INF 212
ANALYSIS OF PROG. LANGS
FINAL LECTURE

Instructors: Crista Lopes

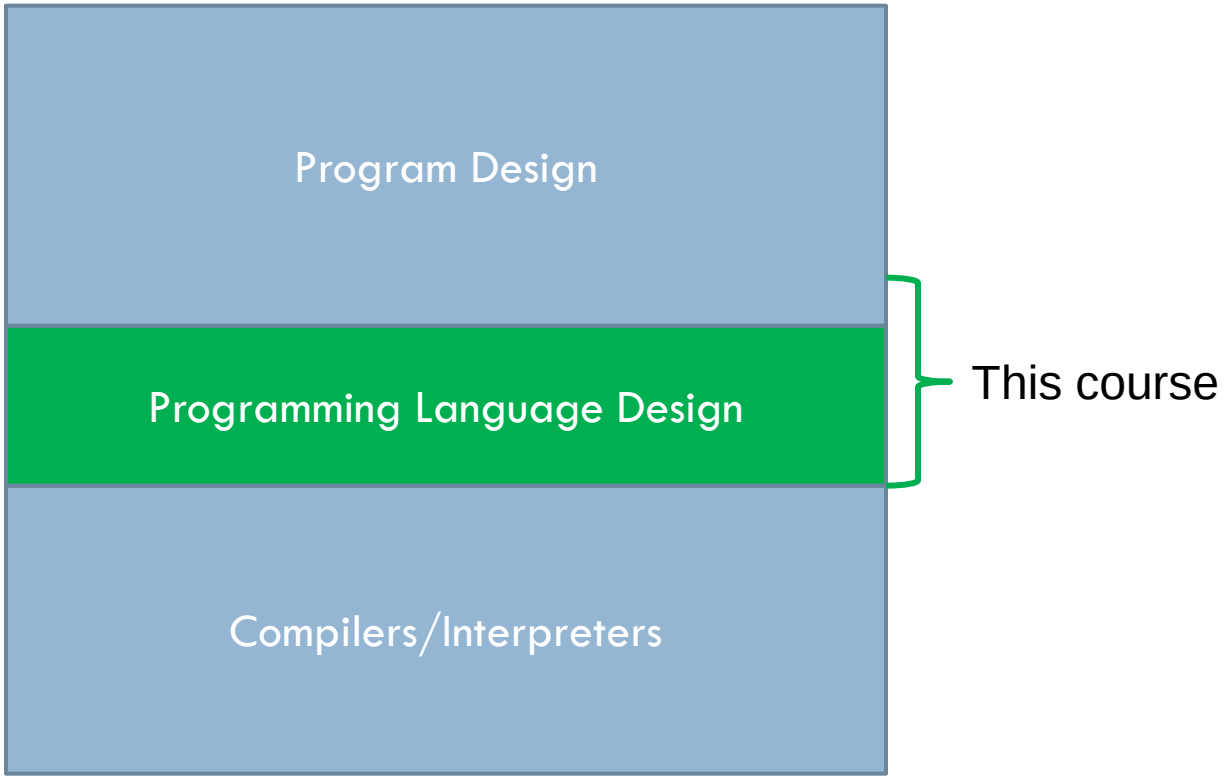
Programming Languages

- Universe of design ideas
- Crazy concepts often became mainstream
  - E.g. garbage collection, recursion, closures, …
  - (other crazy concepts were just crazy)
- Language design concepts often pop out into systems design concepts
  - E.g. Map-Reduce, stateless – REST, …

Course Objectives

- Understand concepts in PLs
  - 100's of PLs, all *look* different → they aren't that different
  - Appreciate history, diversity of ideas in PLs
  - Be prepared for new languages
  - See beyond hype & sales pitches
- Learn some important facts about existing language systems and techniques
- Learn and think critically about tradeoffs

