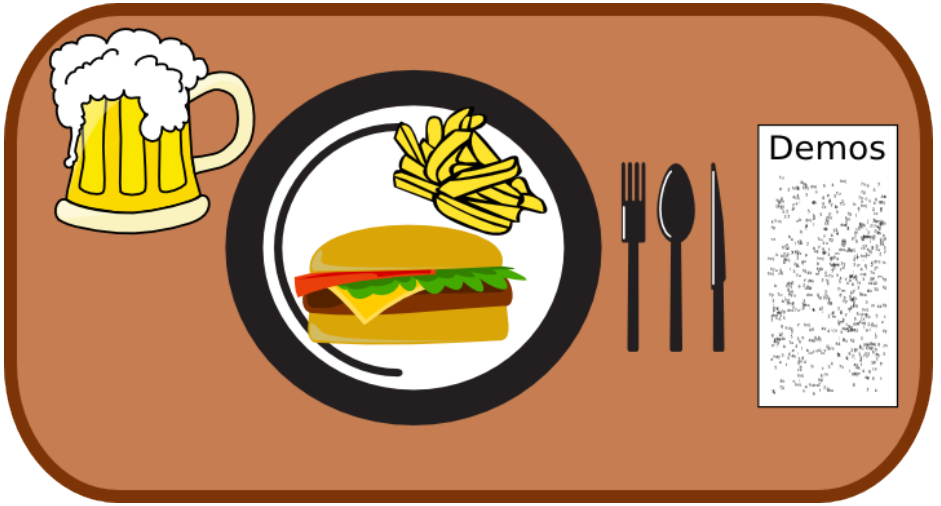
How Rosetta is a cheeseburger:
a simile, unless that's like a metaphor

Steven Lewis

RosettaCON 2013



Slides available; screencast of old version available

Organization exists, but I prefer questions

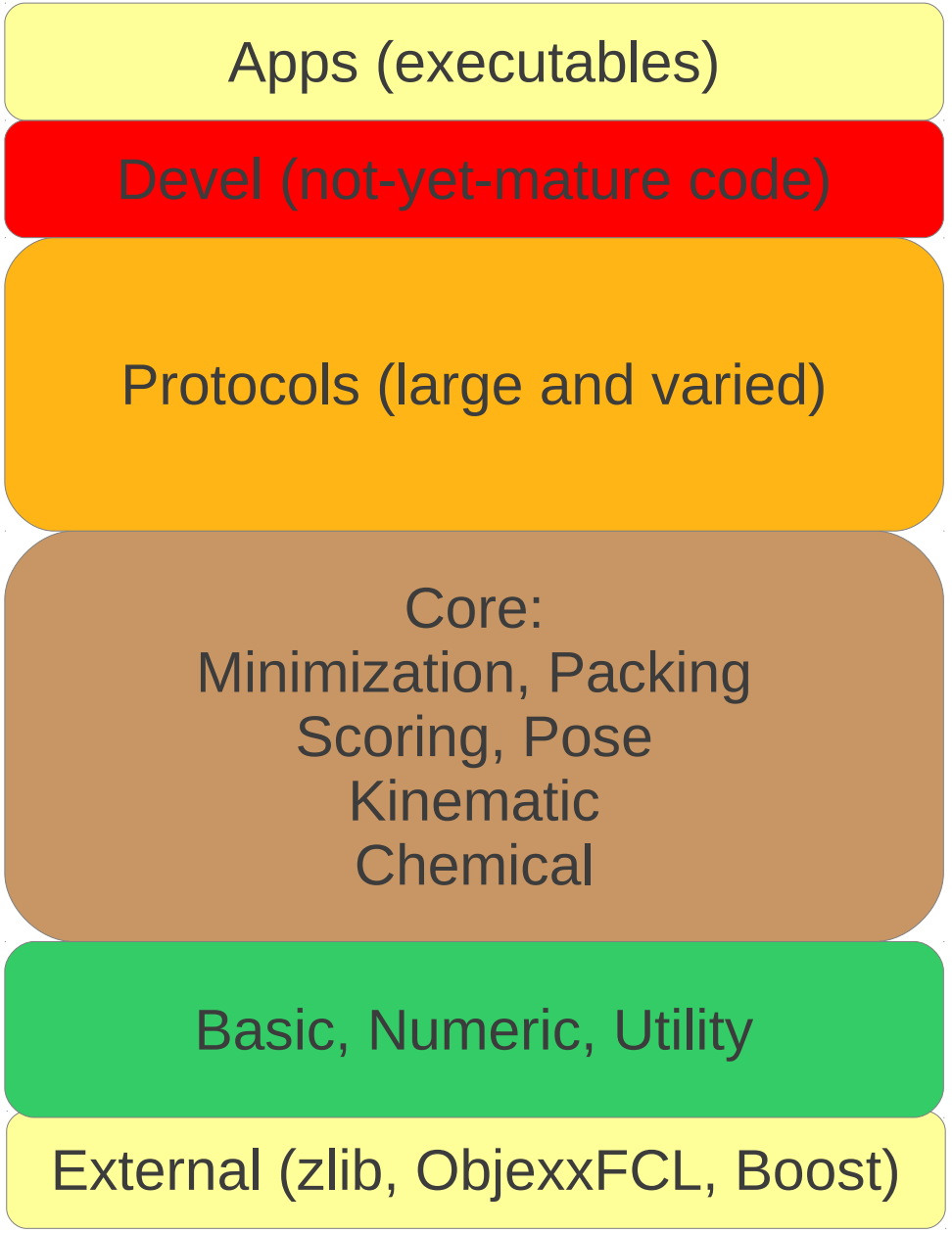
This may be a firehose, but at least you'll get wet

New developers to the front!

