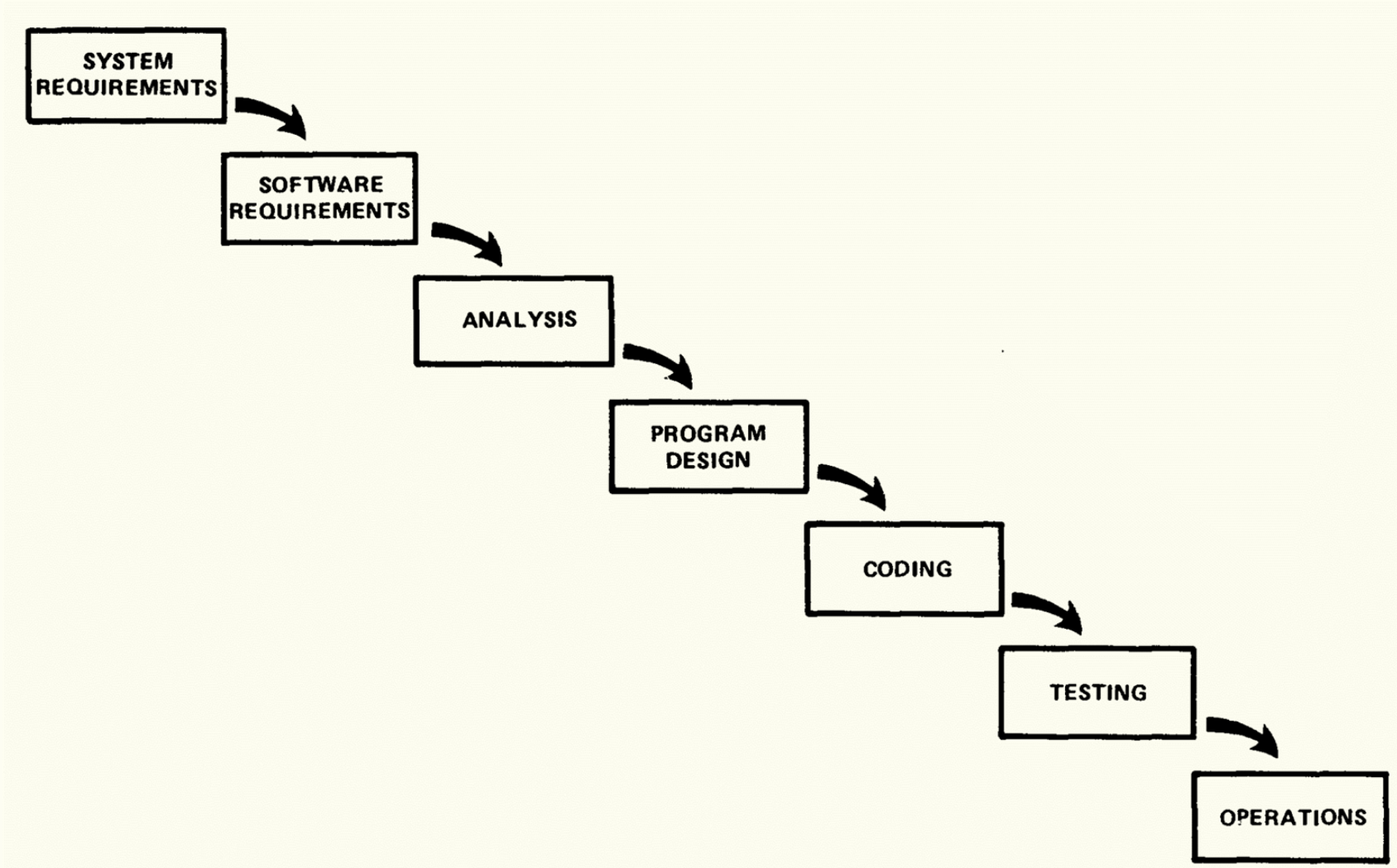
https://avdi.codes

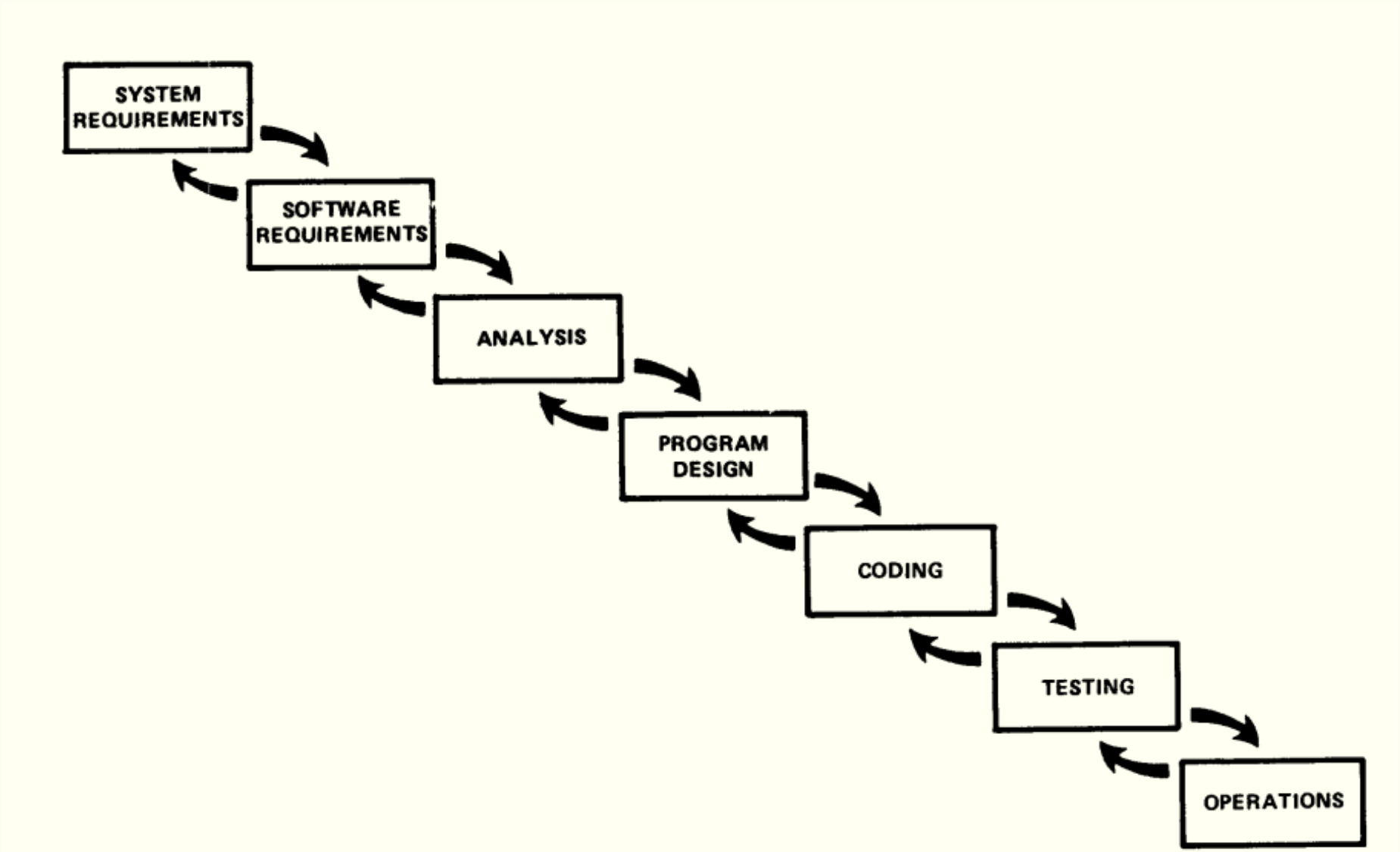


Feedback Loops


Build iteratively
Forget the models. Instead:
1) Prioritize requirements. Identify core functionality.
2) Build framework & core functionality.
3) If there's time & money, continue.
4) If there's still time & money, continue.
5) If there's still time & money, ...


Agile Software Development
What
Find out where you are
Take a small step towards your goal
Adjust your understanding based on what you learned
Repeat


SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS

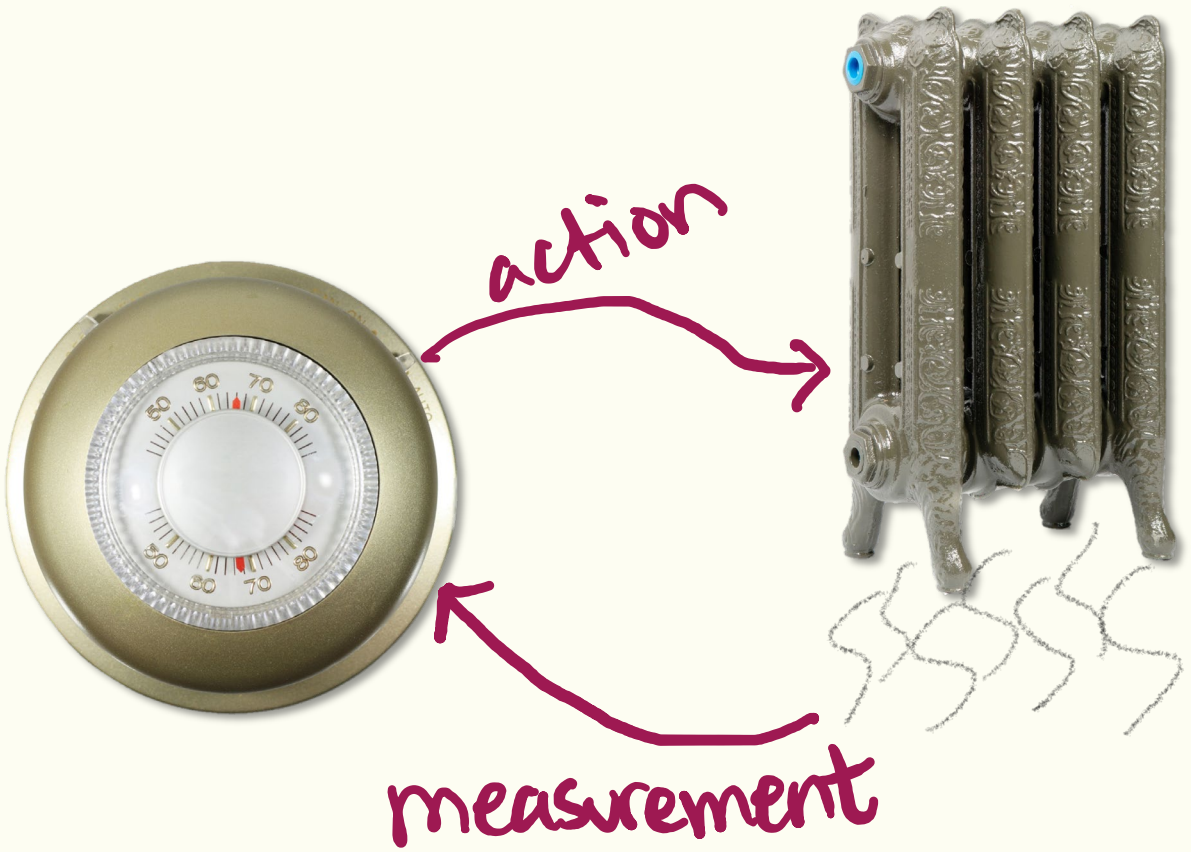

SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS



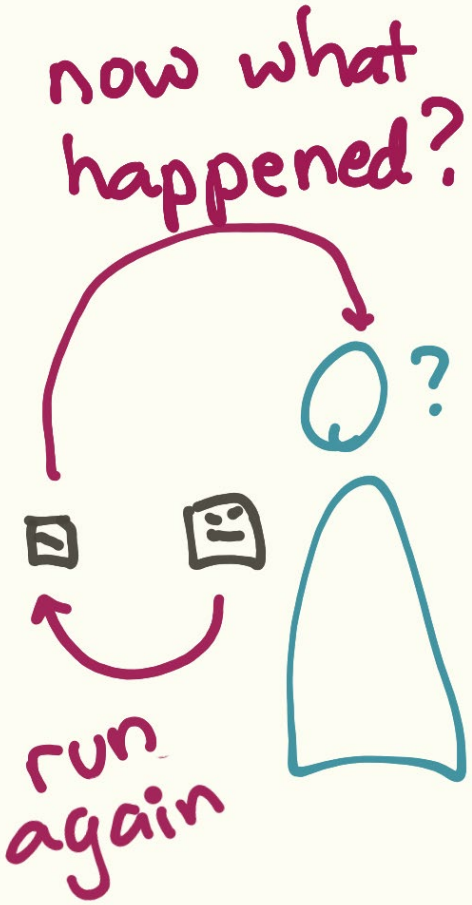
Feedback Loops

We like them!