Where this course stands



We will cover how PLs influence program design and vice-versa

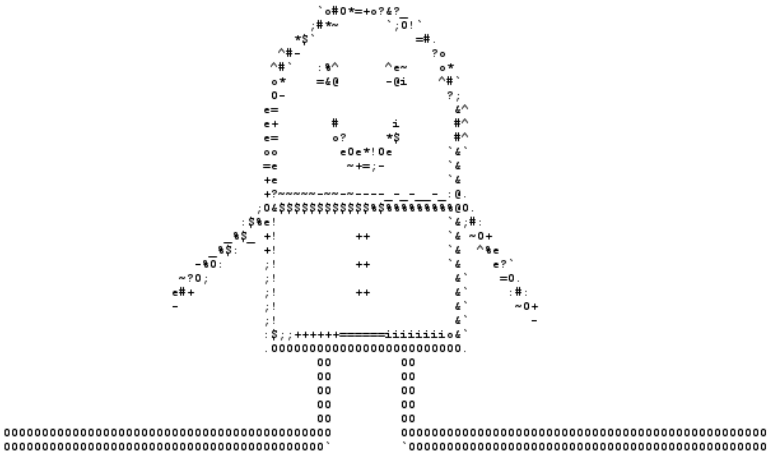
Not covered: implementation of PLs. Recommended:

- CS 241 Advanced Compiler Construction
- EECS 221 Program Analysis

The Course

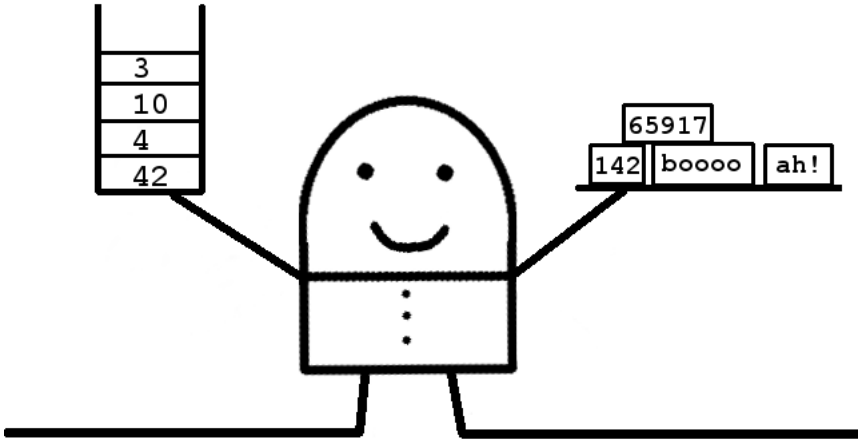
- Historical Languages
- Basics of PLs
- Function composition
- Objects and object interactions
- Reflection and metaprogramming
- Adversity: dealing with the outside world
- Data-centric concepts
- Concurrency
- Interactivity

