MongoDB

Robert M. Dondero, Ph.D
Princeton University

Objectives

- The lecture will:
  - Provide *a taste of* the **MongoDB** DBMS
  - Give you enough information about **MongoDB** to:
    - Help you decide if you want to use it in your project
    - Help you get started with it

MongoDB

- ***MongoDB***
  - **Who**: 10gen, later MongoDB Inc.
  - **When**: 2007
  - **Why**:
    - Document-oriented database
    - Originally: part of a cloud-as-service platform
    - Later: open source flagship product

Motivation

- Why study MongoDB?
  - Popular
    - The most popular **document store** DBMS
    - And more generally…
    - The most popular non-relational (**NoSQL)** DBMS
  - Different
    - Contrasts with relational DBMSs
  - Reasonable for COS 333 project
    - Especially if storing data that is hierarchical to arbitrary depths

Installing MongoDB

- Linux (Debian, Ubuntu)
  - At a bash shell prompt:
    - `sudo apt-get mongodb`

Installing MongoDB

- Mac
  - Install Homebrew
    - Browse to https://brew.sh/ and follow instructions
  - At a bash shell prompt:
    - `brew tap mondodb/brew`
    - `brew install mongodb-community@4.2`

Installing MongoDB

- MS Windows
  - Browse to http://mongodb.com/try/download/community
  - Click on Download
  - In Windows Explorer, double click on the .msi file that was downloaded to launch the installer
  - Use the default installation settings

Installing MongoDB

- MS Windows (cont.)
  - Create a data directory
    - `mkdir C:\data\db`
  - Add `C:\Program Files\MongoDB\server\4.4\bin` to your `PATH` environment variable

Starting the MongoDB Server

- Linux (without transaction support)
  - Starts automatically at login

Starting the MongoDB Server

- Mac (without transaction support)
  - At a bash shell prompt
    - `mongod –config /usr/local/etc/mongod.conf`

Starting the MongoDB Server

- Mac (with transaction support)
  - At a bash shell prompt
    - `mongod -config /usr/local/etc/mongod.conf -replSet r1`
  - At another bash shell prompt:
    - `mongo`
  - In the resulting MongoDB shell
    - `rs.initiate()`

Starting the MongoDB Server

- MS Windows (without transaction support)
  - At a command prompt:
    - `mongod`

Starting the MongoDB Server

- MS Windows (with transaction support)
  - At a command prompt:
    - `mongod -replSet r1`
  - At another command prompt:
    - `mongo`
  - In the resulting MongoDB shell
    - `rs.initiate()`