Control Loop



Exploratory Loop



Not all exploratory loops are created equal



What do good exploratory loops look like?


Develop Your Subconscious
Experience
• do stuff
• observe results
BOXED WATER IS BETTER.
BOXED WATER IS BETTER



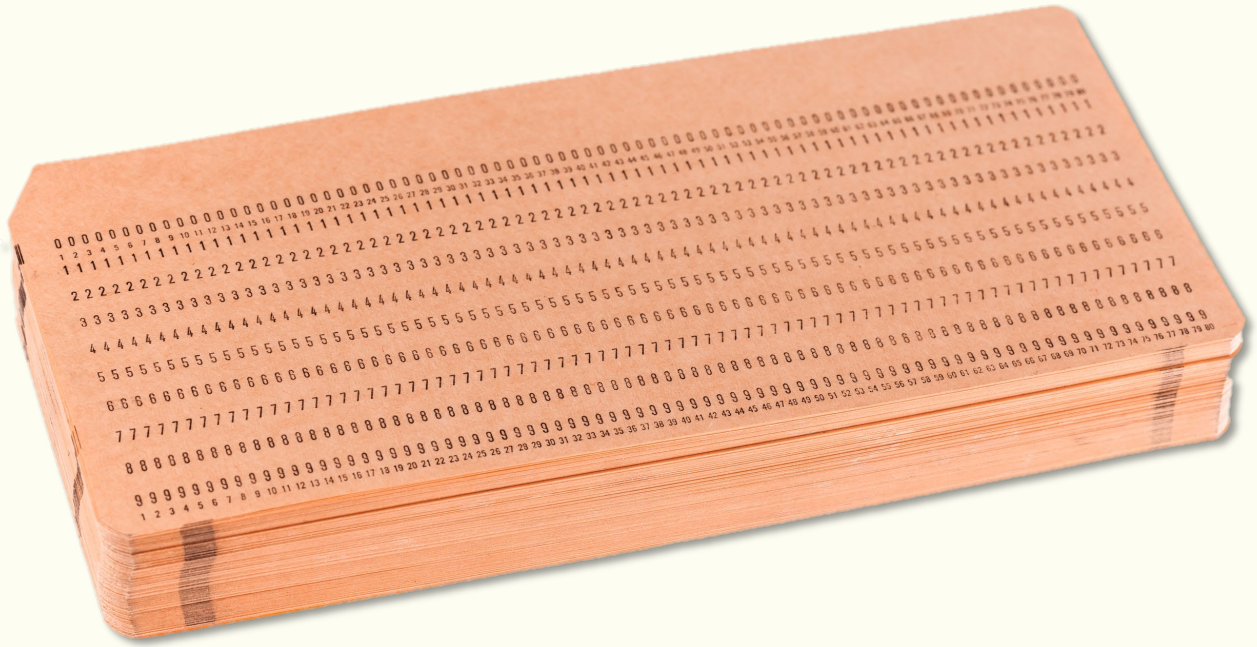
What do good exploratory loops FEEL like?

Roadmap

1. A brief history of exploratory feedback loops
2. Properties of better feedback loops
3. Examples, from small to large



Once upon a time...




MEN
22
MISSOURI PACIFIC TRAINS
SOUTH - SOUTHWEST

1964: LISP on the PDP-1



As soon as the basic PDP-1 LISP system is read into the computer, control stops at register 4. Turn up sense switch 5 for typewriter input; press CONTINUE; and the system enters a waiting loop which causes lamps to light in the program counter, looking like 1335. At this point, the LISP system is ready for manual typewriter input. As soon as the operator types, for example:

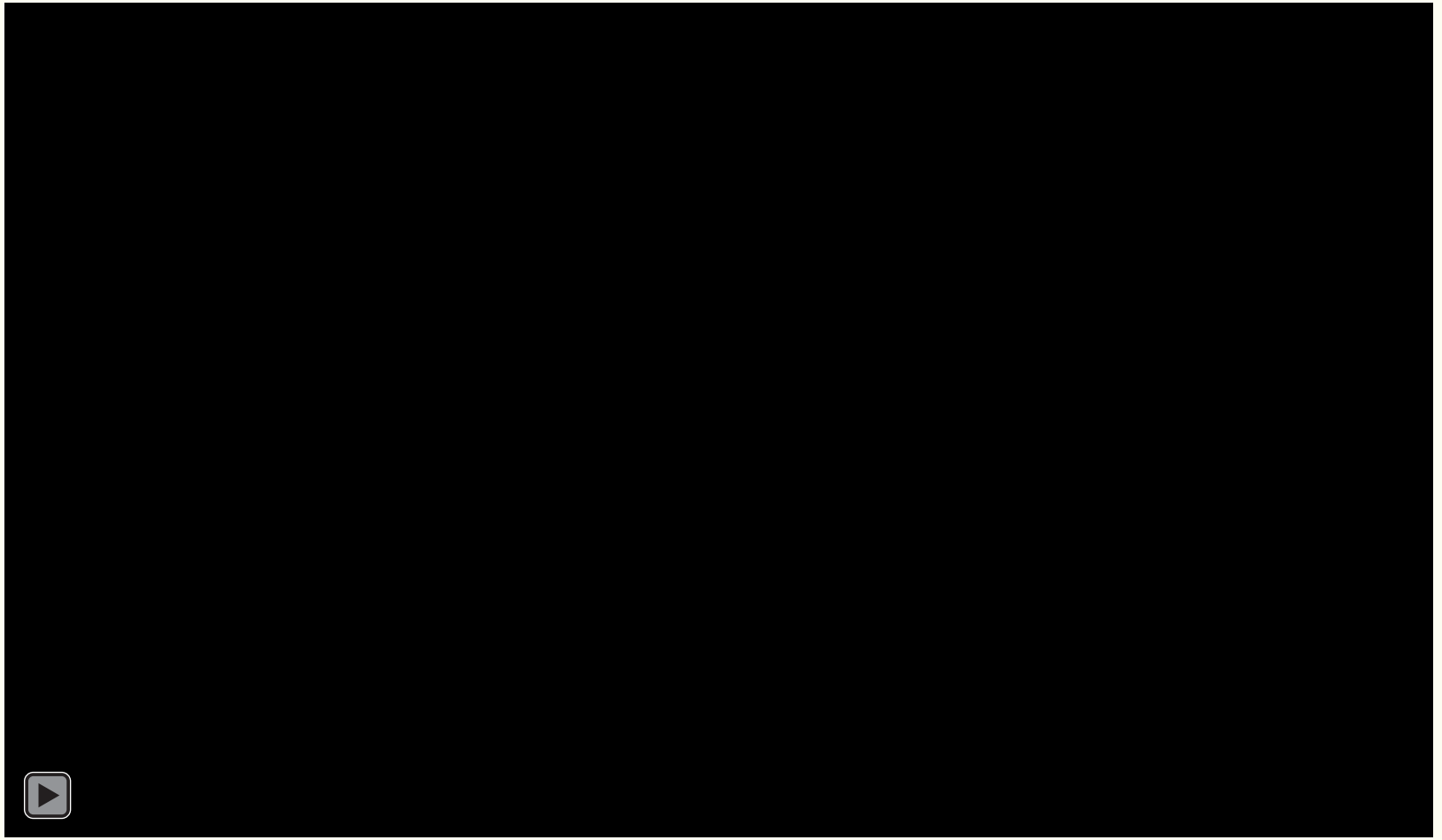
```
(CAR (QUOTE (A B C D)))
```

together with a final space at the end of the last right parenthesis, the computer takes control of the typewriter, impulses a carriage return, and then types out:

```
A
```

which of course is the correct answer. Similarly, for the other suggested test sequences in Table 2 below.

Read, Eval, Print, Loop

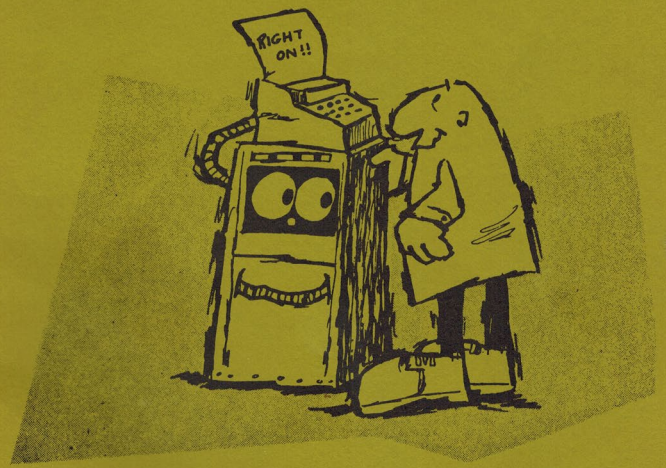


Late 1960s - Early 1970s: BASIC and the DTSS



MY COMPUTER LIKES ME *

by BOB ALBRECHT



*when i speak in BASIC

Dr. Dobb's Journal of

COMPUTER Calisthenics & Orthodontia

Running Light Without Overbyte

Volume One People's Computer Company, Box E, Menlo Park, CA 94025 1976

A Reference Journal
for Users of
Home Computers

Volume One

People's Computer Company

```
SCR

10 PRINT "MY HUMAN UNDERSTANDS ME"
20 END
```

then type

```
RUN
```

and the computer will type

```
MY HUMAN UNDERSTANDS ME
```

```
SCR

10 PRINT "MY HUMAN UNDERSTANDS ME"
20 END
```

