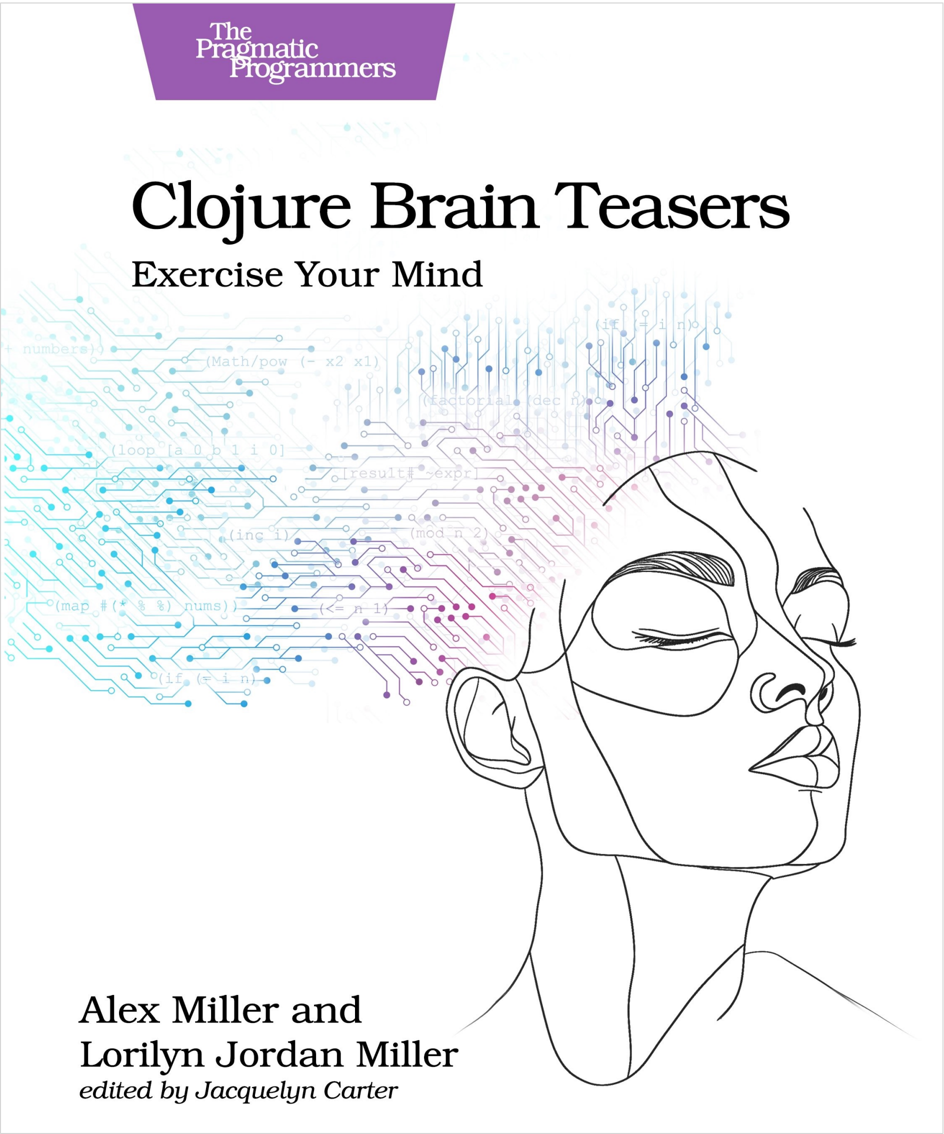

The Pragmatic Programmers

Clojure Brain Teasers

Exercise Your Mind

Alex Miller and
Lorilyn Jordan Miller
edited by Jacquelyn Carter

Puzzle 4

Truthy or Dare

book/src/truthy1.clj

```
(true? true)
(true? :sky-is-blue)
(false? false)
(false? nil)
(false? `())
(false? 0)
```

Guess the Output

Try to guess what the output of these expressions are before moving to the next page.

The program will display the following output:

```
(true? true)
;;=> true
(true? :sky-is-blue)
;;=> false
(false? false)
;;=> true
(false? nil)
;;=> false
(false? `())
;;=> false
(false? 0)
;;=> false
```

Discussion

Clojure has a concept of "logical true" and "logical false"; many people refer to these colloquially as "truthiness" and "falsieness." This means that everything except the values false and nil implicitly resolve to true in a conditional expression or when coerced into a Boolean using (boolean x).

Conversely, the predicate functions (true? x) and (false? x) check if the values of x are specifically the *Boolean values* true or false, which is distinct from the concepts of "logical truth" and "logical false."

You can use and, or, and not to combine logical values. One important and common use is to check whether a value is nil by implicitly using it as a truth value.

Understanding this can improve your code style. For example, to check if the value moms-birthday exists, and if so, print "Happy Birthday Mom!!", we have a few options. Each of these conditional functions has the same output, but one of them is typically preferred over the others.

```
(def moms-birthday "April 20, 1969")

(when-not (nil? moms-birthday)
  (println "Happy Birthday Mom!!"))

(when (some? moms-birthday)
  (println "Happy Birthday Mom!!"))

(when moms-birthday
  (println "Happy Birthday Mom!!"))
```

In the last example, moms-birthday is used as a “logical truth.” In contrast to the other more cumbersome expressions, it’s the most concise and readable. Remember this when reaching for function conditional operations such as if, when, cond, and so on.

Further Reading

ClojureDocs - clojure.core/true?
https://clojuredocs.org/clojure.core/true_q

“Clojure Truthy and Falsey” by Jay Fields
http://blog.jayfields.com/2011/02/clojure-truthy-and-falsey.html

Part II

Collections

Clojure is a data-oriented language, built on immutable collections (composite values). These teasers check your understanding of the Clojure collection types.