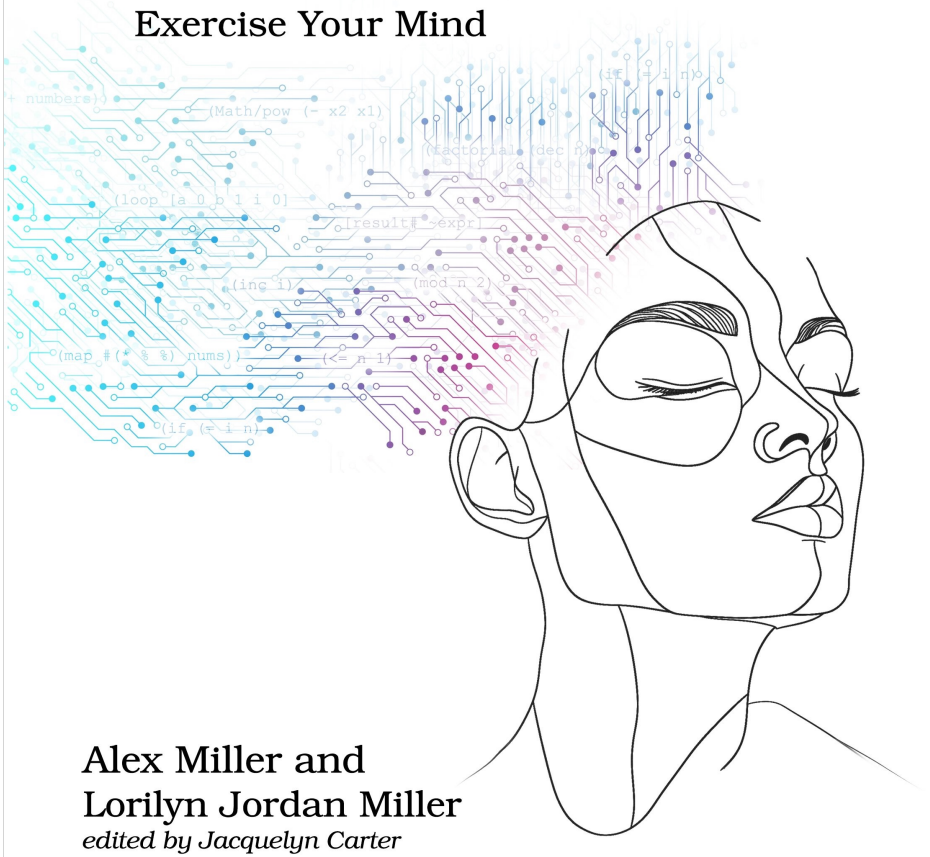
The Pragmatic Programmers

Clojure Brain Teasers

Exercise Your Mind



Alex Miller and
Lorilyn Jordan Miller
edited by Jacquelyn Carter

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit https://www.pragprog.com.

Preface

Clojure was designed to be a general-purpose functional langauge with a simple programming model for information systems (far simpler than the object-oriented models that dominate the industry). Many people have commented that aspects of Clojure like immutable collections, functional programming, state management, concurrency, and open polymorphism seem to fit together and support each other at a suprisingly deep level. Indeed, this is a consequence of the intense design effort that went into Clojure's initial creation and the attention the core team pays to design with every new feature.

However, some aspects of Clojure may defy expectations when coming from other languages, or be unclear if the underlying design is not understood. The authors have spent many years assisting new developers learning Clojure and the same questions arise repeatedly. This book is designed to explore these common questions. Any new or intermediate Clojure developer will find familiar questions in these pages and it is likely you will find some you have encountered in the past.

Additionally, Clojure is by design a *hosted language*, running on top of some other runtime, like the JVM, Javascript, the Microsoft CLR, or even Bash. Being hosted gives Clojure a huge advantage in library availability and integration with existing systems, but also opens a plethora of situations where Clojure and host semantics may differ, also defying user expectations.

About the Book

Each teaser starts with a simple program and asks you to guess the result. The teasers are designed to either produce a `true` or `false` or may in some cases throw an exception.

After you've turned the page, you'll find out what actually happens and get an explanation of why the program returns the answer it does, often with some discussion of related topics. These teasers are not "gotchas" and do not exploit bugs in the language. Rather, these teasers reveal places where your

prior experience or expectations have lead you to a conclusion different than Clojure's design. But fear not, as you read this book the underlying intent of the language will be revealed.

You will be able to use this knowledge in the future to better predict and leverage how your own Clojure programs work.

About the Authors

Alex Miller has been a member of the Clojure core team since 2013 and is the co-author of *Clojure Applied [VM15]* and *Programming Clojure, Third Edition [HB18]*. Alex is the creator or organizer of the Strange Loop, Lambda Jam, Clojure/west, and Clojure/conj conferences.

L. Jordan Miller is a Staff Software Engineer at Nubank, Datomic Developer Advocate, and producer/host of the Lost In Lambduhhs podcast. Passionate about people, programming, and pedagogy, she's a conference speaker, host, organizer, and reviewer for events such as Heart of Clojure, Strange Loop, Re:clojure, and Clojure Conj.

About the Code

Each teaser starts with a small code snippet that can be run at the Clojure REPL (Read-Eval-Print Loop). The easiest way to run the Clojure REPL is to install the Clojure CLI (https://clojure.org/guides/install_clojure), then execute the command `clj` from your shell. If you're using a development environment, just open any REPL to type the teaser code.

In code examples, the REPL prompt is not printed and instead the expression is printed at the beginning of the line. Return values follow in a line starting `;;=>`. If output is printed (not returned), the line will start `;;`.

```
(+ 1 1)
;;=> 2

(println "Hello!")
;; Hello!
;;=> nil
```

In some cases the output has been slightly modified to improve clarity, for example by omitting return values that aren't important or simplifying exception messages.

Each teaser's source code is also included as a standalone source file in the sample code for this book, available on the Pragmatic Bookshelf website

(https://pragprog.com/titles/mmclobrain). There you can download teaser code, participate in discussions, and report issues if needed.

About You

This book assumes some basic familiarity with Clojure and its features. If you are completely new to Clojure, we suggest that you pause each time you encounter something new and read about it in your favorite Clojure introductory text or online resource, like *Programming Clojure, Third Edition [HB18]* or *Getting Clojure [Ols18]*. Because the topics are presented here example-first, you may find that they form a complementary path into learning.

You may want to read this while sitting by a REPL, trying each teaser before you turn the page. Or you may find it sufficient to just predict the answer in your brain REPL. Or you may just want to turn the page and read the answer, gauging your level of surprise. All of these are okay!

Have fun!