then type

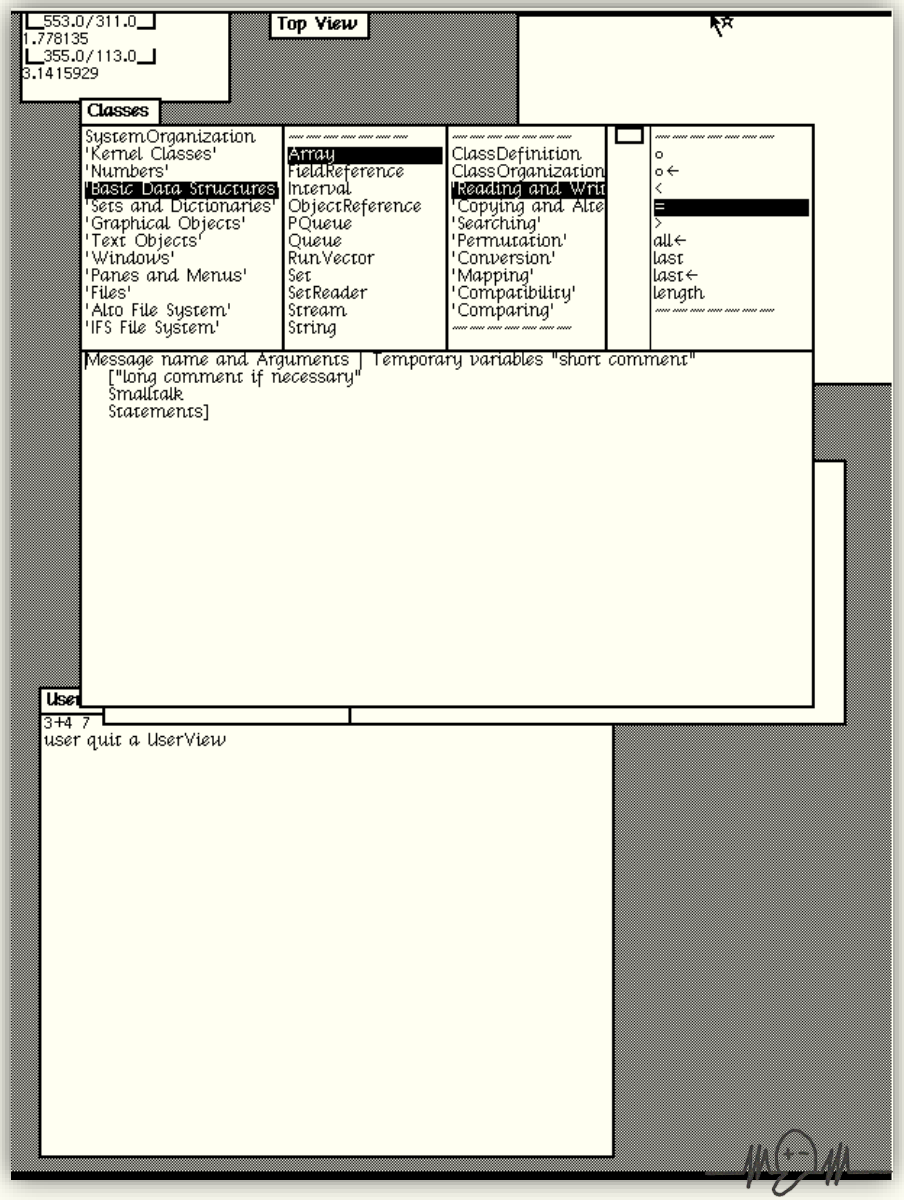
```
RUN
```

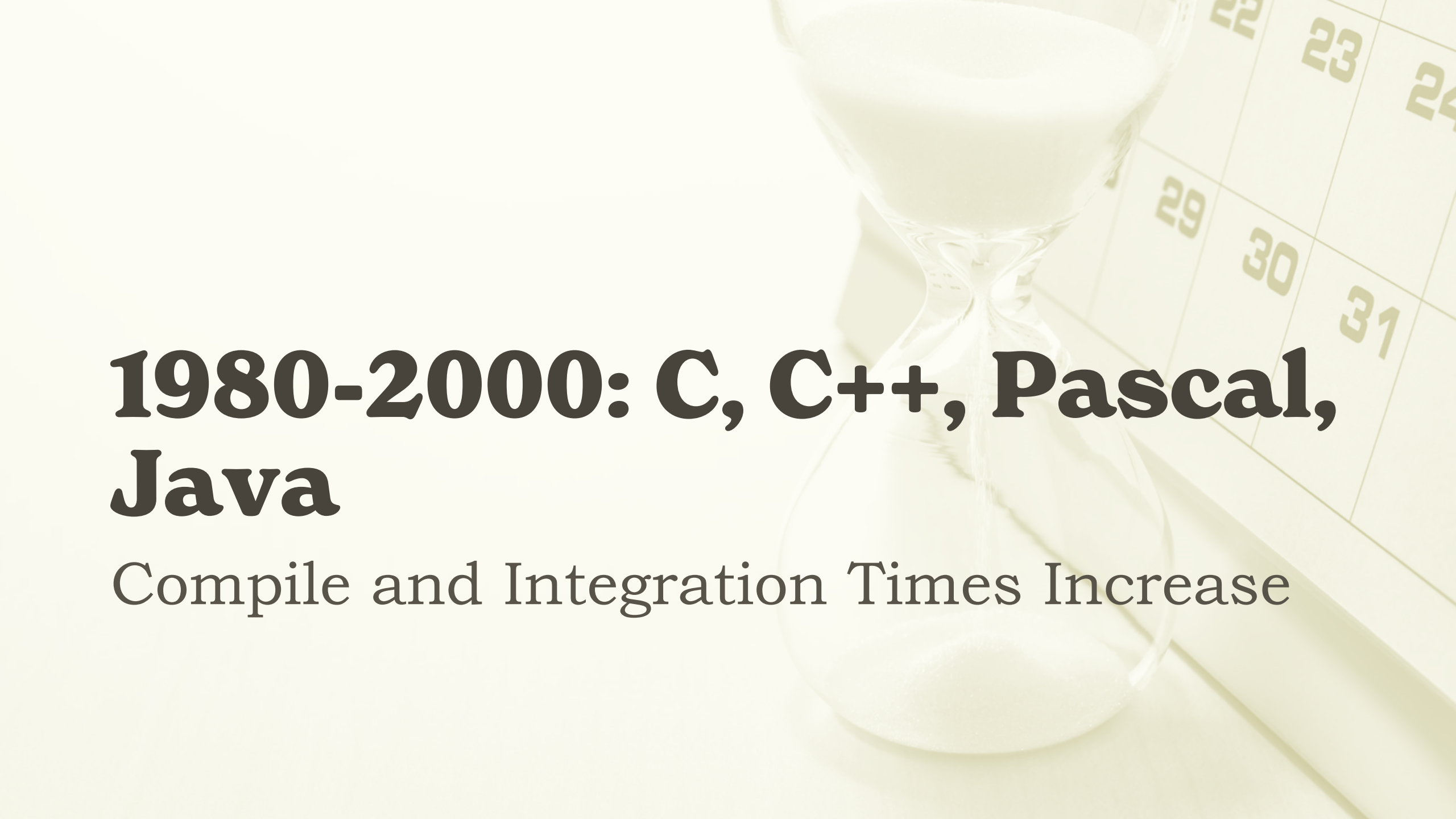
and the computer will type

```
MY HUMAN UNDERSTANDS ME
```



1970s: Smalltalk





1980-2000: C, C++, Pascal, Java

Compile and Integration Times Increase

Debugger-Driven Develop-ment



~2000: Test-Driven Development!



2000s: "Scripting Languages"

TCL, Scheme, Perl, Python, Ruby, PHP, JavaScript, Groovy…



We keep trying to get back to this state of grace

- Because it's approachable
- Because it's satisfying
- Because it's productive
- Because it *feels good*

It feels good because it matches how humans evolved to think

*"human interaction with software is embedded in **a task-artifact cycle**"*

— John M. Carroll, 1992

Ideas are formed: *"when expressed [in] conversation with… representations, artifacts, or **objects-to-think with**"*

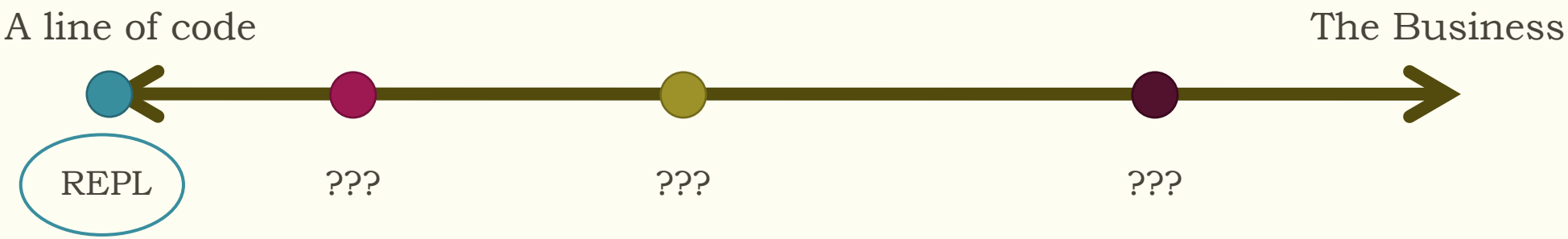
— Edith Ackermann (on Seymour Papert), 2001

"Knowledge fluidly grows through action taken in a complex system"

— Kit Martin (z"l), 2020

REPL: The gold standard of exploratory feedback in-the-small

Can we have this feeling at larger scopes?



REPL Nature

What properties make REPLs feel so good?

READ

There is a prompt, welcoming you

Invitational. Approachable.

What can you enter? Anything!

It's *Open-ended*

Most REPLs provide autocomplete or suggestions

Discoverable. Serendipitous.



You can always hit the up-arrow

No pausing for perfection. It’s highly *incremental.*

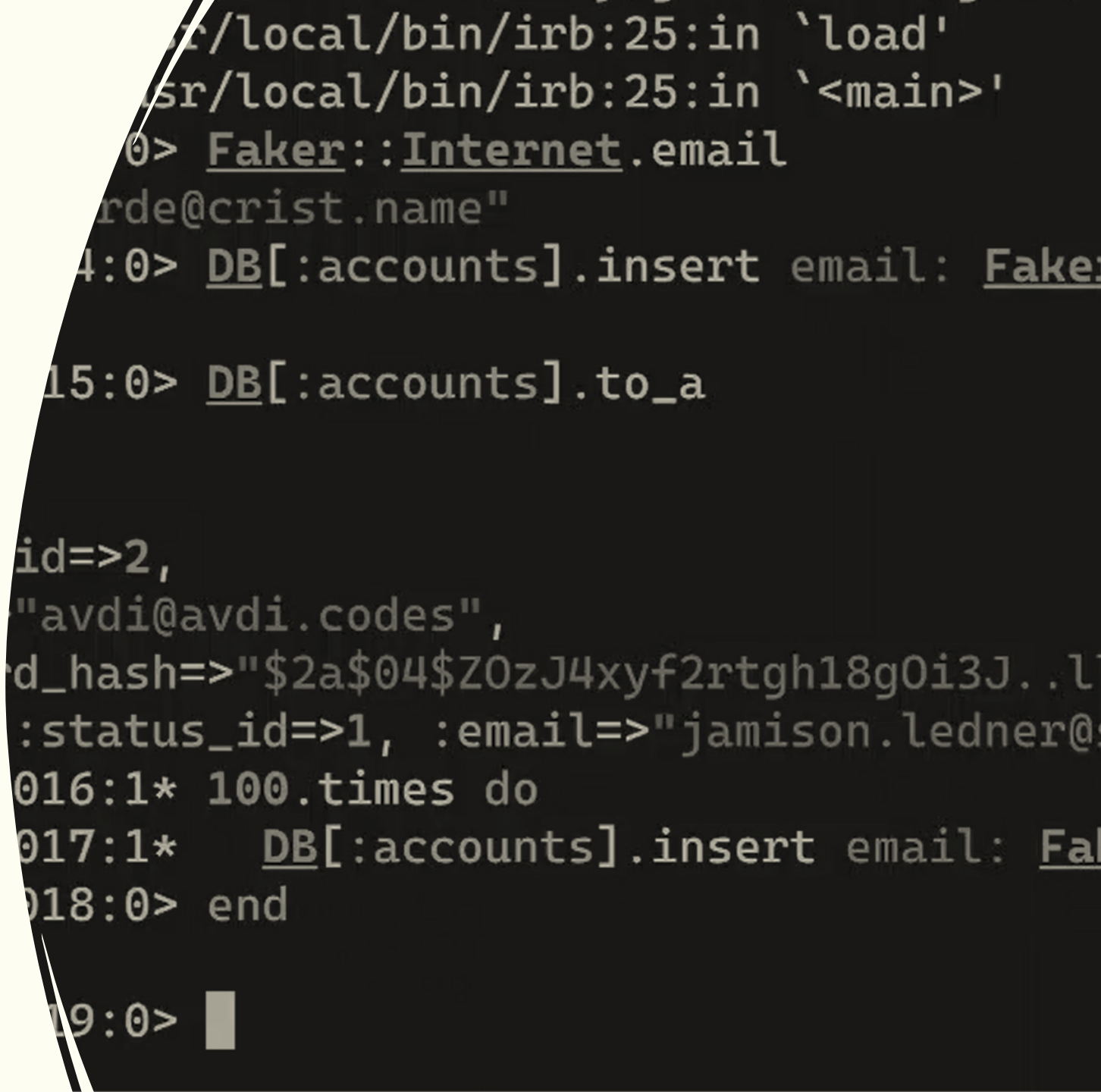
REPLs are SAFE

This is a private, intimate place to try silly things.

EVAL

Use the language you're used to

No further abstractions. REPLs are *familiar*, *concrete*, and the knowledge you build is directly *transferable*.

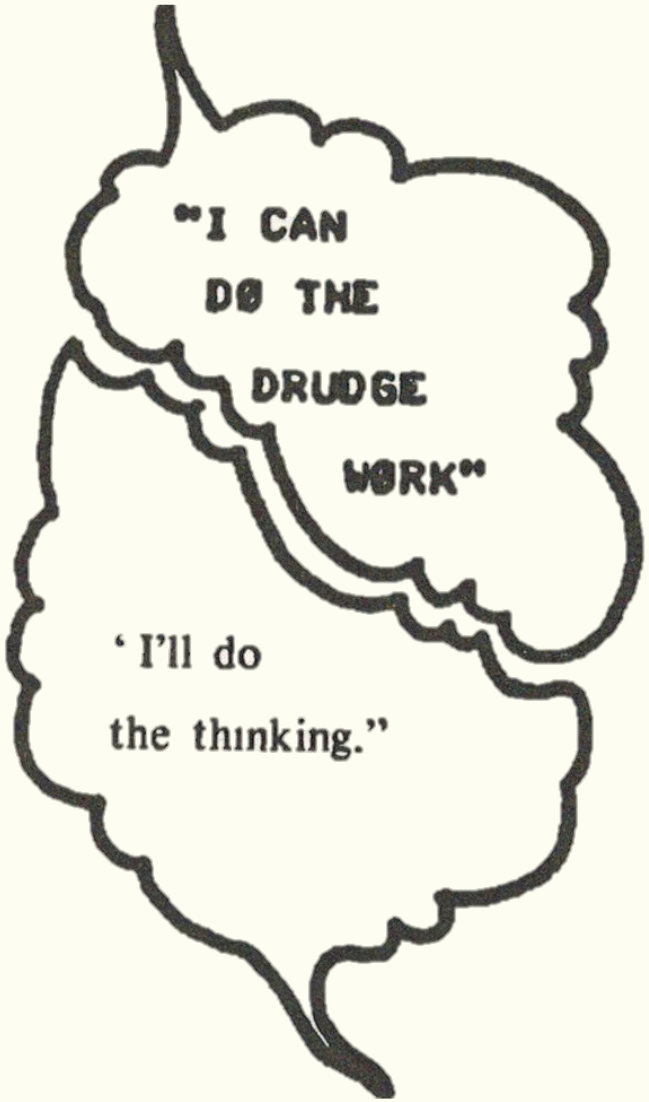




Poke the system with a stick

Don’t just ask questions, *perturb* the system and see what falls out.

