Elements of Functional Programming

* Pure Functions
  + Only rely on arguments
  + Have zero side-effects
  + Don’t change state
* First Class Functions
  + Functions can be…
    - assigned to symbols
    - passed as arguments
    - returned from other functions
* Immutability
  + Data structures cannot be modified
* Higher-Order Functions
  + Functions that operate on functions
* Recursion
  + Replaces iteration, “for” loops
  + Accumulators can be used to convert recursive process into iterative process
  + Tail recursion optimization (recur) can convert iterative process to iterative code
* Lazy Evaluation
  + Operations and functions are evaluated when used not when called
  + Simplifies the logic
* Currying
* Memoization
* Destructuring
* Collection Pipelines
* List Compressions
* Raw Data + functions
  + No objects

Why is immutability important?

* Concurrency is simpler
* No need for private data
* Easier to test, debug, and understand program

Elements of Clojure Code

* Symbols
* Keywords
* Literals
* Lists
  + ‘()
* Vectors
  + []
* Maps
  + {}
* Sets
  + #{}
* Functions
  + fn
  + defn
  + #(%1 %2)
* Macros
* Special forms (functions)

Important

* No built-in operators, just functions
* No precedence