Tutorial

- I assume:
  - You know some C++
  - You know what classes are, and inheritance
  - You've glanced at the codebase at least once
- We'll cover:
  - Conventions
  - Structure/organization
  - Major classes
- Use FloppyTail as an example

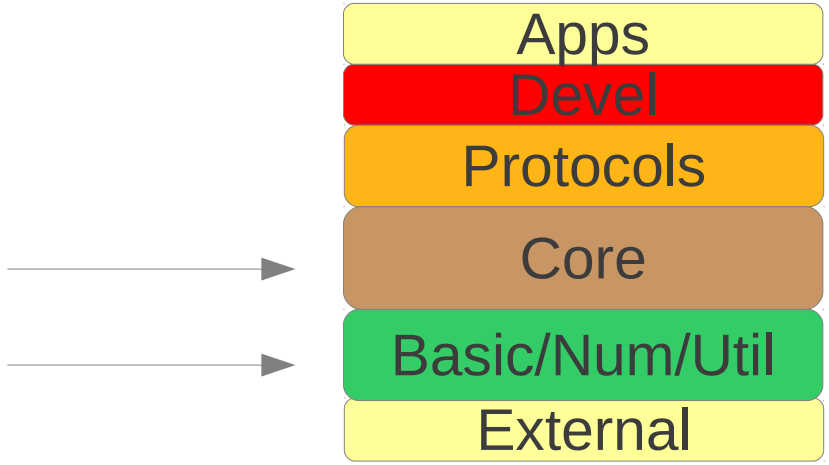
Rosetta has layers



- Layers = libraries
- Organization
- Enforcement:
  - Directory structure
  - Namespacing
  - SCons building

5 patties
20 slices of cheese

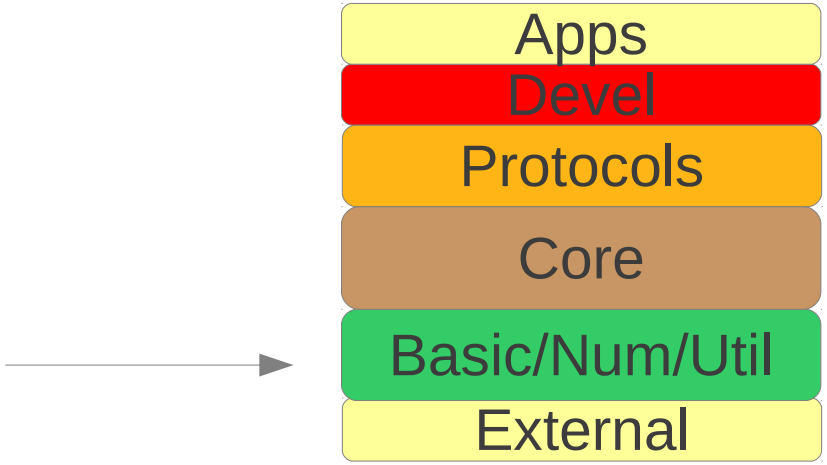
Basic conventions



- Owning pointers
  - OP, COP
  - Inherit from `ReferenceCount`
- `vector1`
  - Index from 1, not 0
  - Bounds check <=
- `Core::Size` (uint)
- `Core::Real` (double)

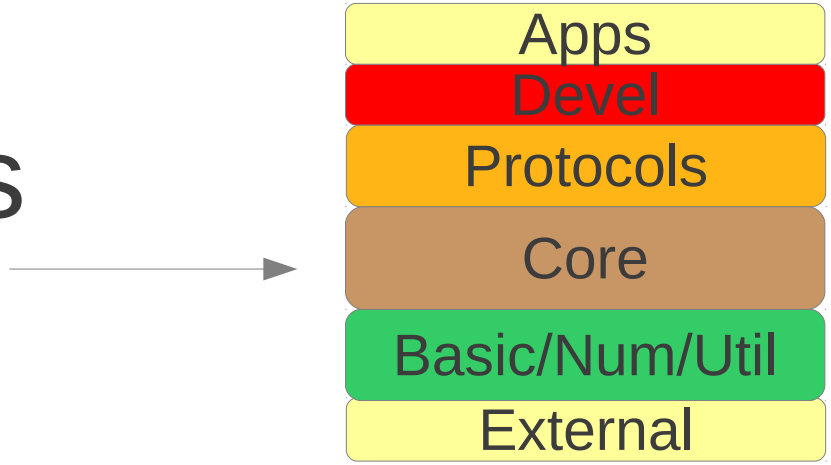
- `Tracer`
  - Replaces `std::cout`
  - Labels output
  - Provides mute control

Low-level libraries



- Owning pointers
- `vector1`
- `Tracer`
- `option` object
  - Holds values from commandline options
  - `options_rosetta.py`

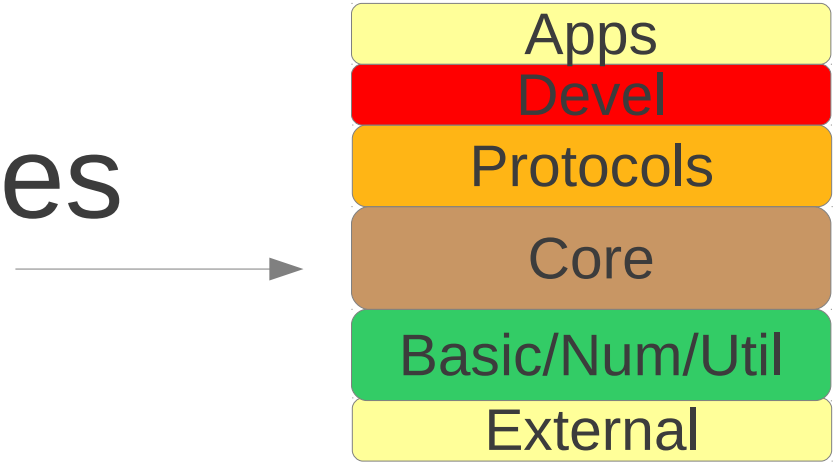
Core::chemical classes



- Abstract representations
- `AtomType`
- `ResidueType`
  - Defines what atoms are in a residue (or ligand)
  - How they connect internally