Historical concepts



Good Old Times

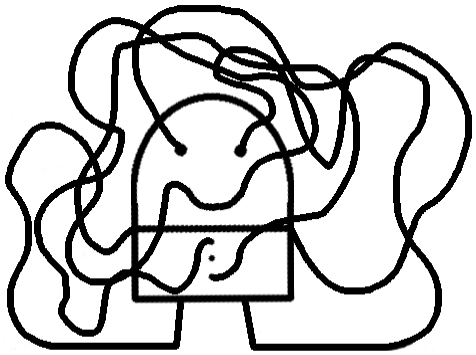
- Flat memory
- No names



Go Forth

- Stack machine

Basic Concepts

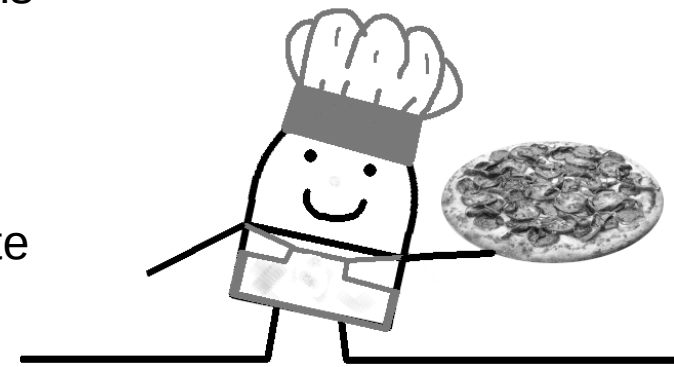


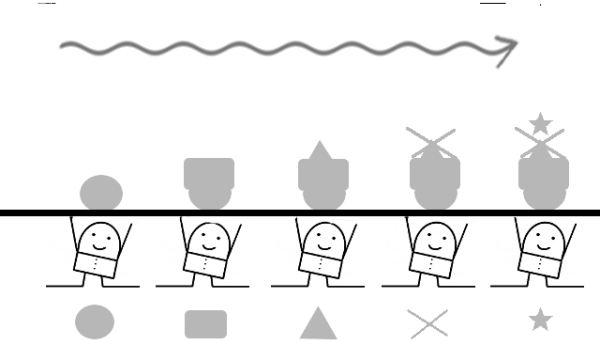
Monolithic

- No definition of abstractions
- No use of abstractions

Cookbook

- Shared state
- Steps



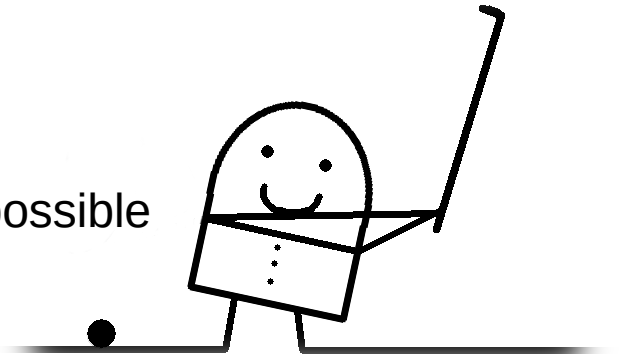


Candy Factory

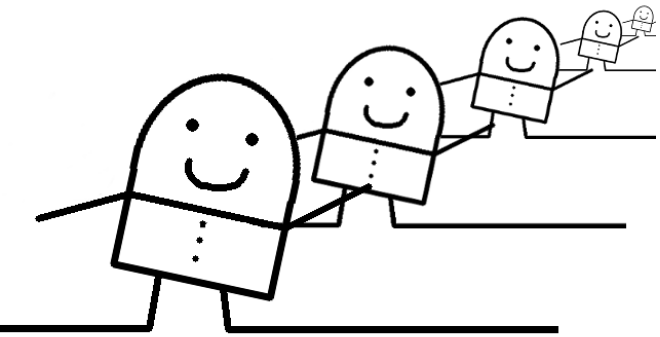
- No shared state
- Pipeline of functions