REPLs are AUTOMANUAL

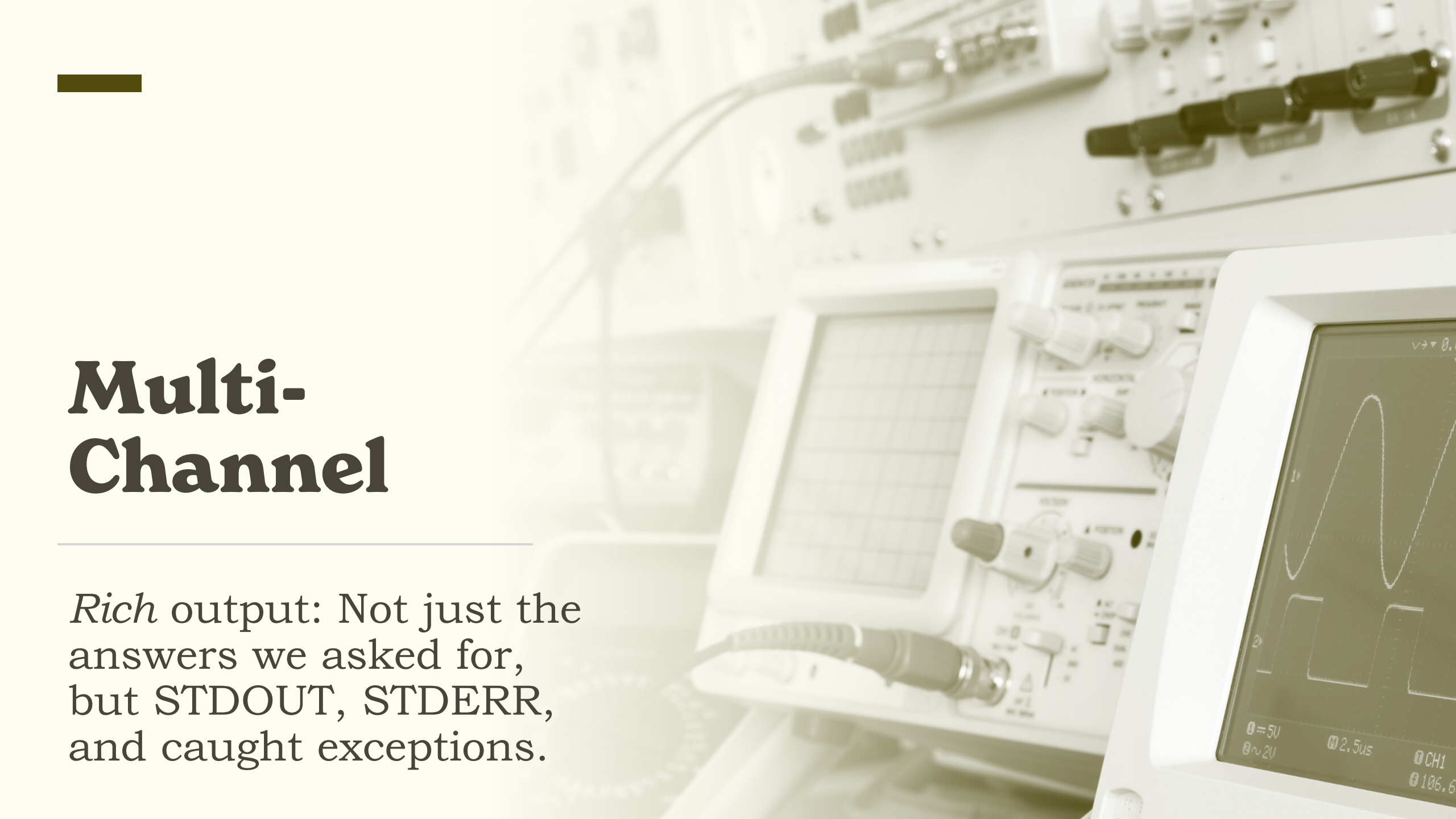


PRINT



Multi-Channel

Rich output: Not just the answers we asked for, but STDOUT, STDERR, and caught exceptions.



mistrakes

Do you occasionally make mistakes? We do, watch.

```
10 PTINT 2*3+4
```

We misspell PRINT.

```
SYNTAX ERRØR
```

The computer tells us we made a mistake.

The error message may be different on your computer. That's not the point. The point is, if we had noticed that we hit T when we meant to hit R, we could have corrected our mistake by using the back arrow.



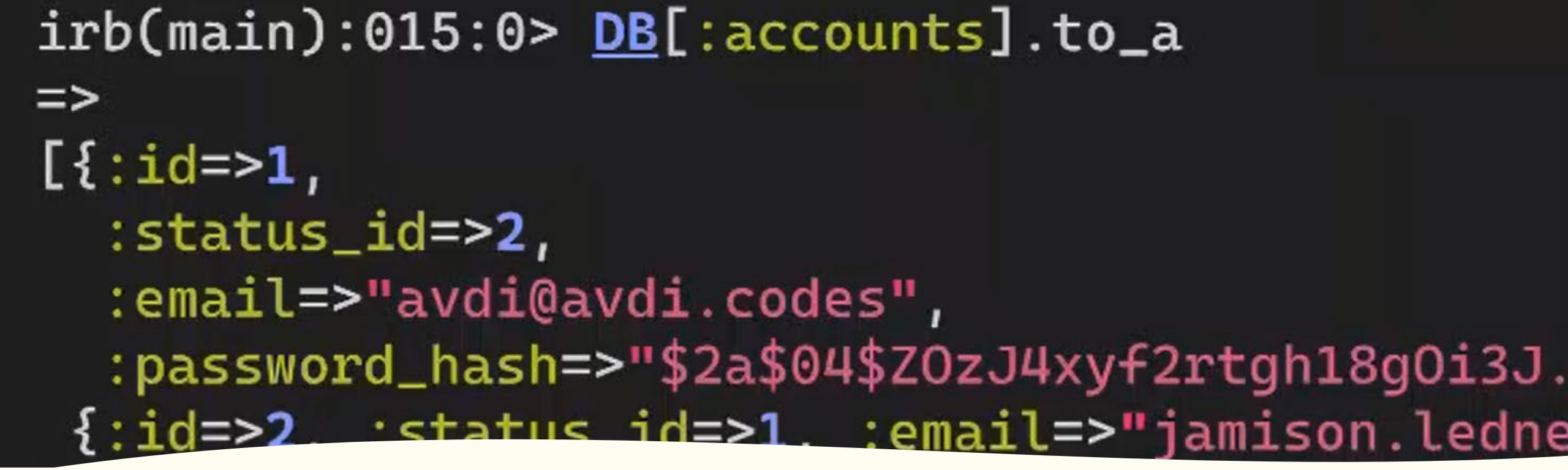
Rich Mistakes

```
irb(main):004:0> 1.up_to(10)
(irb):4:in `<main>': undefined method `up_to' for 1:Integer (NoMethodError)
Did you mean?  upto
        from /usr/lib/ruby/gems/3.0.0/gems/irb-1.3.5/exe/irb:11:in `<top (required)>'
        from /usr/bin/irb:23:in `load'
        from /usr/bin/irb:23:in `<main>'
irb(main):005:0>
```

Storytelling

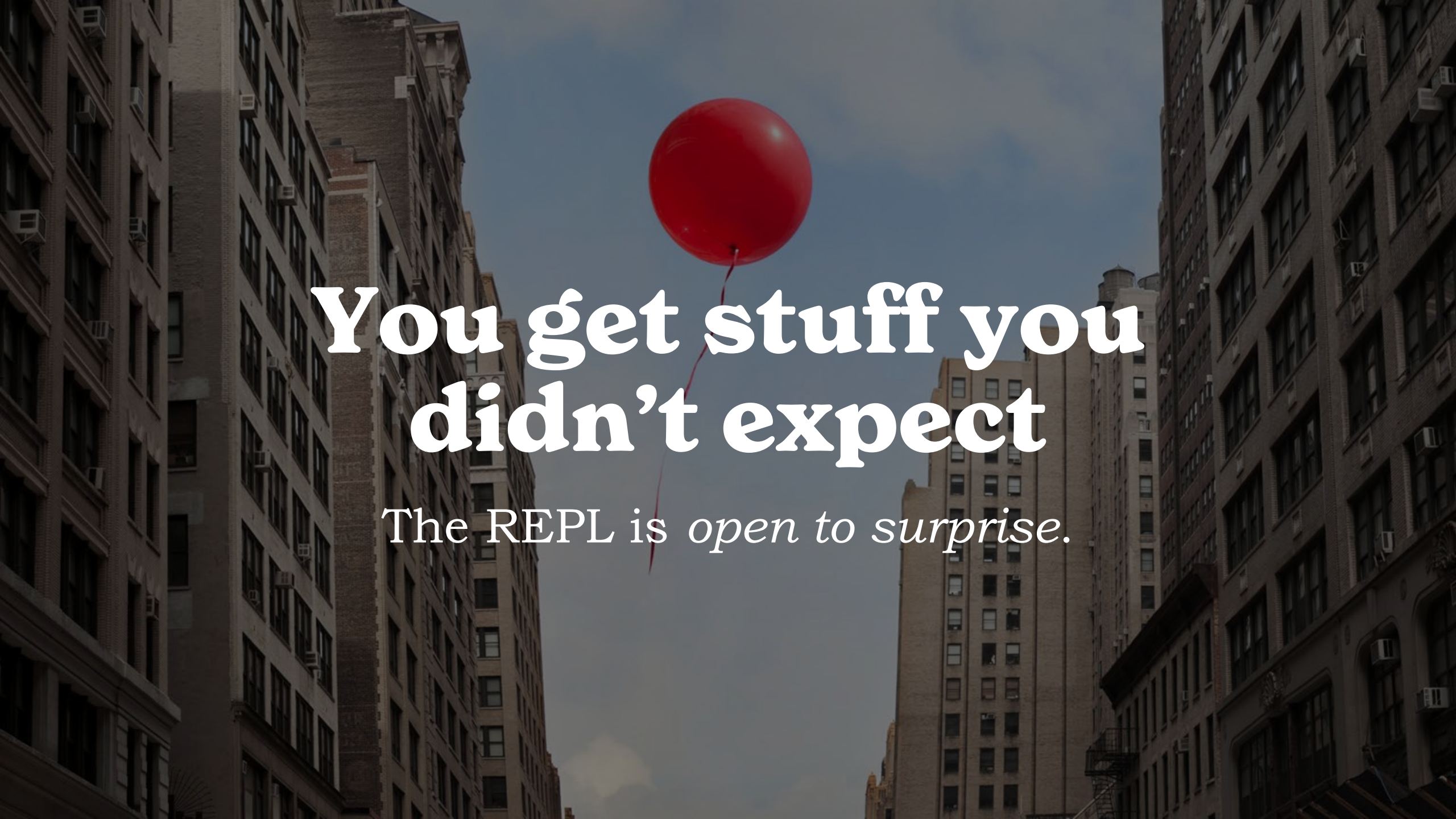
A story, but an ephemeral one. A narrative.





Serialized data structures

This is *exposed ductwork:* it gives you a clue about further questions you might ask.


You get stuff you didn't expect
The REPL is *open to surprise*.



REPLs are EXPLORAMENTAL!

LOOP

REPLs build up context over time

Variables, helpers, changes to the data.





