CPS506 Comparative Programming Languages
Functional Programming

Dr. Dave Mason
Department of Computer Science
Ryerson University

RYERSON UNIVERSITY

Why FP?

- simpler model
- no state to reason about
- easier for proofs
- very expressive
- leverage multi-core

When FP?

- LISP - 1957 - first-class functions
- APL - 1962 - no globals
- ML - 1973 - Hindley-Milner type inference
- Hope*! - early 1970s - call-by-pattern, algebraic data types
- Miranda* - 1985 - proprietary
- Haskell* - 1990

Who FP?

- LISP/Scheme - John McCarthy / Guy Steele
- APL/FP - Ken Iverson / John Backus
- ML - Robin Milner / Dave MacQueen / Robert Harper
- Haskell - Simon Peyton Jones / Paul Hudak / Phillip Wadler

How FP?

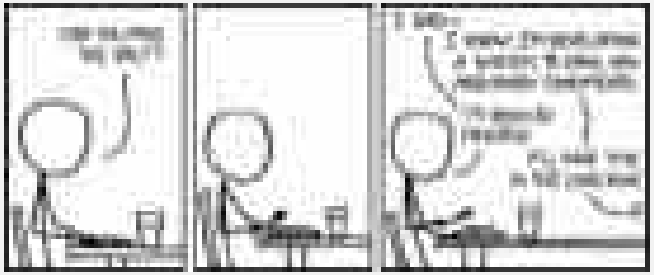
- λ-calculus
- first-class functions
- garbage collection
- literal constructors
- tail-recursion
- closures

What FP?

- return values vs. modify state
- functions as values
- list comprehensions
- limited/no mutability (FRP)

Where FP?

- lots of companies where thinking more important than coding
- ...'tho that can be taken too far



- all of the languages we're discussing used in some large companies
- Paul Graham of Y-combinator fame made his money using LISP as competitive advantage

Assignment versus Binding

- non-mutation
- binding
- matching
- scope

Simple Functions

- simple functions: `fn x -> x + 1 end`
- function composition
- function piping

First-Class Functions

- map
- filter

Closures

- first-class functions retain bindings
- static scope
- pure-functional makes this easy