Actors

What, Why, and How

What Is an Actor?

Actor Model

- http://en.wikipedia.org/wiki/Actor_model
- The Actor model is a mathematical model of concurrent computation that treats "actors" as the universal primitives of concurrent digital computation.

An Actor Is a Process

An Actor Can Change It's Own State

An Actor Can Create Another Actor and Get it's Address

An Actor Can Send a Message To Any Addresses It Knows

An Actor Can Wait for a Specific Message to Arrive in it's Mailbox

Why Use Actors?

Isolation

- Only an Actor can change it's own state, simplifying logic
- Locking not required
- Potential for race conditions reduced

Simple Control Flow

- Each Actor's control flow is independent
- Code can be written straight line or with simple loops

Message Passing

- Easy to distribute
  - Across cores
  - Across UNIX process boundaries
  - Across machines

Simplified Error Handling

- Most exceptional conditions occur while waiting for a message
  - Timeouts
  - Network errors
- Isolates error handling code
- Makes it easier to build fault tolerant systems

How Could We Implement Actors?

A dissection of a prototype I built in Python

http://bitbucket.org/fzzzy/python-actors/

Existing Actor Systems

- Erlang http://erlang.org/
- Io http://www.iolanguage.com/
- Python
  - PARLEY http://osl.cs.uiuc.edu/parley
  - Dramatis http://pypi.python.org/pypi/dramatis
  - Candygram http://candygram.sourceforge.net/

I Built My Own
...for science!

http://bitbucket.org/fzzzy/python-actors

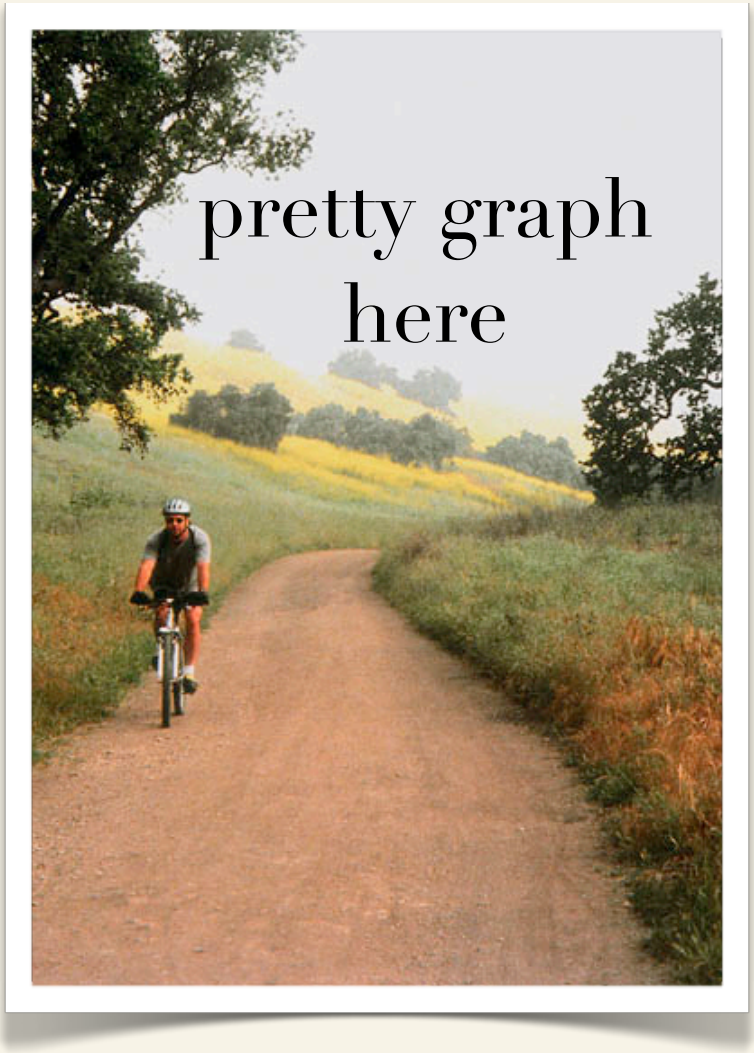
What Did I Choose?