- Variant system
  - C-terminal OXT atom
- `ResidueTypeSet`
- `ChemicalManager`
  - Singleton
  - Single read of database

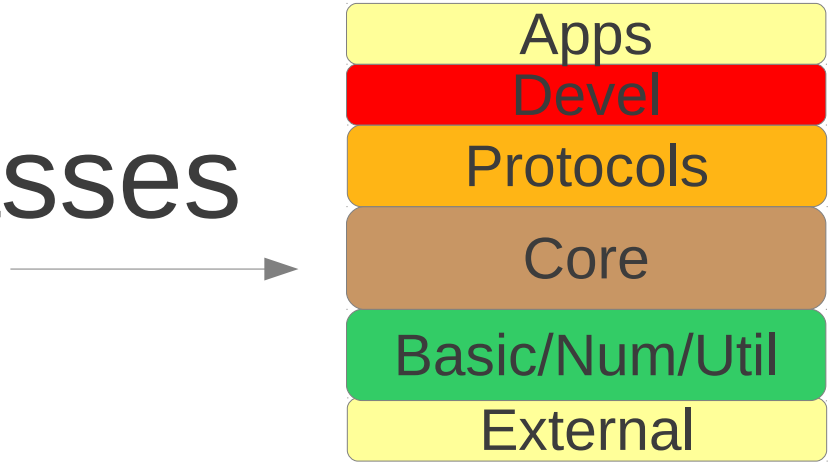
Core::kinematics classes



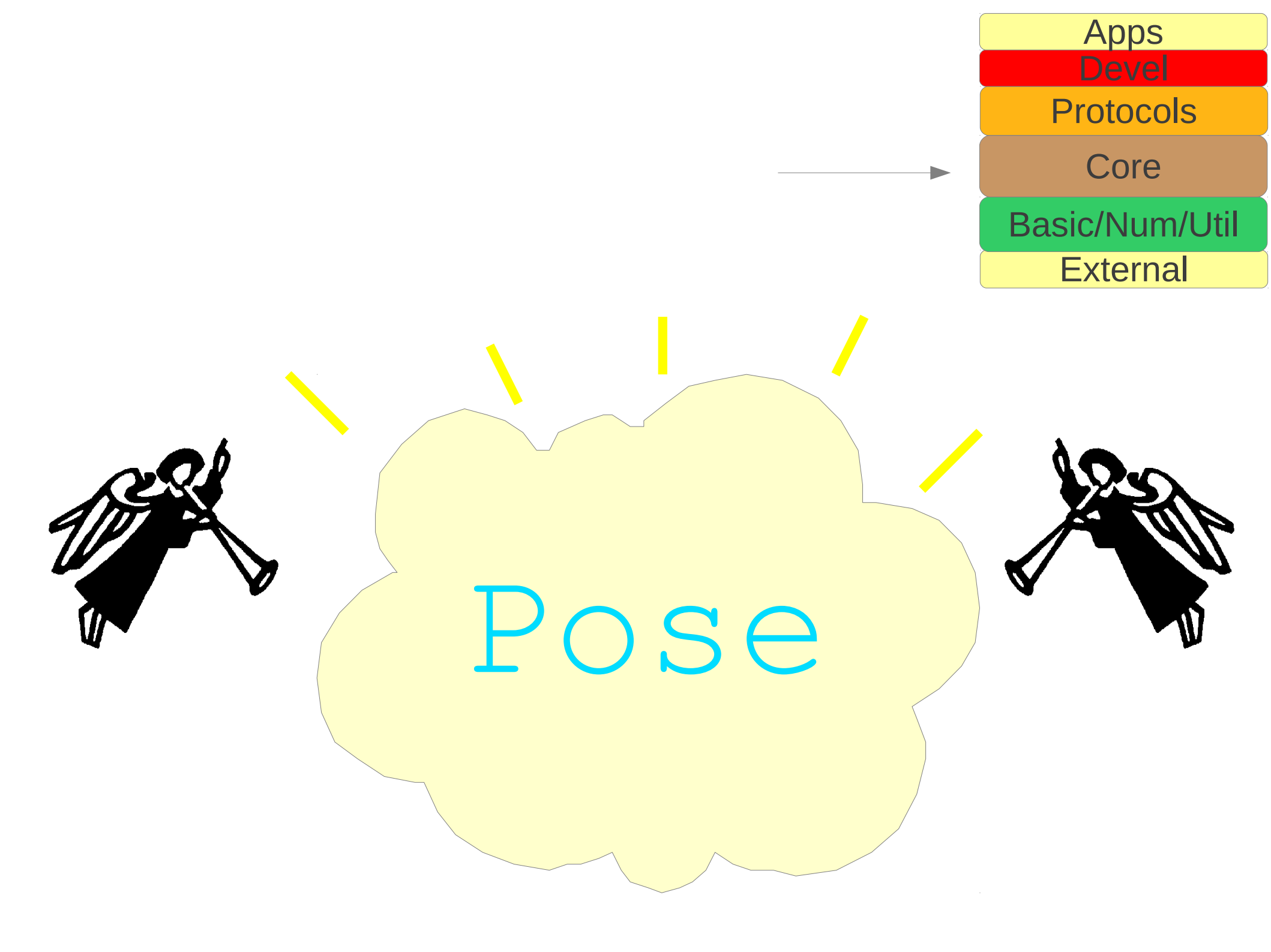
- `AtomTree`
  - Defines atomic connectivity
  - Internal → (x, y, z)
- `FoldTree`
  - Defines residue connectivity
  - Human interface