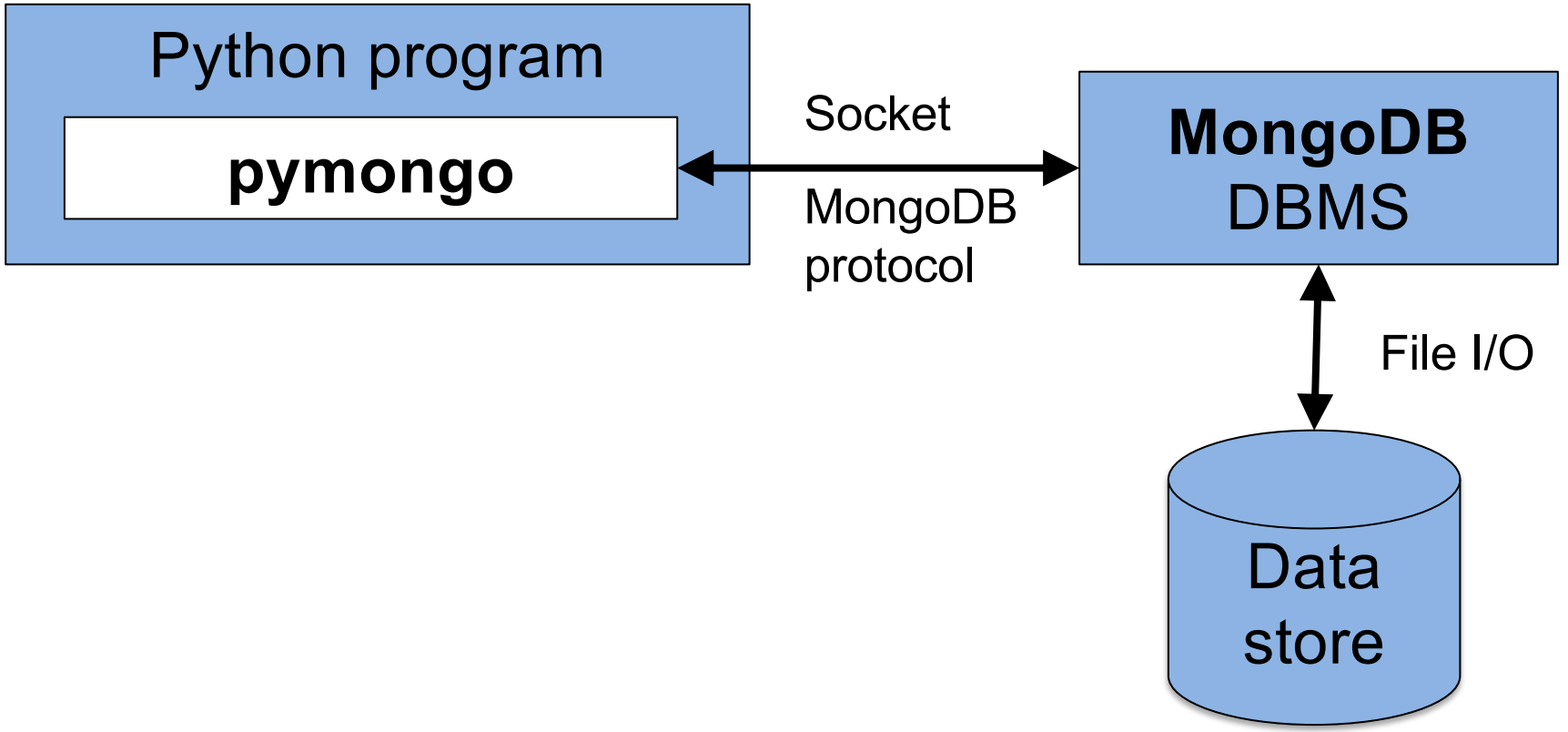
Starting the MongoDB Server

- DBMS listens for connections at a default port 27017
- MongoDB supports authentication, but not by default

The MongoDB Shell

- Linux, Mac, and MS Windows:
  - At bash shell / command prompt:
    - `mongo`
    - (Enter JavaScript statements)
    - `exit`

MongoDB via Python



Installing the MongoDB Driver

- Linux, Mac, and MS Windows
  - At a bash shell prompt:
    - Activate your cos333 virtual environment
    - `python -m pip install pymongo`

The MongoDB Data Model

- The MongoDB data model

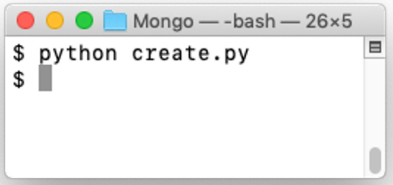
Relational Model	MongoDB
Database	Database
Table	Collection
Row	**Document**

MongoDB Pgmming in Python

- ***Document***
  - **Internally**:
    - A fragment of ***JavaScript Object Notation (JSON)***
    - (JavaScript and JSON covered later in course)
  - **Externally** in Python:
    - A primitive object (`int`, `float`, `bool`, `str`)
    - Or a `dict` object containing `dict` or `list` or primitive objects,
    - Or a `list` object containing `dict` or `list` or primitive objects
    - With nesting to any level

MongoDB Pgmming in Python

- See **Mongo/create.py**



  - DBMS runs as a distinct process on a specified computer (localhost)
  - DBMS listens for connections at a known port (27017)

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
  - Create client
  - Index into client to get database

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
  - Index into pennyDB to get bookCollection
    - Corresponds to combination of books and authors tables

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
  - zipcodeCollection
    - Corresponds to zipcodes table
    - `insert_one()` returns a result
    - Result contains an `inserted_id` field
    - `inserted_id` field is document reference

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
  - customerCollection
    - Corresponds to customers table
    - Contains references to documents in zipcodeCollection

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
  - orderCollection
    - Corresponds to orders table

MongoDB Pgmming in Python

- Data relationships: books and authors
  - Relational DB
    - Book data is stored in books table
    - Author data is stored in authors table
    - Each rows in books table is related to row(s) in authors table by isbn
  - MongoDB
    - Book data is stored in bookCollection
    - Author data is merged into bookCollection
    - Each book document contains a list/array of authors
    - Relationship is implemented via **denormalization**