Code Golf

- As few LOCs as possible



Function Composition Concepts

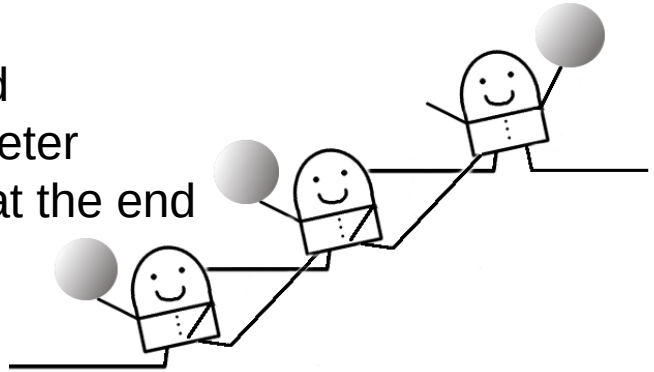


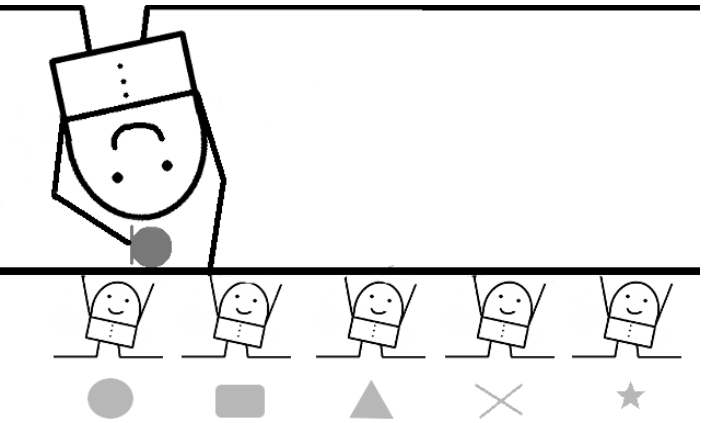
Infinite mirror

- Induction

Kick Teammate Forward

- Extra function parameter
- That function called at the end

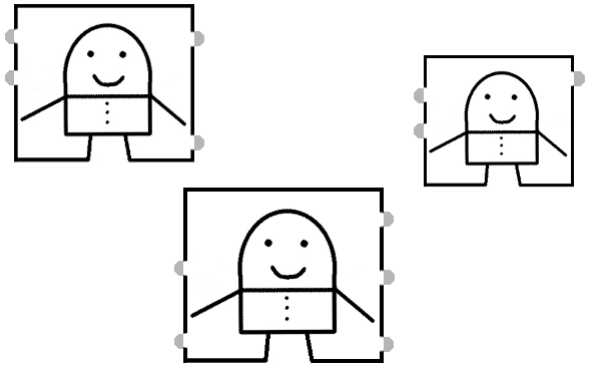




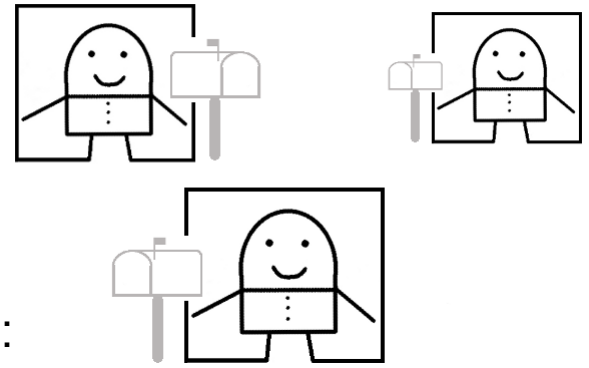
The One

- Universal object wraps around values
- Chains functions via bind

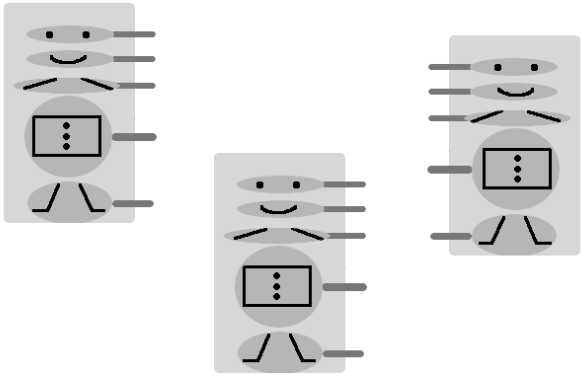
Objects and...



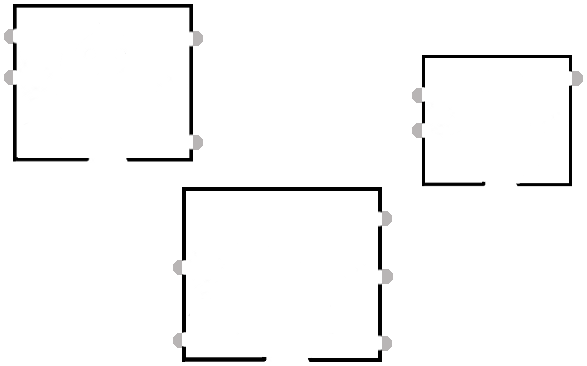
Things
- Capsules of data exposing procedures