- `MoveMap`
  - Contains lists of mobile, immobile degrees of freedom

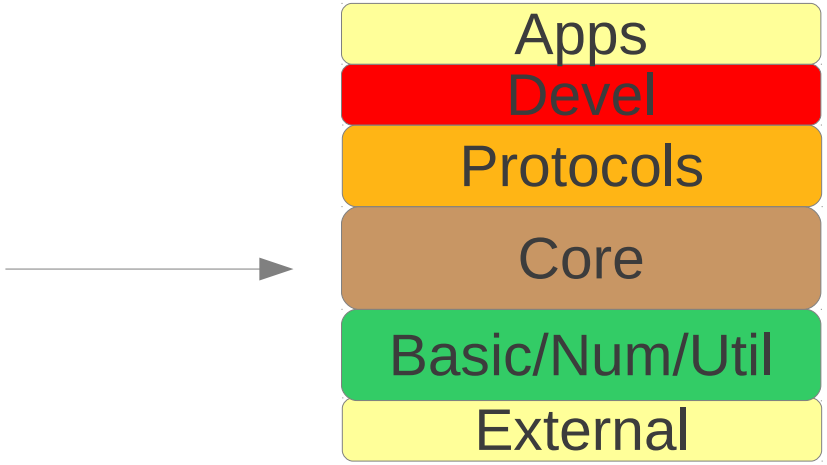
Core::conformation classes



- Abstract chemical + kinematic layers
  - Concrete, distinct
- `Atom`
  - Gives an (x, y, z) to an `AtomType`
- `Residue`
  - Puts `Atom` objects on `ResidueType` skeleton
- `Conformation`
  - Contains `Residue` objects
  - Linked by kinematic layer to describe internal-coordinate folding


Apps
Devel
Protocols
Core
Basic/Num/Util
External
Pose

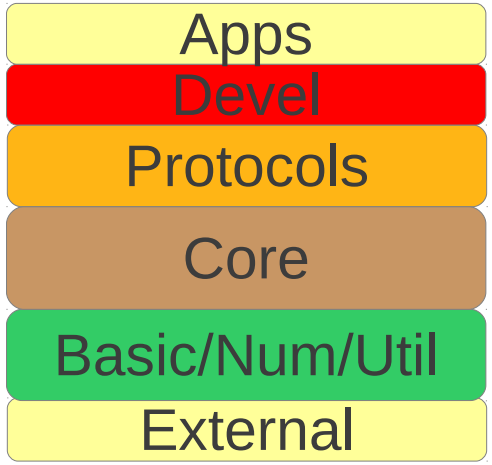
Core::scoring classes



- `Energies`
  - Caches scores, lives in `Pose`
- `ScoreFunction`
  - Scores
  - OPs to `EnergyMethod`s
- `EnergyMethod`
  - Scoring terms
  - Many & varied

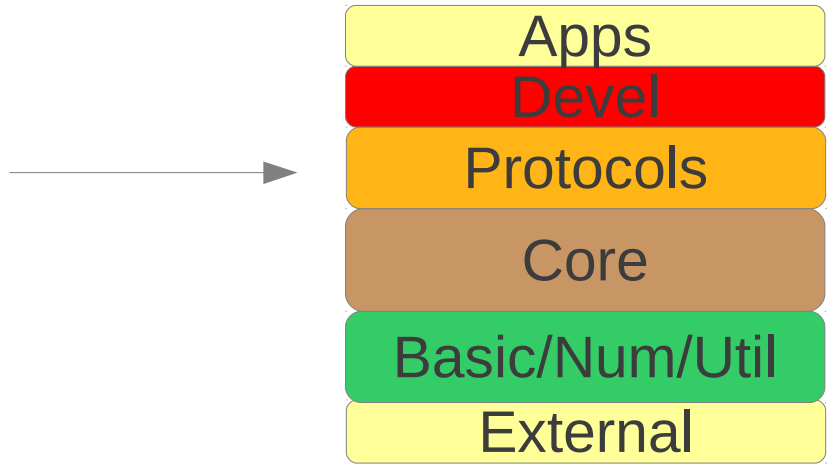
- `ScoringManager`
  - Singleton!
  - Single read of database