MongoDB Pgmming in Python

- Data relationships: customers and zipcodes
  - Relational DB
    - Customer data is stored in customers table
    - Zipcode data is stored in zipcodes table
    - Each row in customers table is related to a row in zipcodes table by zipcode
  - MongoDB
    - Customer data is stored in customerCollection
    - Zipcode data is stored in zipcodeCollection
    - Each customer document is related to a zipcode document by document reference
    - Relationship is implemented via **references**

MongoDB Pgmming in Python

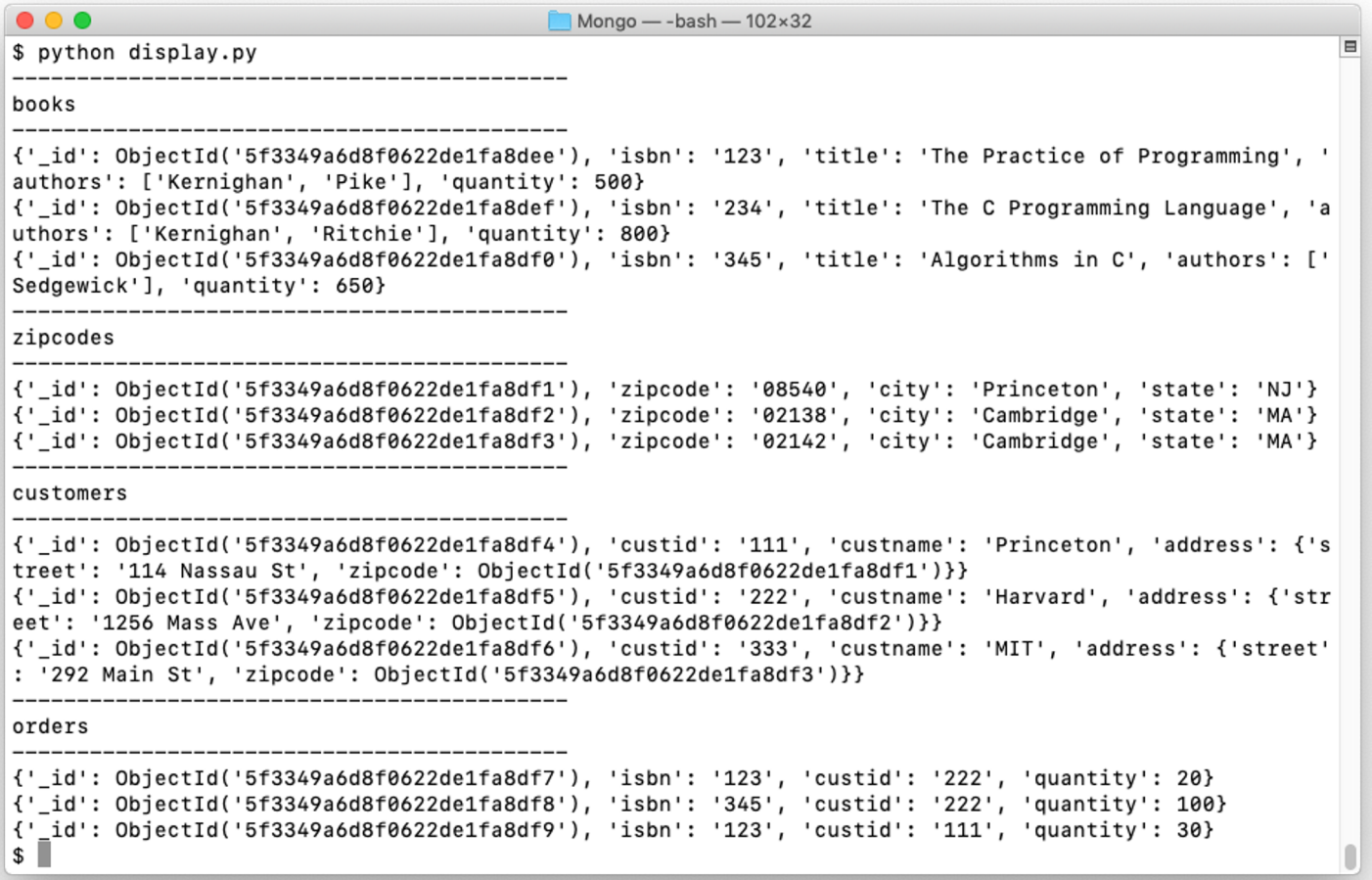
- Data relationships: books, customers, orders
  - Relational DB
    - Book data is stored in books table
    - Customer data is stored in customers table
    - Order data is stored in orders table
    - Each row in books is related to row(s) in orders by isbn
    - Each row in customers is related to row(s) in orders by isbn
  - MongoDB
    - Book data is stored in bookCollection
    - Customer data is stored in customerCollection
    - Order data is stored in orderCollection
    - Each document in bookCollection is related to document(s) in orderCollection by isbn
    - Each document in customerCollection is related to document(s) in orderCollection by isbn
    - Relationship is implemented via **data**