REPL Interactions are Lossy

REPLs are fast

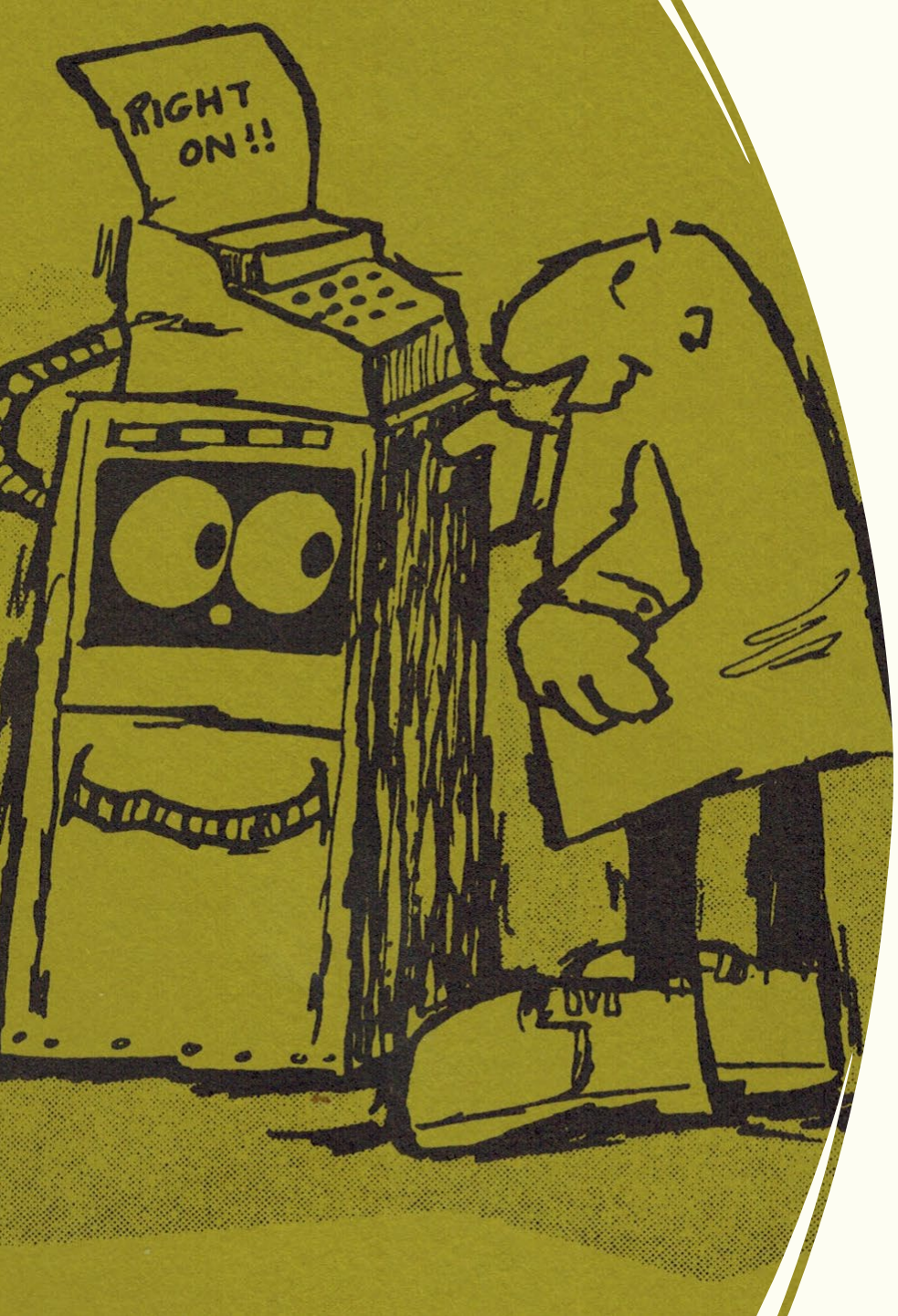
This is not “must go faster”




УБРАНЫ
УБРАНЫ
ВЫСОТА
АВИАГОРИЗОНТ
ПОДЪЕМ
СПУСК
УКАЗАТЕЛЬ ПОВОРОТА
ГЛАВНЫЙ ВЫКЛЮЧАТЕЛЬ
Raadio
off
НАИМЕНЬШИЙ ВЕС ПРИ ПОЛЕТЕ В ОДИНОЧКУ
68 кг

REPLs ~~are fast~~ move at the speed of your questions

REPLs are *responsive*: they provide a pacing of iteration appropriate to the scope.



REPLs are CONVERSATIONAL

REPL NATURE

READ

- Invitational
- Approachable
- Open-Ended
- Discoverable
- Incremental
- Private

Safe

EVAL

- Familiar
- Transferable
- Concrete
- Reflective
- Reactive

Automanual

PRINT

- Ductwork Exposed
- Gradually Refined
- Rich
- Narrative
- Open to Surprise

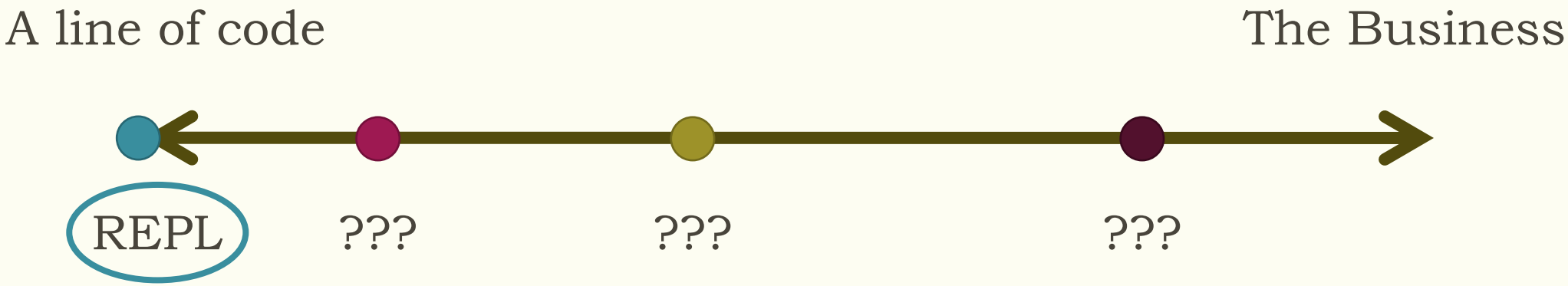
Exploramental

LOOP

- Contextual
- Ongoing
- Infinitely Deep
- Lossy
- Responsive

Conversational

Movin' on up...

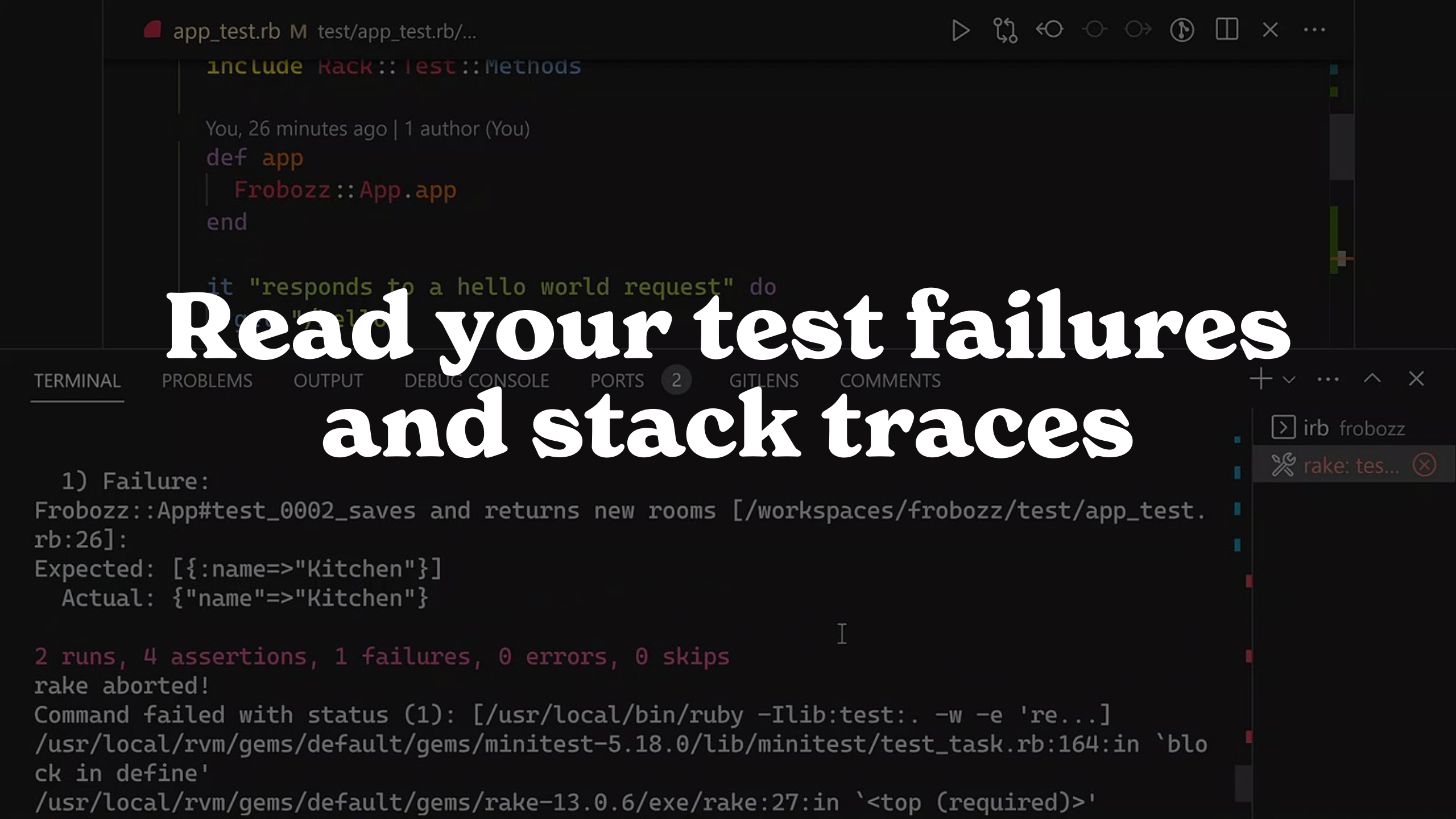



A line of code
The Business
REPL


A line of code
The Business
A Unit

Practice test-driven design

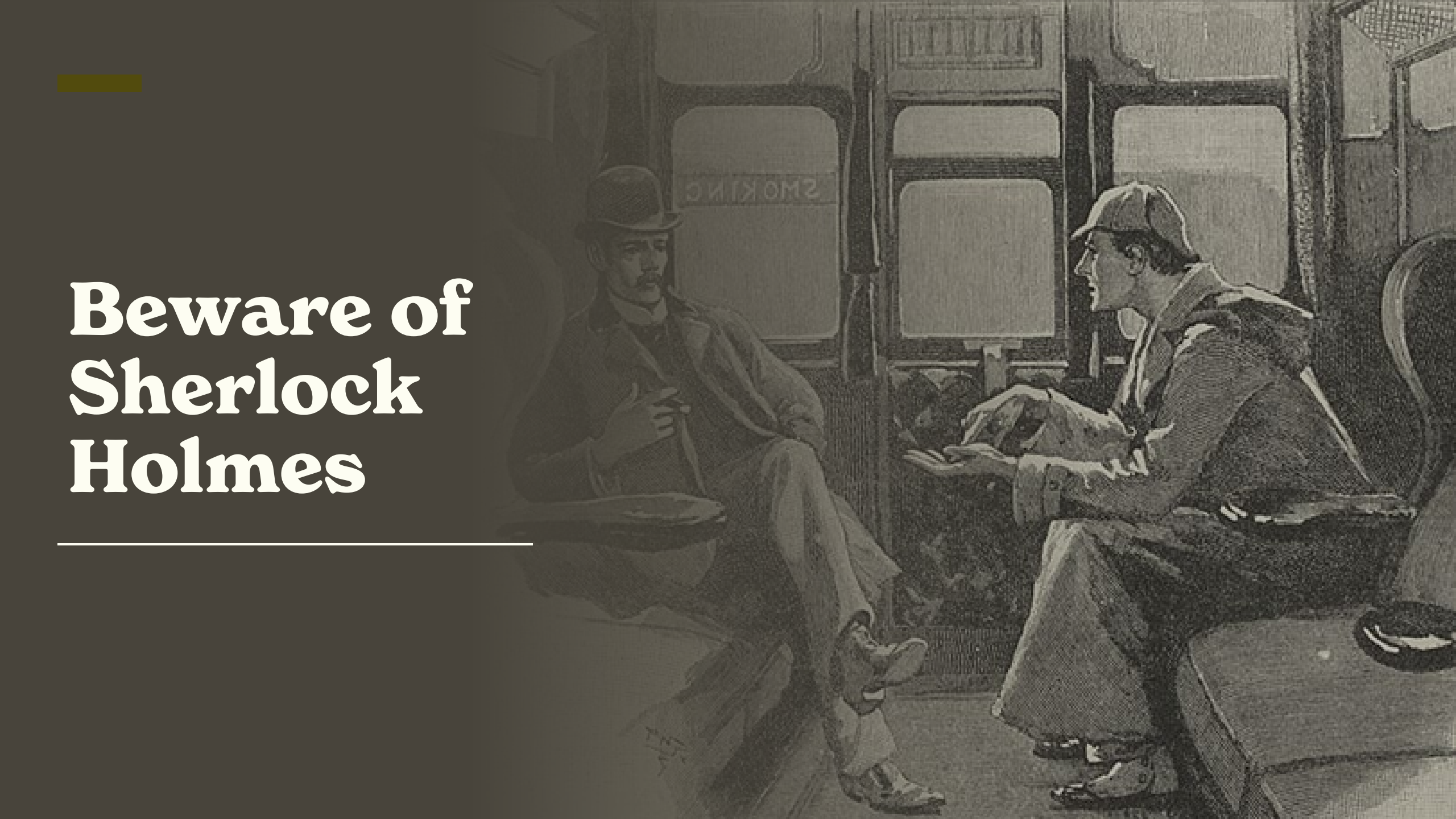




Read your test failures and stack traces


A line of code
The Business
A Feature

Beware of Sherlock Holmes



Tracing

Another view. Below is a **TRACE** of the program. *(Trace? Sure! A TRACE traces the path the computer takes through the program it is working on, and shows what values are assigned to the variables at any step in the program. Ain't it obvious?)* In the column marked N we show the value of N *after* the statement on the same line has been carried out by the computer.