Core::pack & ::optimization



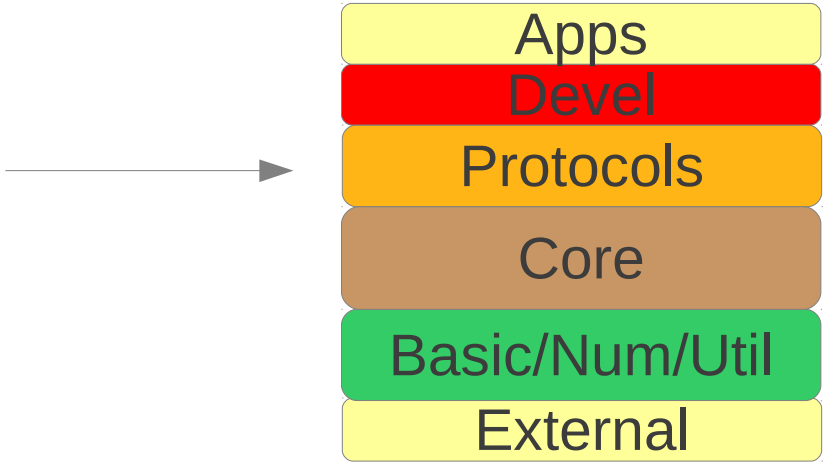
- Guts of packing and minimization
- Very little direct use – almost all through protocols layer
- `PackerTask`
  - Set up what's allowed in packing
  - Disposable
- `TaskFactory`
  - Set up new `PackerTask`s as needed
  - uses `TaskOperation`s

Protocols layer



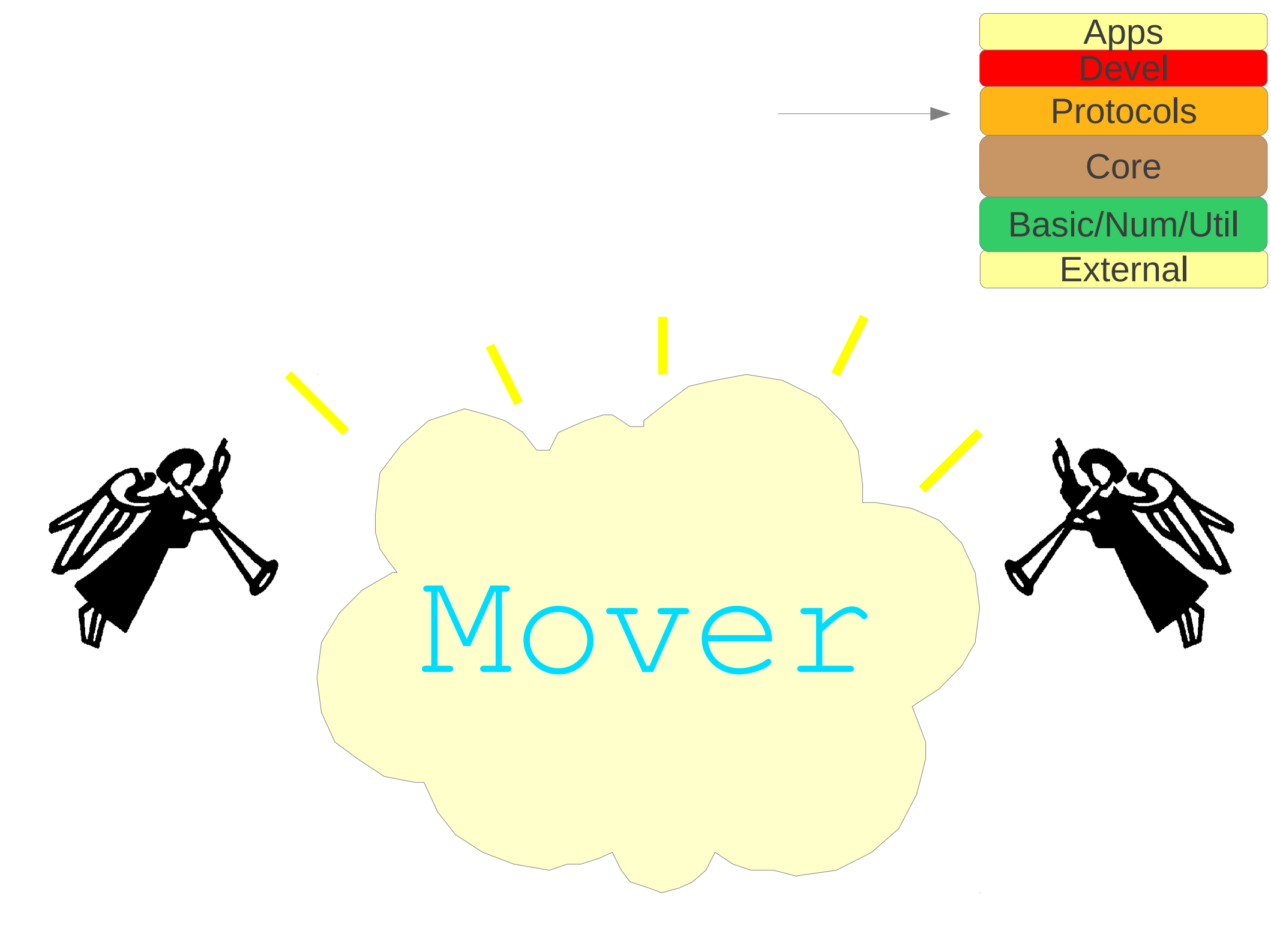
(protocols graph)

Protocols layer

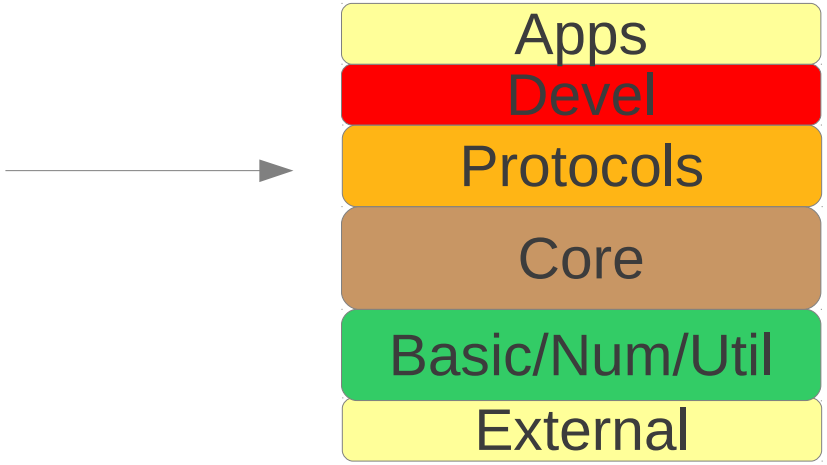


- `MonteCarlo`
  - Tracks trajectory
- Job distribution
  - 1UBQ_0001.pdb ...
  - Communication layer for MPI
  - New system in Rosetta 3.1 from 3.0
- Init
  - Load time factory reg.