Letterbox
- Single procedure: receive message

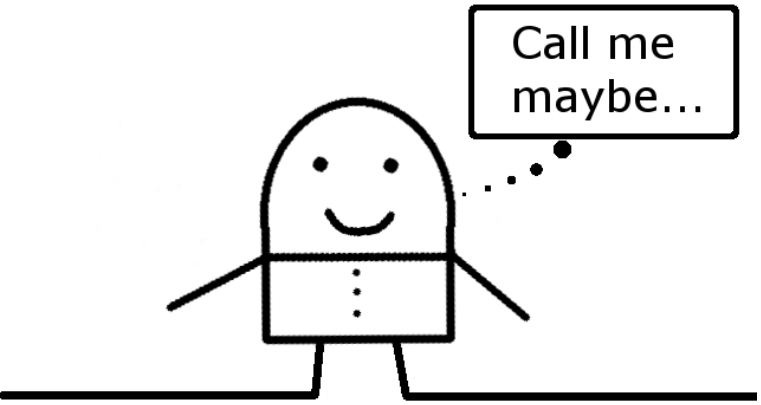


Closed maps
- Hashes mapping keys to data or procedures



ADTs
- Abstract interfaces to things

...Object Interactions

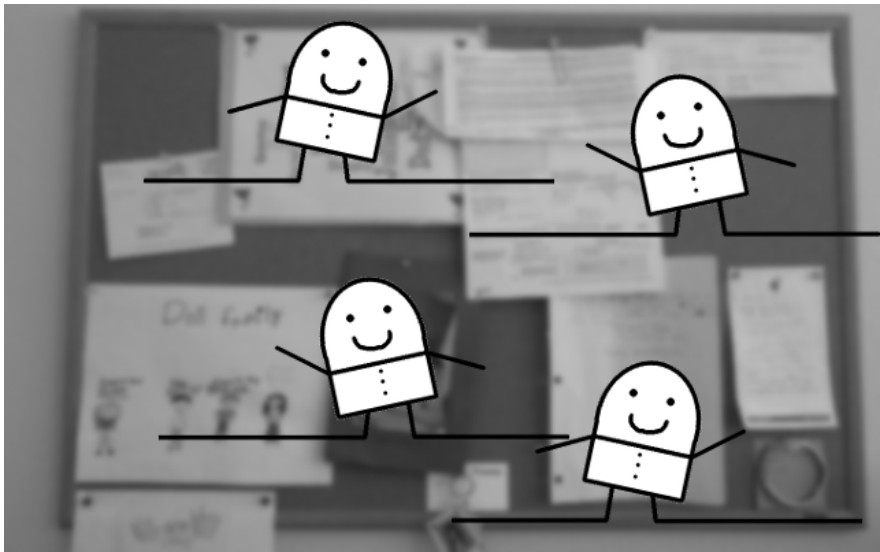


Hollywood

- “Don’t call me, I’ll call you”