- Green Threads for "Processes"
  - Using eventlet
    - Which uses greenlet
- JSON for message serialization
  - Used to copy messages between in-process actors
- WSGI/HTTP for network protocol

Green Threads?

- Why not:
  - POSIX Threads
  - OS Processes

- Speed
- Memory Usage



Spawning an Actor

code here

Message Copying

- Messages are serialized into JSON
- Messages to in-process actors are thus copied
  - Preserves isolation
- JSON is ready to go over the network

Sending a Message

code here

Pattern Matching

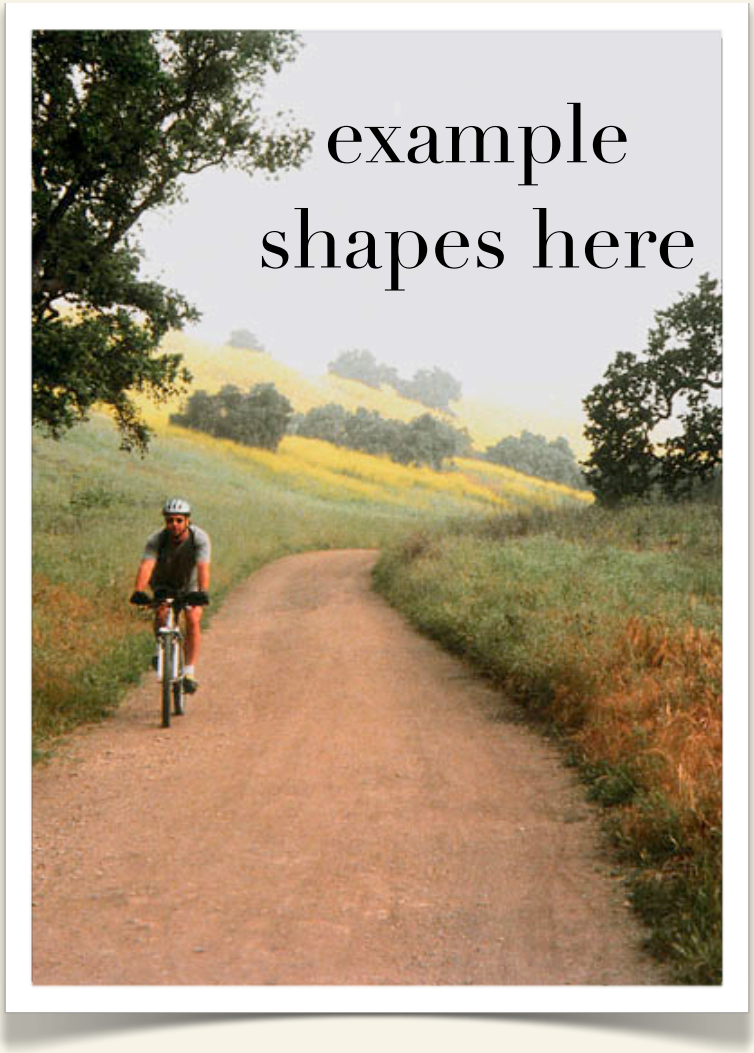
- "An Actor Can Wait for a Specific Message to Arrive in it's Mailbox"
  - In Erlang this is called "Selective Receive"
  - Also known as "Pattern Matching"
  - Python doesn't have this
    - My implementation is called "shaped"

Receiving a Message

code here

Possible Message Contents

- dict
- list
- tuple
- set
- string
- unicode
- int
- float



Network Protocol

- I chose:
  - HTTP
    - wsgi application
  - JSON
  - REST

REST Interface

- PUT spawns an actor
- POST sends a message to actor
- GET gets the current state of the actor
- DELETE sends a Killed exception to the actor

Problem

Python modules contain global state

Module Global Problem

- Possibility:
  - Keep a unique copy of sys.modules for every actor
  - Seal modules in wrapper objects to prevent modification
- Reality:
  - Just don't use global module state

Q&A

Tell me what I don't know about Actors and Python