- Gobs of protocols
  - Abinitio
  - Loops
  - Relax
  - Fixbb
- `Mover`...


Apps
Devel
Protocols
Core
Basic/Num/Util
External
Mover

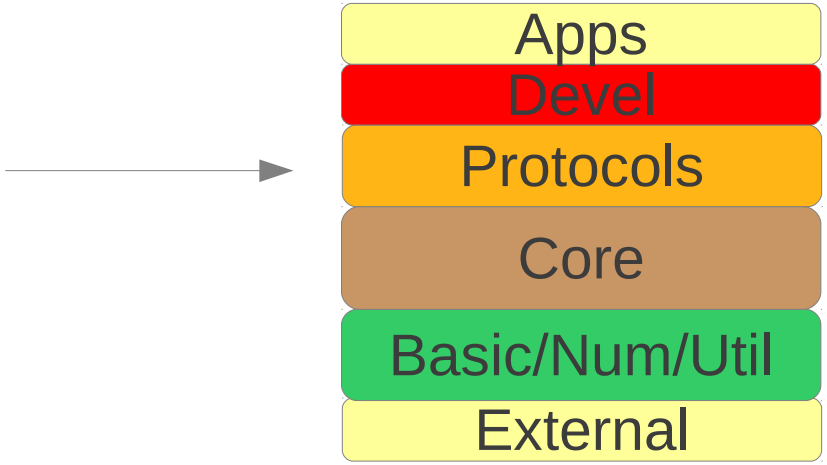
Movers



- `Mover`...
  - Beloved workhorse
  - Centralizes `Pose` alteration
  - Movers can call other movers
    - A protocol is born
  - `virtual void apply( Pose & )`

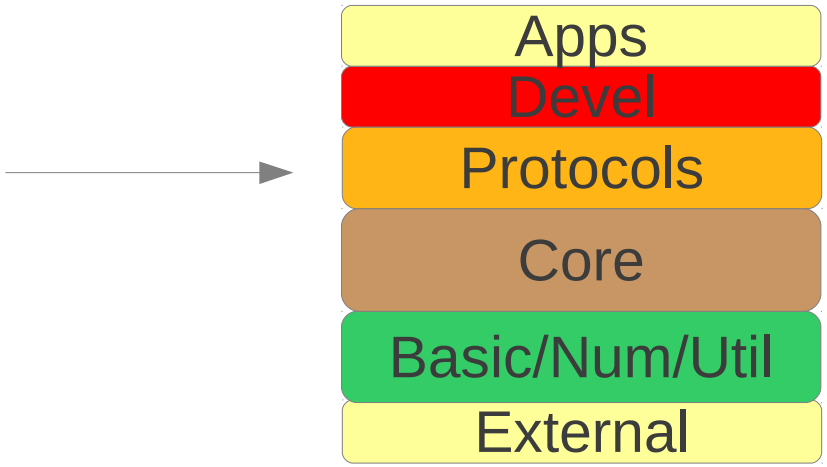
- Simple modifiers
- Empty boxes
- Not-really-a-Mover
- Whole protocols

Simple Movers



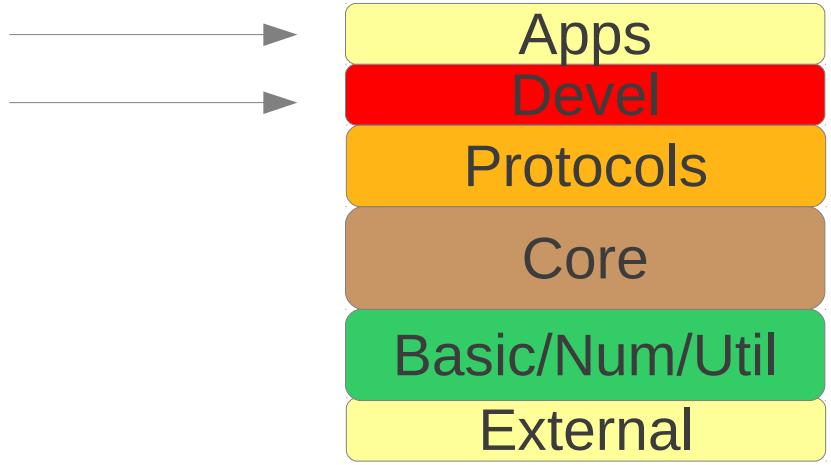
- Traditional R++ functions
- Packing
  - `PackRotamersMover`
  - `RotamerTrialsMover`
- Backbone movement
  - `SmallMover`
  - `ShearMover`
  - Fragment movers
- Minimization
  - `MinMover`