MongoDB Pgmming in Python

- Data relationships in MongoDB:
  - **Option 1**: denormalization
  - **Option 2**: references
  - **Option 3**: data

MongoDB Pgmming in Python

- See **Mongo/display.py** (cont.)

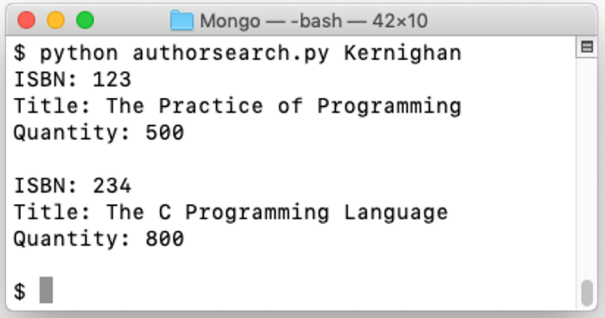


MongoDB Pgmming in Python

- See **Mongo/display.py** (cont.)
  - `find()` returns an iterable object
  - Each iteration yields a document
  - Each document is a Python dict
  - Each document has a unique `_id`
  - Each customer document references a zipcode document

MongoDB Pgmming in Python

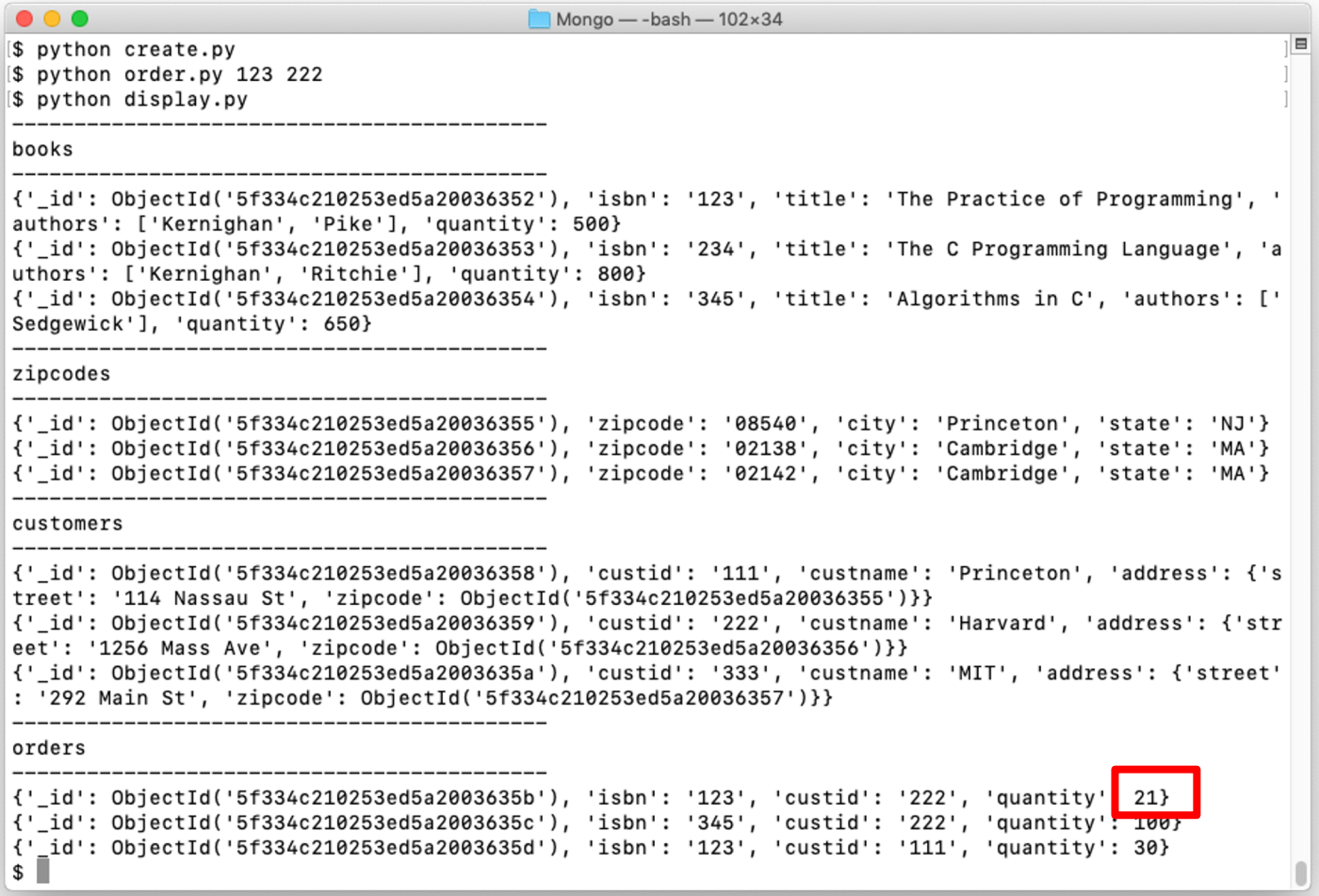
- See **Mongo/authorsearch.py**



  - `collection.find()` method
  - MongoDB supports search through documents

MongoDB Pgmming in Python

- See **Mongo/order.py**

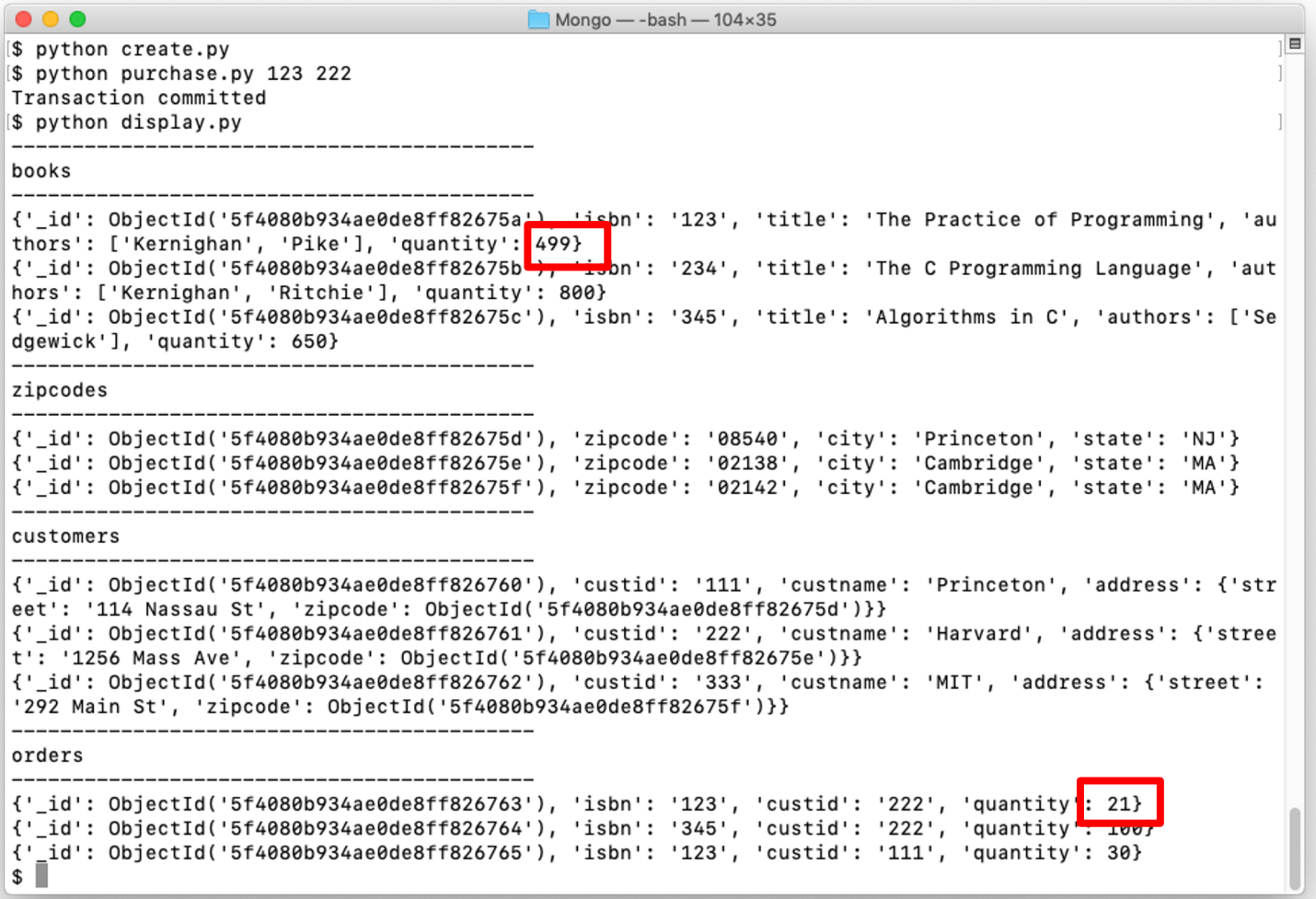