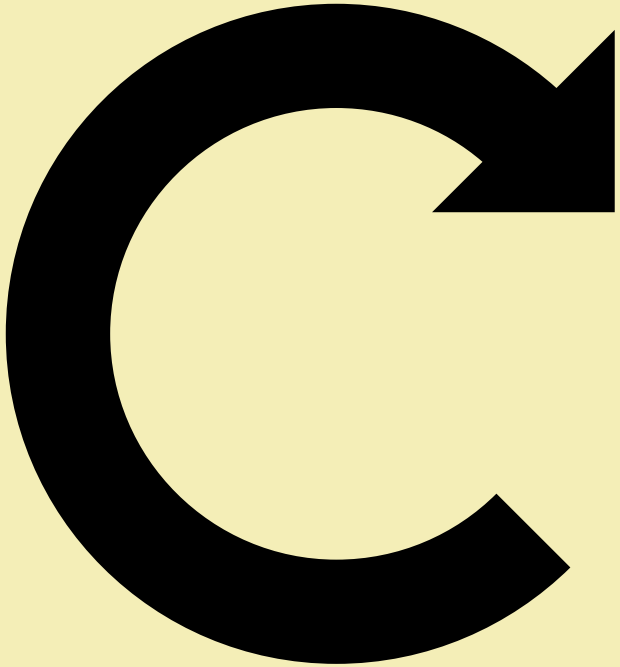
STATEMENT	N	REMARKS
`10 LET N=1`	1	
`20 PRINT N`	1	Print the value of N.
`30 LET N=N+1`	2	Increase N by 1 *(add 1 to the old value of N).*
`40 GØ TØ 20`	2	Go to beginning of loop.
`20 PRINT N`	2	Print the value of N.
`30 LET N=N+1`	3	Increase N by 1.
`40 GØ TØ 20`	3	Go to beginning of loop.
`20 PRINT N`	3	Print the value of N.
`30 LET N=N+1`	4	Increase N by 1.
`40 GØ TØ 20`	4	Go to beginning of loop.

```
root@9677044702e7:/workspaces/examples# strace -e open bundle 2>&1 | less
root@9677044702e7:/workspaces/examples# strace -f -e open bundle 2>&1 | less
root@9677044702e7:/workspaces/examples# strace -f -e openat bundle 2>&1 | less
```

Use the debugger (judiciously)



Reload-driven development



Leverage hot-reload




A line of code
The Business
A Service

gracefuldev.session.sql

Detach file from gracefuldev | Run on active connection | Select block

```
1  select * from wp_postmeta where meta_key like "wpf%";
2
3
```

gracefuldev: select * from wp... | gracefuldev: select * from wp... | gracefuldev: select * from wp...

meta_id	post_id	meta_key	meta_value
Filter...	Filter...	Filter...	Filter...
126997	28154	wpf_settings_buddypress	a:1:{s:10:"apply_tags";a:1:{i:0;s:26:"gracefuldev:grouptype:team";}}
127009	28155	wpf_settings_buddypress	a:1:{s:10:"apply_tags";a:1:{i:0;s:30:"gracefuldev:grouptype:interest";}}
127021	28156	wpf_settings_buddypress	a:1:{s:10:"apply_tags";a:1:{i:0;s:29:"gracefuldev:grouptype:special";}}
128534	28322	wpf_settings_buddypress	a:0:{}
128548	28323	wpf_settings_buddypress	a:1:{s:8:"group_link";a:1:{i:0;s:34:"gracefuldev:profiletype:instructor";}}
128562	28324	wpf_settings_buddypress	a:1:{s:8:"group_link";a:1:{i:0;s:34:"gracefuldev:profiletype:pro-member";}}
109443	25668	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
109637	26169	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
109670	26172	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
109711	26186	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
109734	26193	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
109774	26203	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
109809	26209	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
109978	25670	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
109997	25679	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}
110016	25681	wpf-settings	a:4:{s:8:"redirect";i:0;s:12:"redirect_url";s:0:"";s:11:"apply_delay";i:0;s:12:"lock_content";b:0;}

Make friends with SQL

ChatGPT: The Web as your REPL



rewrite gracefuldev_add_xprofile_fields_to_registration such that $field_ids is sourced from the gracefuldev_registration_xprofile_field_ids option, and ensure $field_ids is always an array

You can modify the `gracefuldev_add_xprofile_fields_to_registration` function to source `$field_ids` from the `gracefuldev_registration_xprofile_field_ids` option and ensure that `$field_ids` is always an array. Update the function in your site-specific plugin or child theme's `functions.php` file as follows:

php Copy code

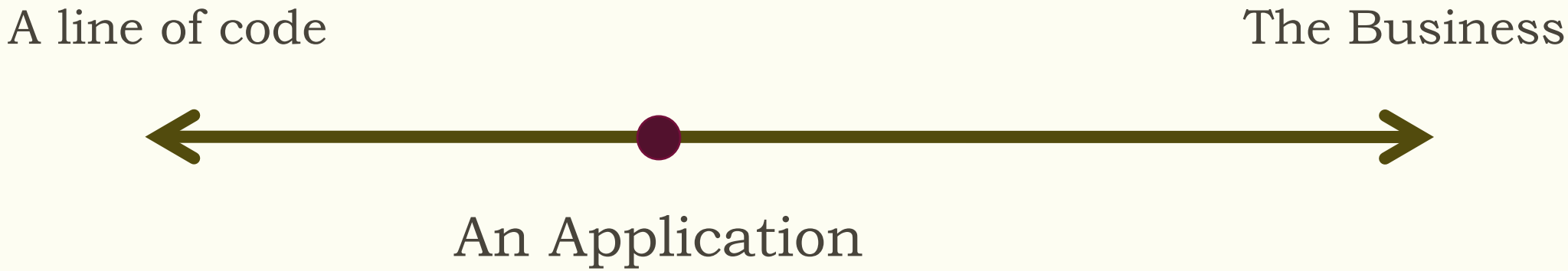
```php
function gracefuldev_add_xprofile_fields_to_registration() {
    // Fetch the saved xProfile field IDs from the option
    $field_ids = get_option( 'gracefuldev_registration_xprofile_field_ids',

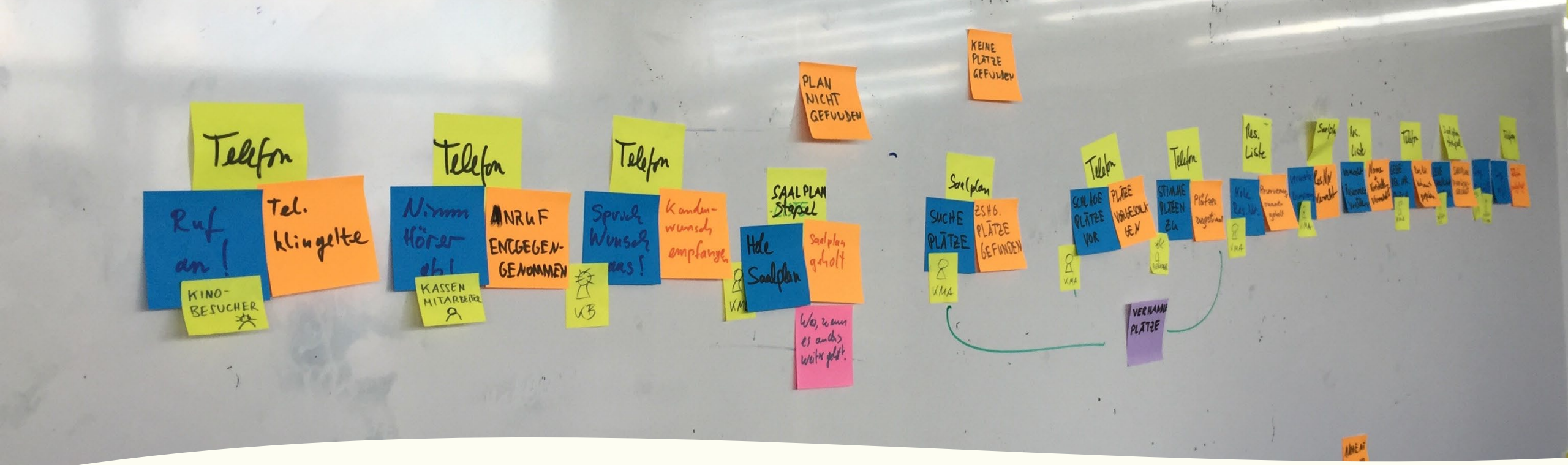
    // Ensure $field_ids is always an array
    if ( ! is_array( $field_ids ) ) {
        $field_ids = array();
    }

    foreach ( $field_ids as $field_id ) {
        // Get the field object
        $field = xprofile_g
```

Regenerate response

Send a message.