Bulletin Board

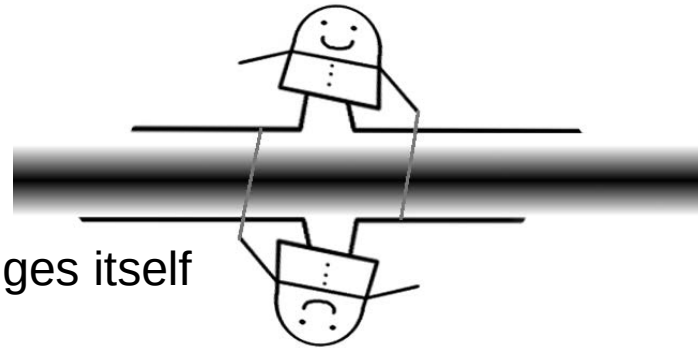
- Publish events
- Subscribe to events



Reflection and Metaprogramming

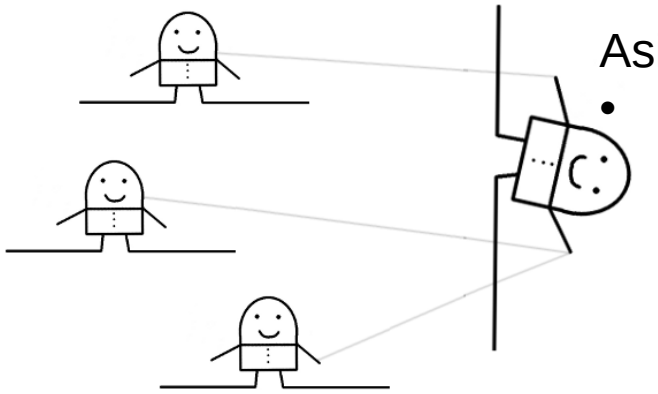
Introspection

- The program sees itself



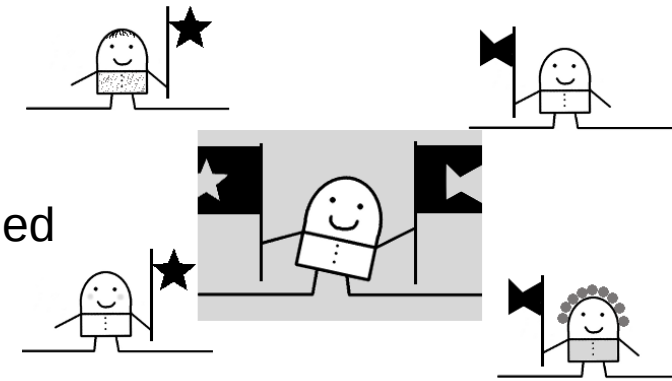
Reflection

- The program changes itself



Asides

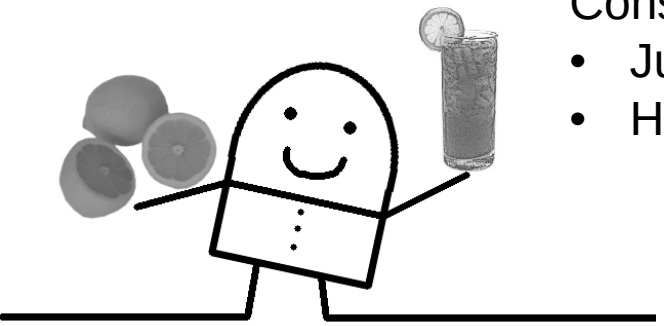
- A separate piece of code affects base functions



Plugins

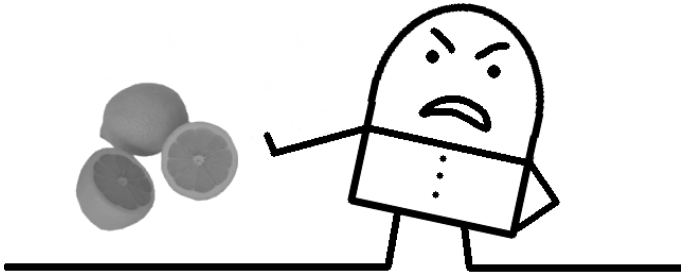
- Alternatives loaded dynamically

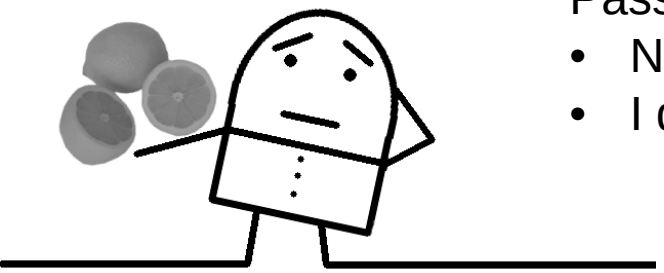
Adversity Concepts



Constructivist
- Just fix it
- Hope for the best

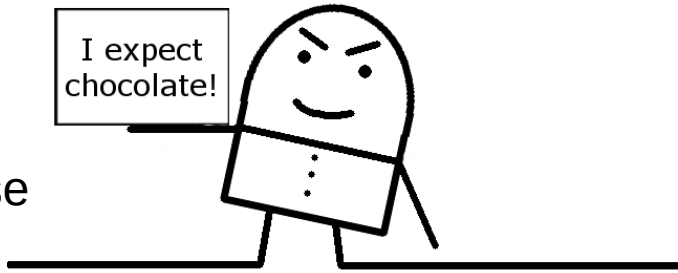
Tantrum
- NO GO!



