Elements of Functional Programming

* Pure Functions
  + Functions whose output depends only on its input arguments
  + Have zero side-effects
  + Importance: easier…
    - Testing
    - Debugging
    - Understanding the program
* First Class Functions
  + Functions can be assigned as variables, passed as arguments, or returned from functions
  + Importance:
    - Flexibility
    - Generality
* Higher-Order Functions
  + Functions that take other functions as arguments or return functions
  + Functions that operate on other functions
  + Importance:
    - Fewer details
    - Higher level logic
    - Concurrency
* Immutability
  + Data structures cannot be modified
  + Importance:
    - Concurrency
    - No need for private data
* Lazy Evaluation
  + Operations and functions are evaluated when used, not when called
  + Importance:
    - Simplifies logic
* Recursion
  + Replaces iteration
* Currying
  + Method of breaking down a function that takes multiple arguments into a series of functions that take part of the arguments
* Memoization
  + (memoize f)
  + Caches result of function f and uses that value next time f is called with same arguments
* Destructuring
* Collection Pipelines
  + Pipe a sequence through a set of functions passing the result of a function as input to the next
  + Why important?
    - Higher level logic
    - Concurrency
* List Compressions
* Raw Data + functions

Elements of Clojure Code

* Atoms
* Agents
* Symbols
* Keywords
* Literals
* Lists
* Vectors
* Maps
* Sets
* Functions
* Macros
* Special forms (functions)

Tail Recursion Optimization

* Process by which a tail recursive function call is optimized to use a single stack frame
* In a recursive function implementing an iterative process, the compiler can optimize the recursion into iteration

Processing Collections Consider Using

* map
* reduce
* filter
* for
* some
* repeatedly
* sort-by
* keep
* take-while
* drop-while

Control Structures

* Loops
* Blocks
* Branch

Branching

* if
* if-not
* if-let
* if-some
* when
* when-not
* when-let
* when-first
* when-some
* cond
* condp

Idiomatic Clojure

* Using collections as functions

Rest verses next

* next has to look at the next element, causing it to be computed
* rest does not look at the next element