A line of code
The Business
An Application

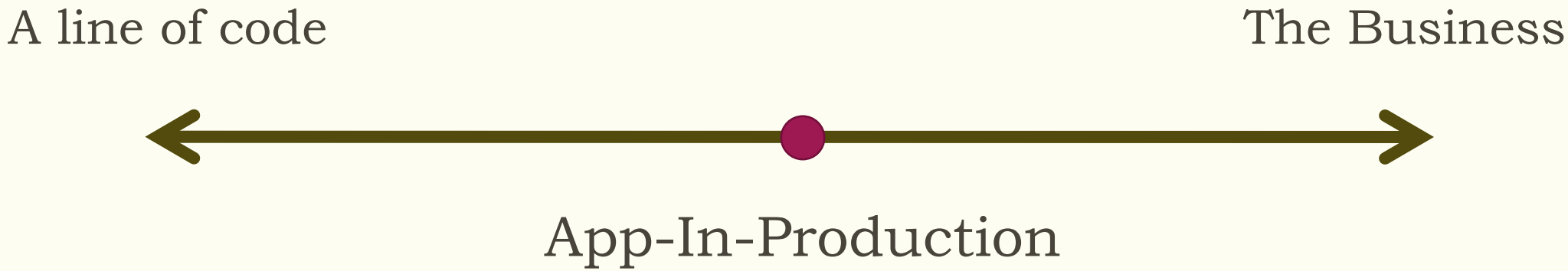


Design with tactile, conversational methods

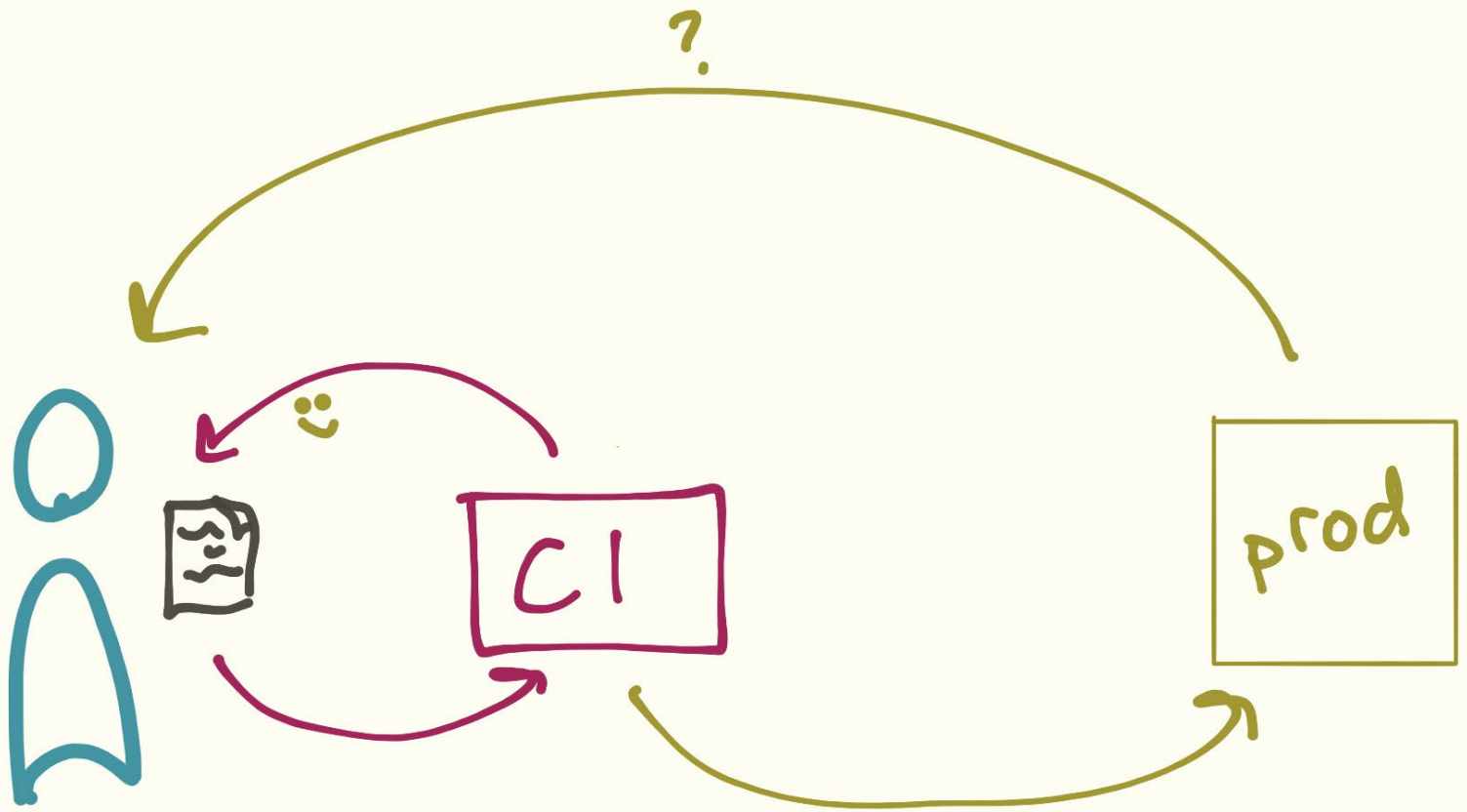
- CRC Cards
- Event Storming
- Whiteboards

Automate scenario reproduction

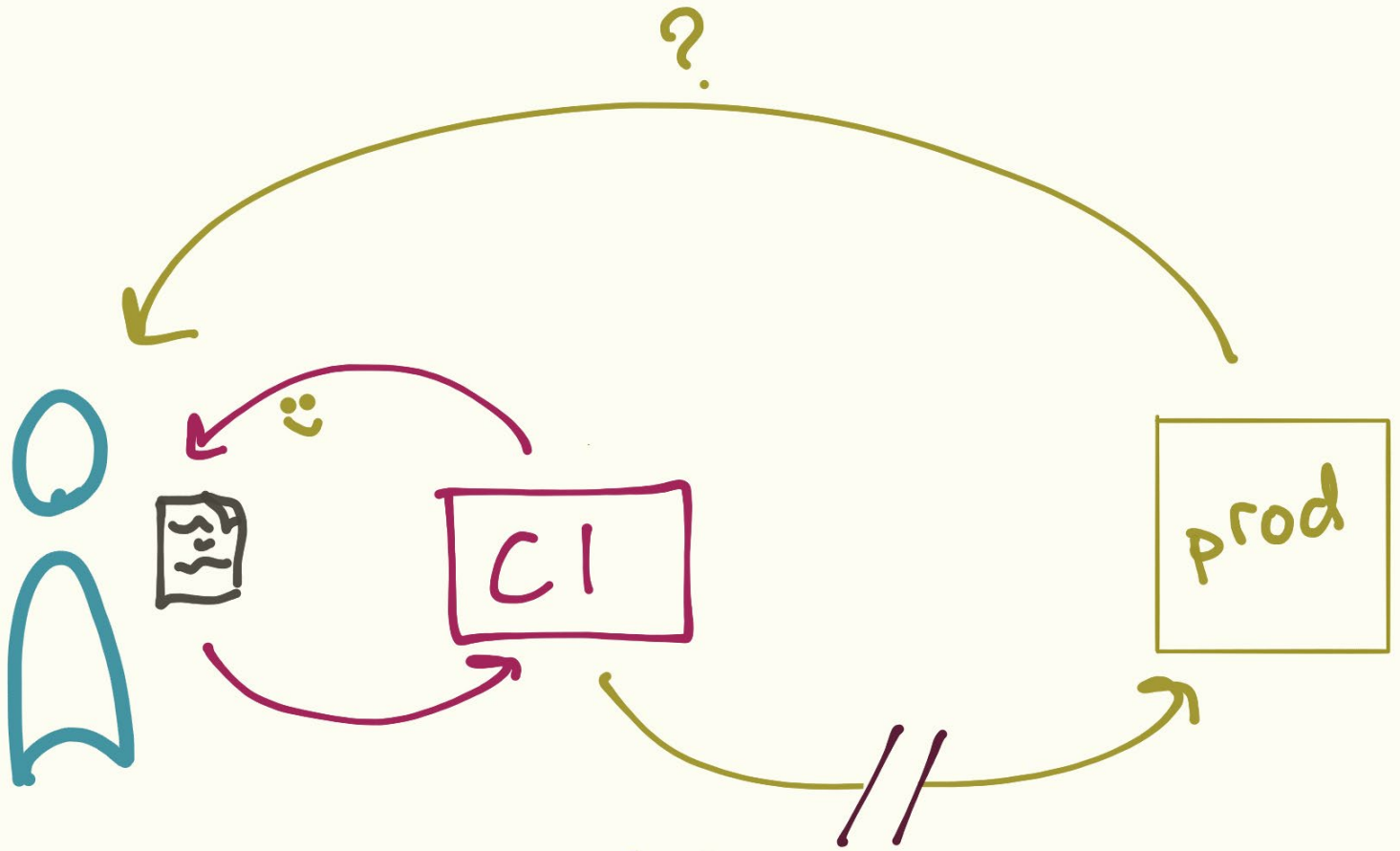
```yaml
options:
  courses:
    shared_steps: true
lessons:
  - name: Lesson A
  - name: Lesson B
  - name: Shared Lesson
    primary_course: Course A
courses:
  - name: Course A
    lessons:
      - Lesson A
      - Shared Lesson
  - name: Course B
    lessons:
      - Lesson B
      - Shared Lesson
students:
  - name: Avdi
    current_step: Lesson B
```


A line of code
The Business
App-In-Production

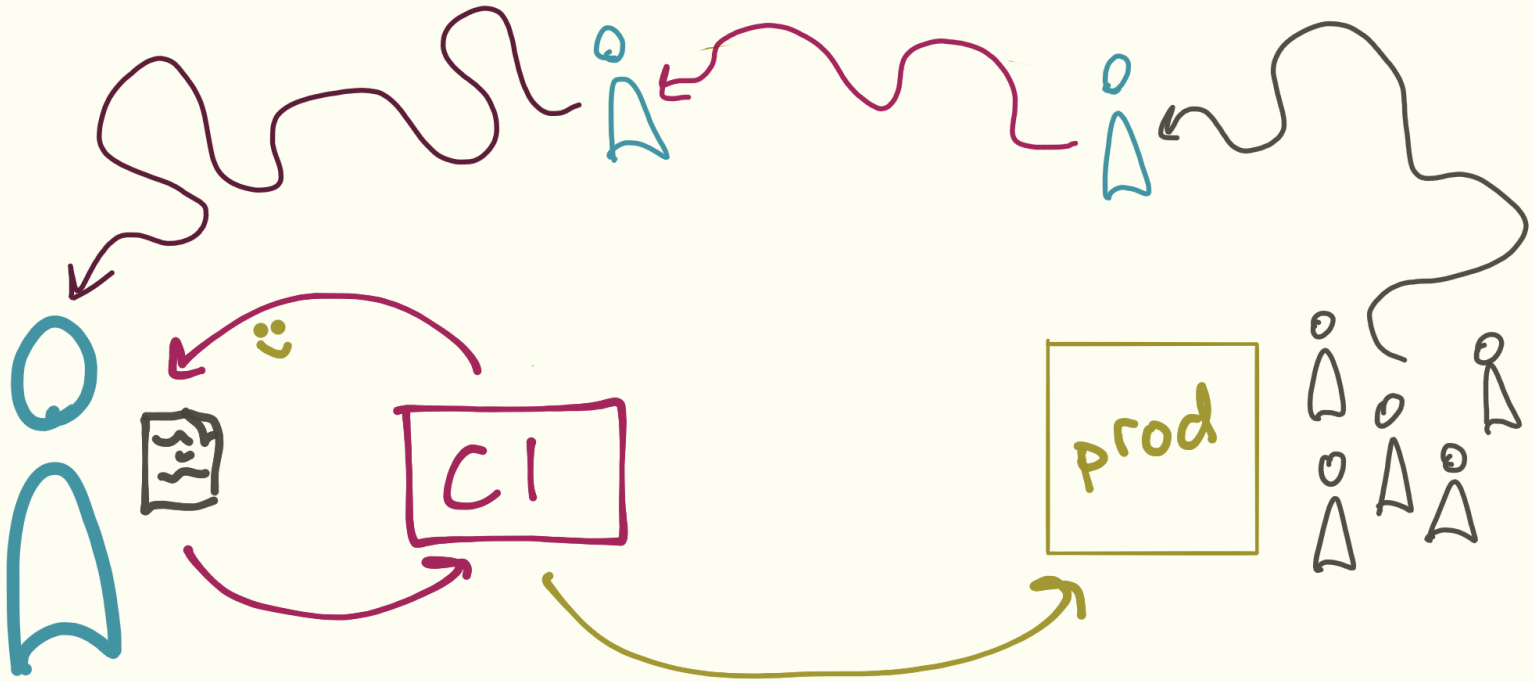

who swiftly spots the error via his instrumentation
then adds tests & instrumentation to better detect
and promptly commits a fix
virtuous
loop:



Use continuous deployment



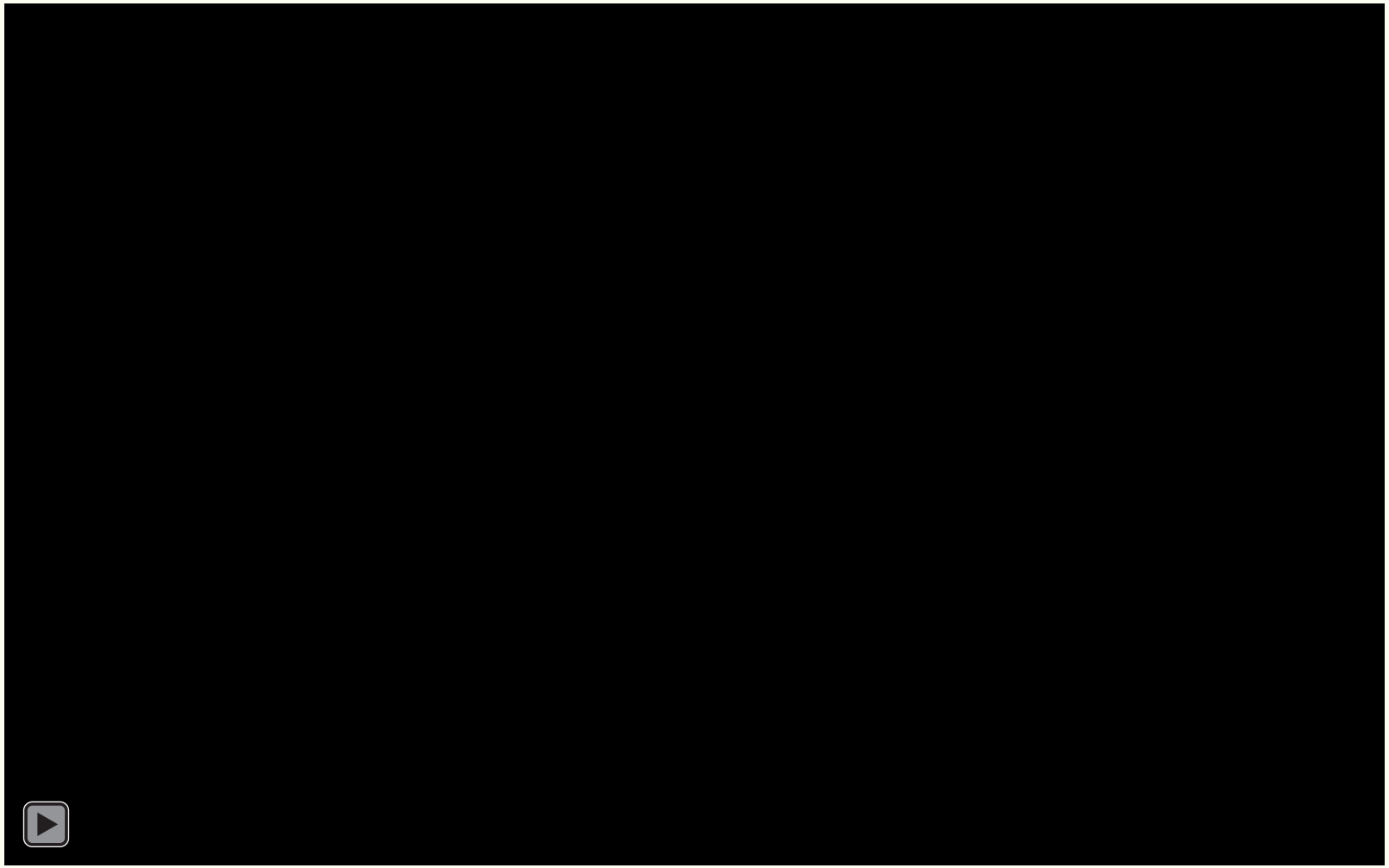
...but beware!