Other Movers

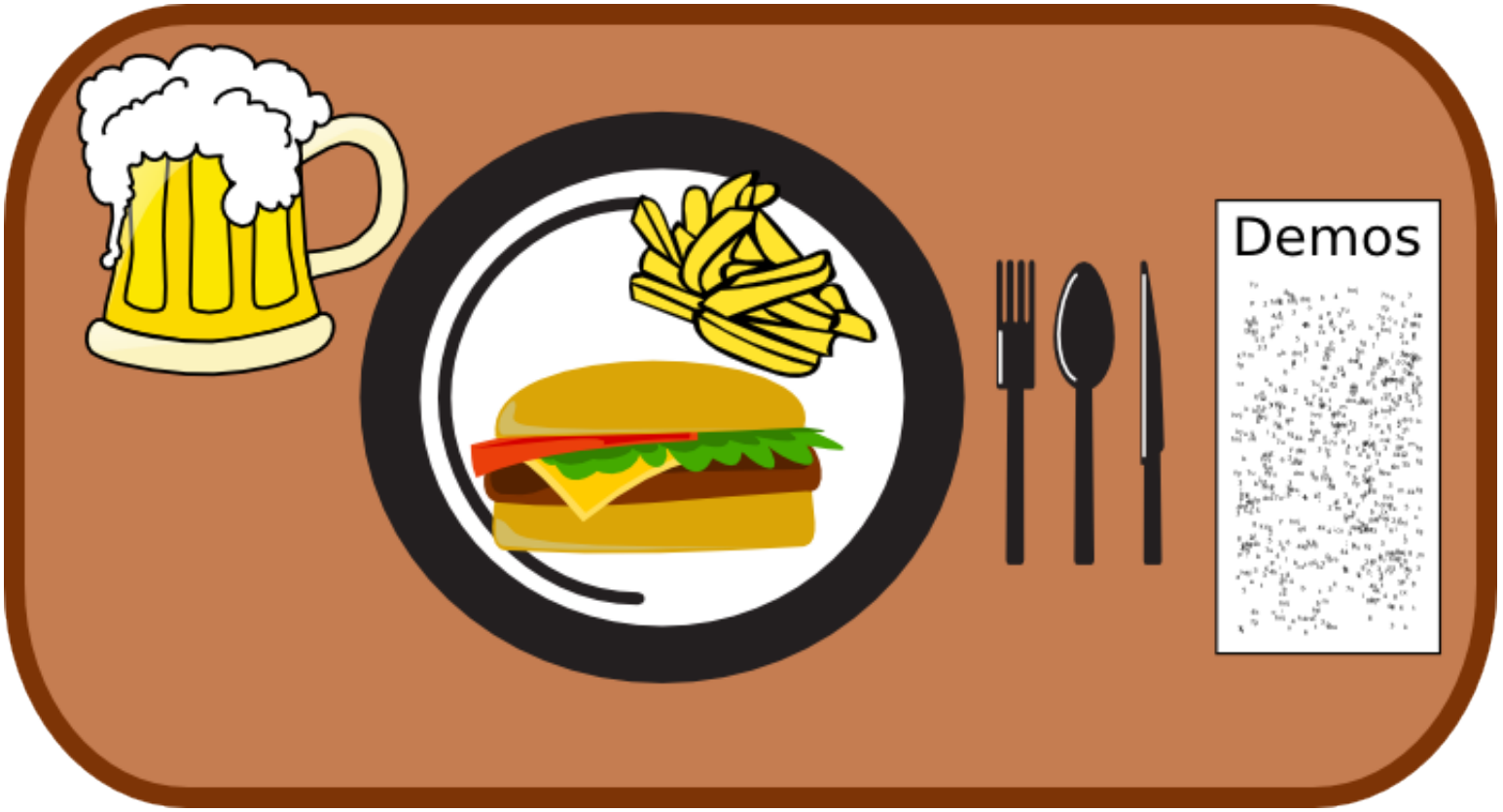


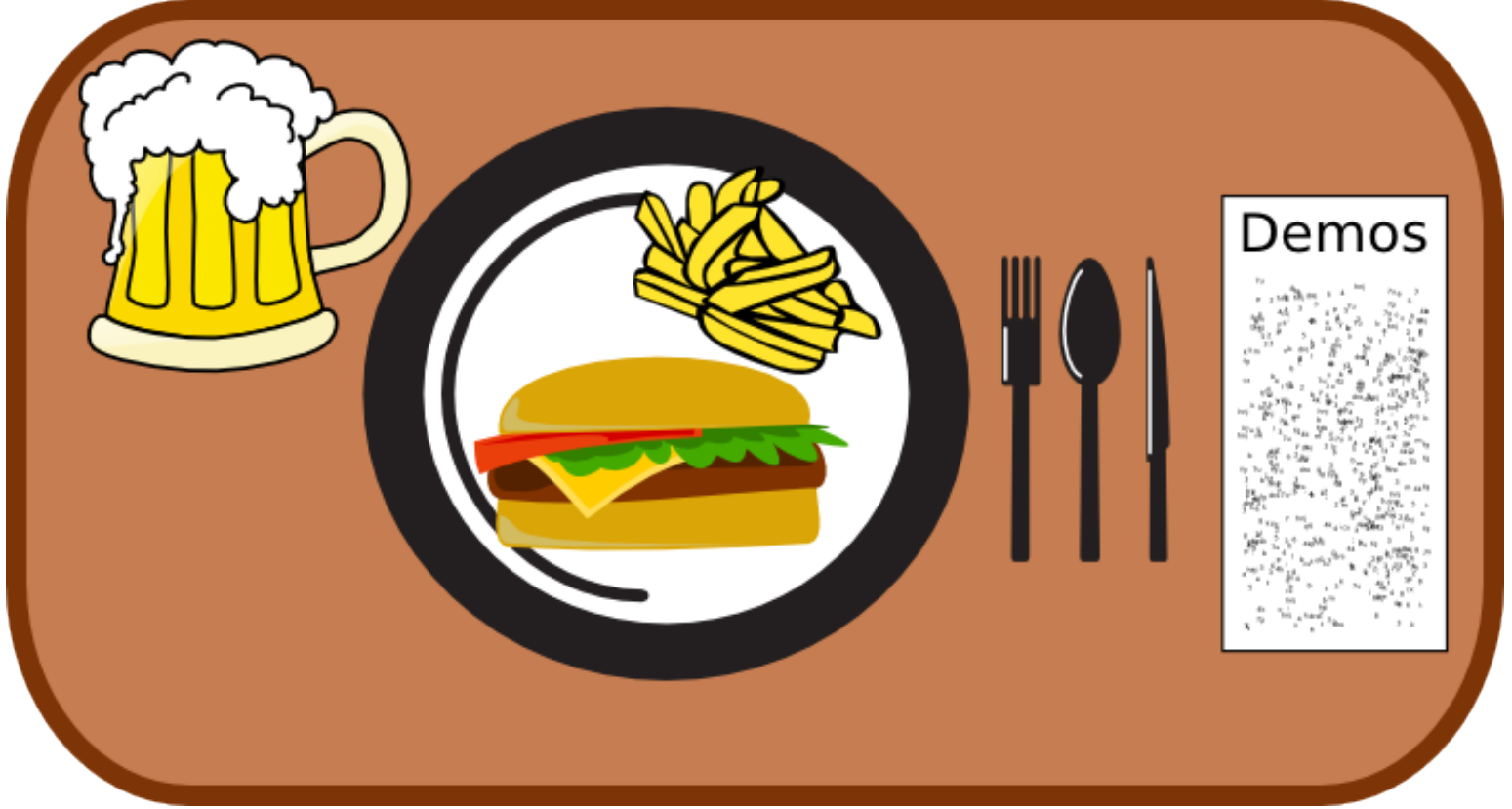
- Whole protocols
  - Job distributor runs a `MoverOP`
  - Allows your protocol to call others
  - Pushes complexity out of Apps layer, into Protocols (reuse)

Upper layers



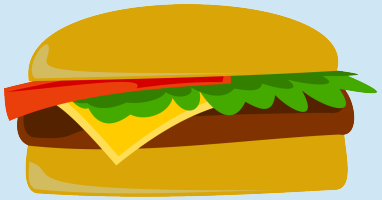
- Devel
  - Not present in public releases
  - Code under development, not for general use
- Apps
  - Not a library: just executables
  - Pilot not released (under development)
  - Public released


Demos



main

database



source

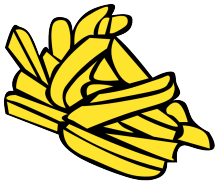
tests

tools

demos

Database