MongoDB Pgmming in Python

- See **Mongo/order.py** (cont.)
  - `collection.update_one()` method
  - MongoDB supports complex queries and various kinds of updates

MongoDB Pgmming in Python

- See **Mongo/purchase.py**

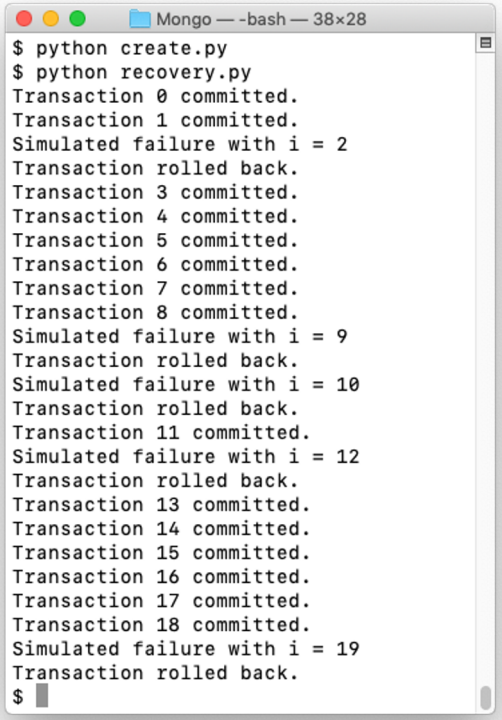


MongoDB Pgmming in Python

- See **Mongo/purchase.py** (cont.)
  - MongoDB 4.0 and newer supports transactions

MongoDB Pgmming in Python

- See **Mongo/recovery.py**



MongoDB Pgmming in Python

- See **Mongo/recovery.py** (cont.)
  - Transactions work!

Stopping the MongoDB Server

- Linux
  - Stops automatically at logout
- Mac
  - At a bash shell prompt
    - Type Ctrl-c to abort `mongod` command
- MS Windows
  - At a command prompt
    - Type Ctrl-Fn-Pause to abort `mongod` command

MongoDB Assessment

- MongoDB assessment
  - (pro) Fast
    - (In old versions) no transactions => no locking, no commits, no rollbacks
  - (pro) Simple
    - No table/collection joins
  - (pro) Flexible data model
    - Good for storing hierarchical data
    - No (actually, optional) schemas
  - (pro) Uses OO model
    - No impedence mismatch

MongoDB Assessment

- MongoDB assessment (cont.)
  - (con) Less robust
    - (In old versions) no transactions => no locking, no commits, no rollbacks
  - (con) Maybe less convenient
    - No table/collection joins => extra logic in client
  - (con) Less data integrity
    - No (actually, optional) schemas

Summary

- The lecture has:
  - Provided *a taste of* the **MongoDB** DBMS
  - Given you enough information about **MongoDB** to:
    - Help you decide if you want to use it in your project
    - Help you get started with it