...also beware!

Release with feature flags





Use distributed tracing


A line of code
The Business
Product Team


Make friends with
customer support



Make friends with your salesperson

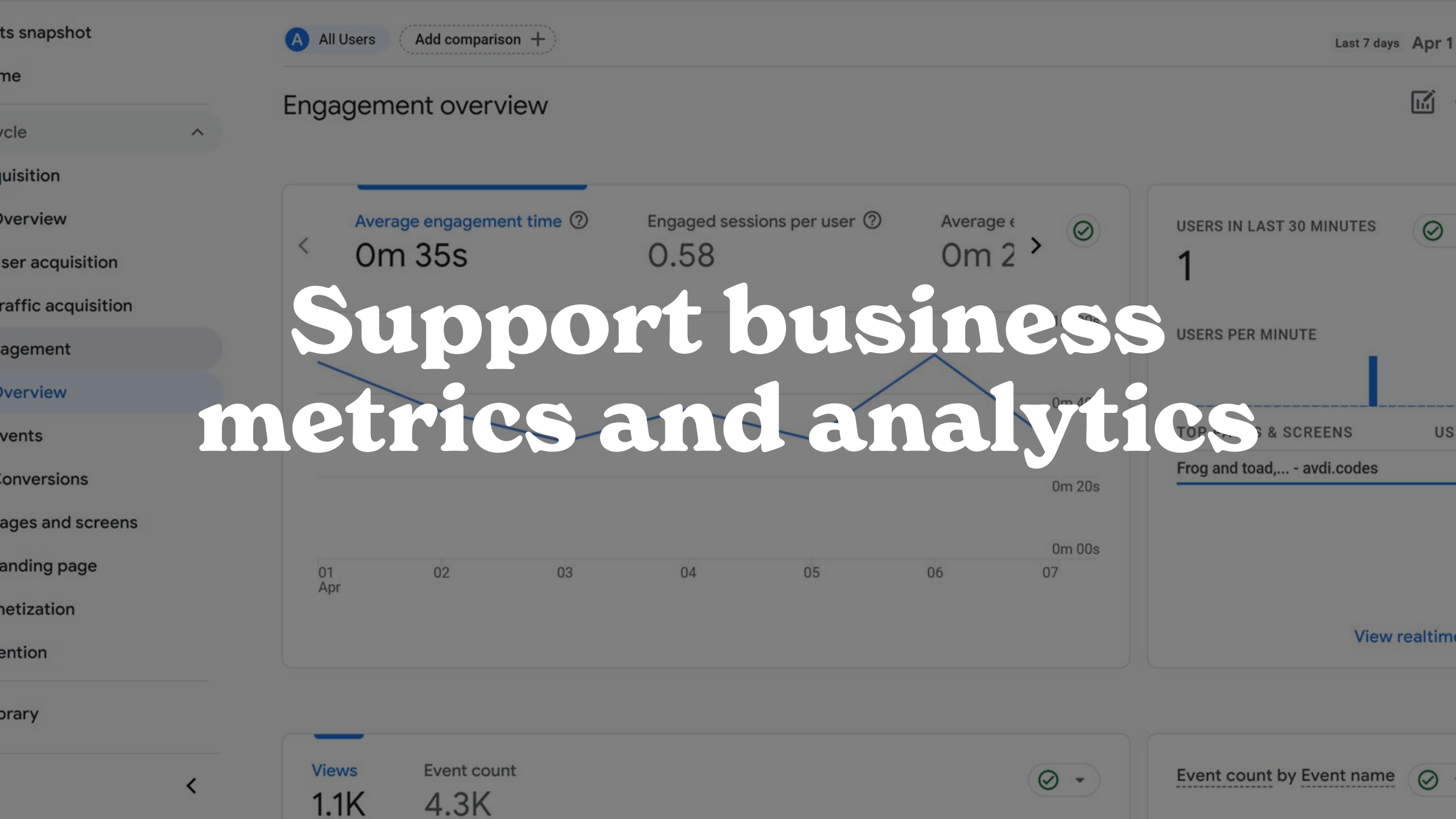
A line of code

The Business



Facilitate REPL Nature

(because it’s not all about you)



Support business metrics and analytics


Provide spreadsheets



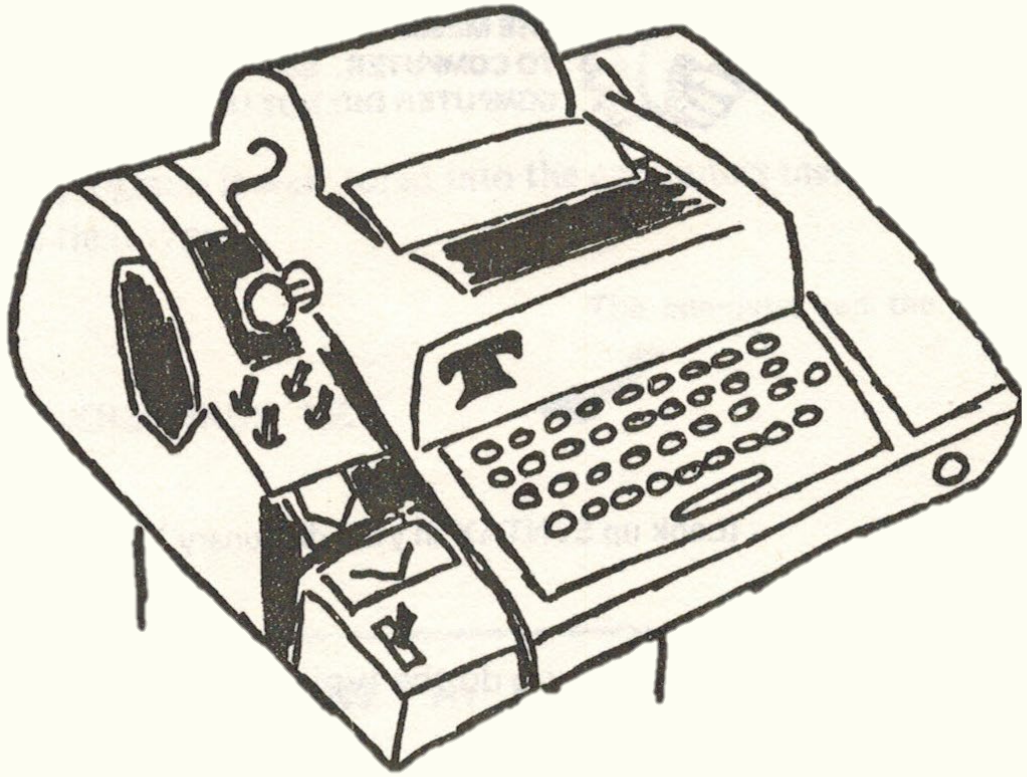
WordPress

Commodity Software for Commodity Needs

Stop making engineering the bottleneck for business experiments!

Seek Exploratory Feedback

READ	EVAL	PRINT	LOOP
• Cultivate safe ways to play	• Automate the repetitive, leave the rest open-ended	• Hope for surprise • Manufacture serendipity	• Keep trying • Seek new questions, not mere answers.
Safe	**Automanual**	**Exploramental**	**Conversational**



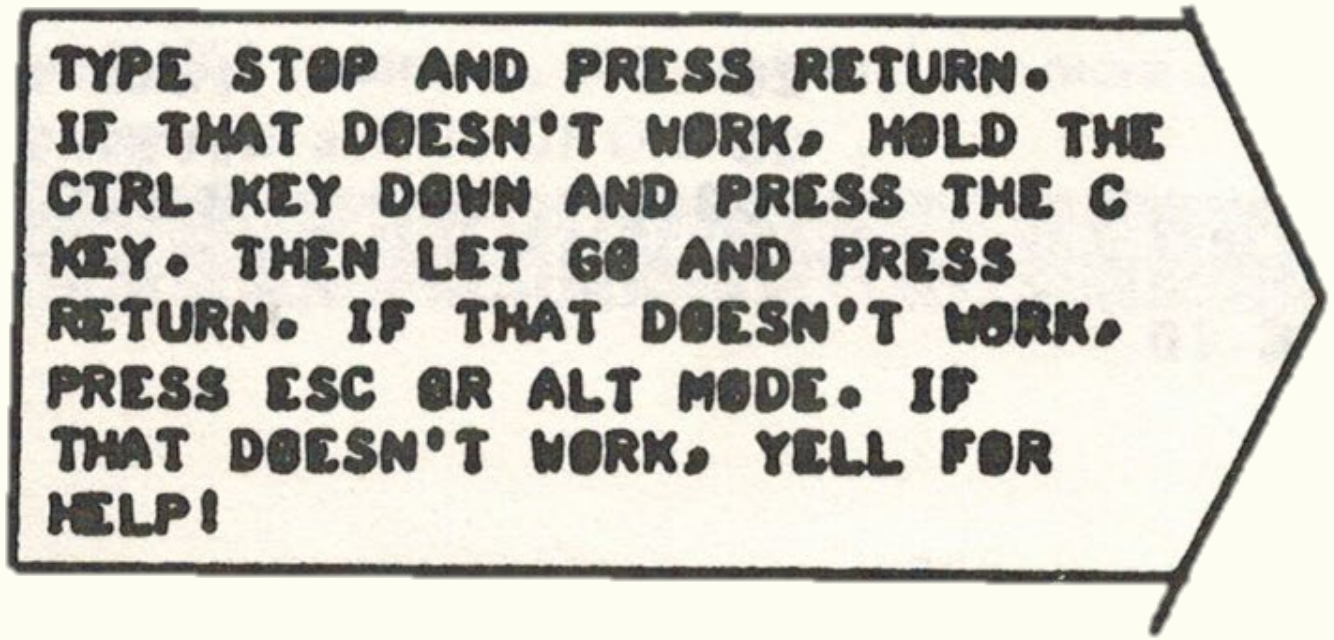
Chase new questions.

You'll get better answers.

Special thanks: Jessica Kerr







Good luck!

GRACEFUL DEV

graceful.dev/GOTOchgo/

goto;

Don't forget to
vote for this session
in the **GOTO Guide app**

References

- Albrecht, Bob 1972: *My Computer Likes Me When I Speak Basic*
- Carrol, John M. 1992: *Making Errors, Making Sense, Making Use*. In *Software Development and Reality Construction*
- Deutch, L. Peter and Berkely, Edmund 1964: *The Lisp Implementation for the PDP-1 Computer*
- DTSS Emulator
- Flutter, by Google
- Hendrickson, Elizabeth 2013: *Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing*
- Martin, Kit; Horn, Michael; Wilensky, Uri 2020: *Constructivist Dialogue Mapping Analysis of Ant Adaptation*
- Papert, Seymor and Harel, Idit 1991: *Situating Constructionism*
- Perez, Carlos 2018: *Touch (Not Vision) is the Foundation of Human Cognition*
- Project Jupyter
- Rankin, Joy Lisi 2018: *A People's History of Computing in the United States*
- Royce, Winston 1970: *Managing the Development of Large Software Systems*
- Thomas, Dave 2015: *Agile is Dead, Long Live Agility*

Credits

- Photo of a PDP-1: Alexey Komarov, CC BY-SA 4.0 via Wikimedia Commons
- Event storming example process: Henning Schwentner, CC BY-SA 4.0, via Wikimedia Commons