- Raw data for Rosetta
- `.params` files become `ResidueType`s
- Weights used to populate `ScoreFunction`s
- Phsyical parameters to calculate `EnergyMethod`s



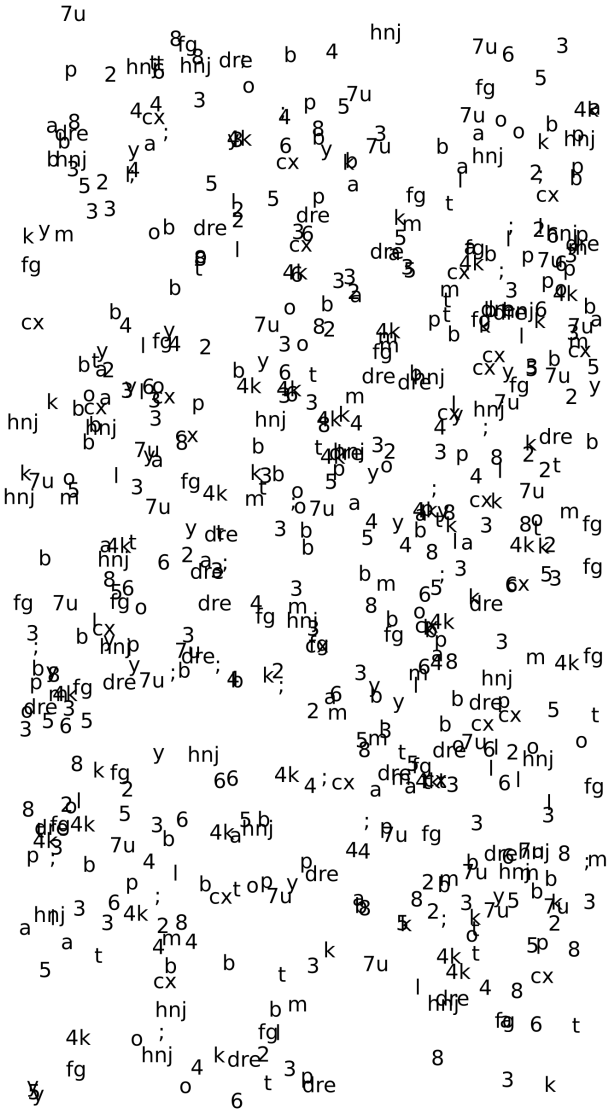
Tests

- Integration tests (!)
  - Ultra easy test framework
  - Inferior for some things, great for others
  - Make them! Use them!
- Performance/Scientific/etc. tests
- NOT the unit tests (those were in source)
  - Very granular, useful within development

Tools 

- Mostly utility scripts
  - Working with code
  - Working with Rosetta data types
    - fragments

Demos



- Lots of examples
- Pay the bills: write your own as you learn!