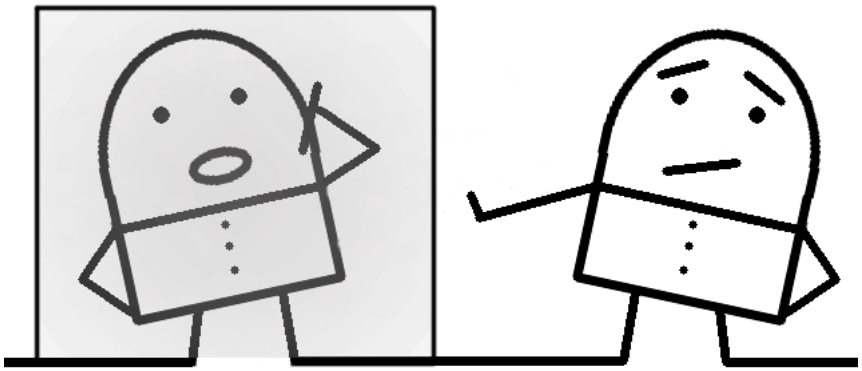


Passive-Aggressive
- No go, but
- I don't want to deal with it

Declared intentions
- Good types or else



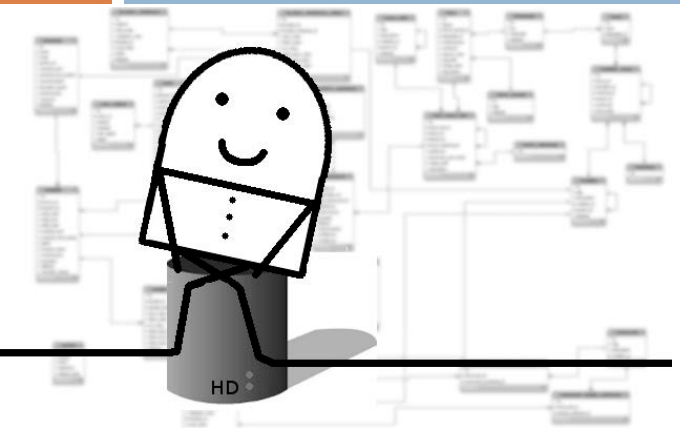
Adversity Concepts



Quarantine

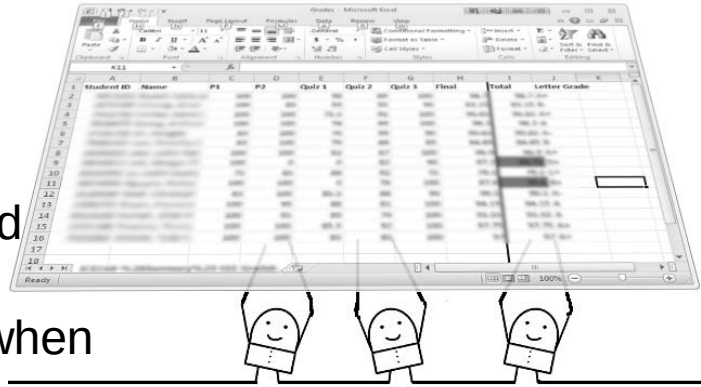
- IO: eeeeewww!
- Isolate IO functions

Data-Centric Concepts



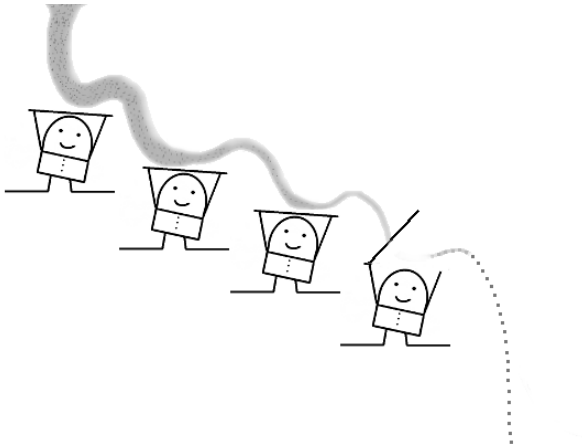
Persistent Tables

- Relational data
- Queries over data



Spreadsheet

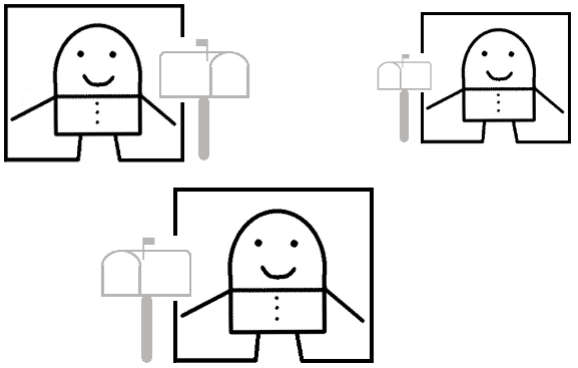
- Data with associated update functions
- Functions invoked when data changes



Lazy rivers

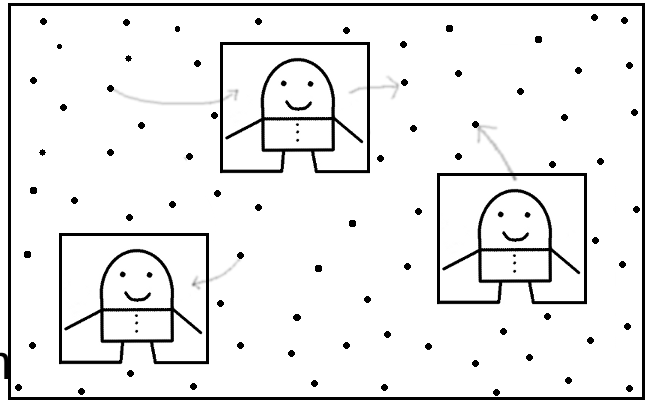
- Iterators
- Generators

Concurrency Concepts



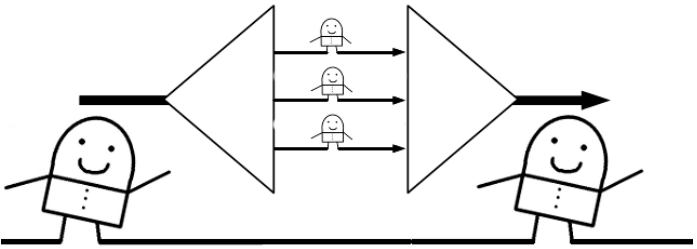
Free Agents

- Things with threads
- Message passing
- Thread-safe queues



Dataspaces

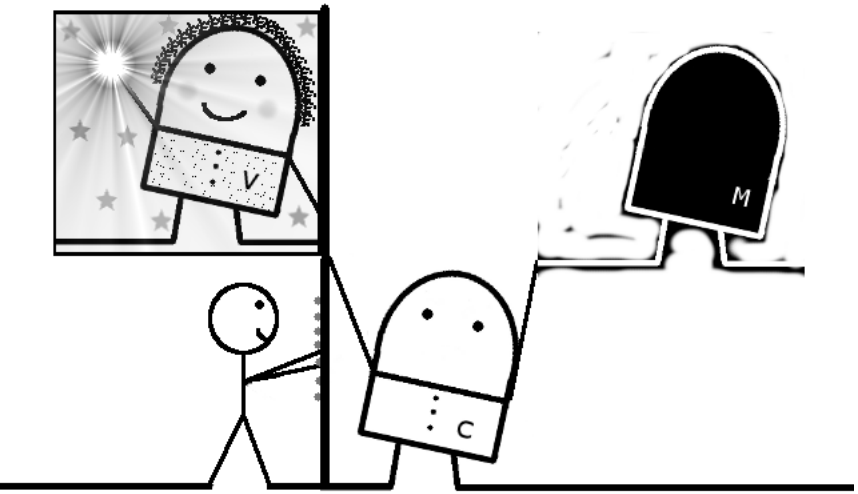
- Thread-safe data spaces
- Workers get/put data in them



Map Reduce

- Data is partitioned in independent chunks
- Chunks given to workers (mappers)
- Partial results given to workers (reducers)

Interactivity Concepts



Trinity Model/View/Controller

- 3 categories of code elements

RESTful / Forgetful

- Request-Response interaction
- Resources
- Uniform interfaces
- Hypermedia as engine of app state

What's worth studying?

- Dominant languages and paradigms
  - Leading languages for general systems programming
  - Explosion of programming technologies for the web
- Important implementation ideas
- Performance challenges
- Design tradeoffs
- Concepts that research community is exploring for new programming languages and tools
  - E.g. Multi-core