Materials databases

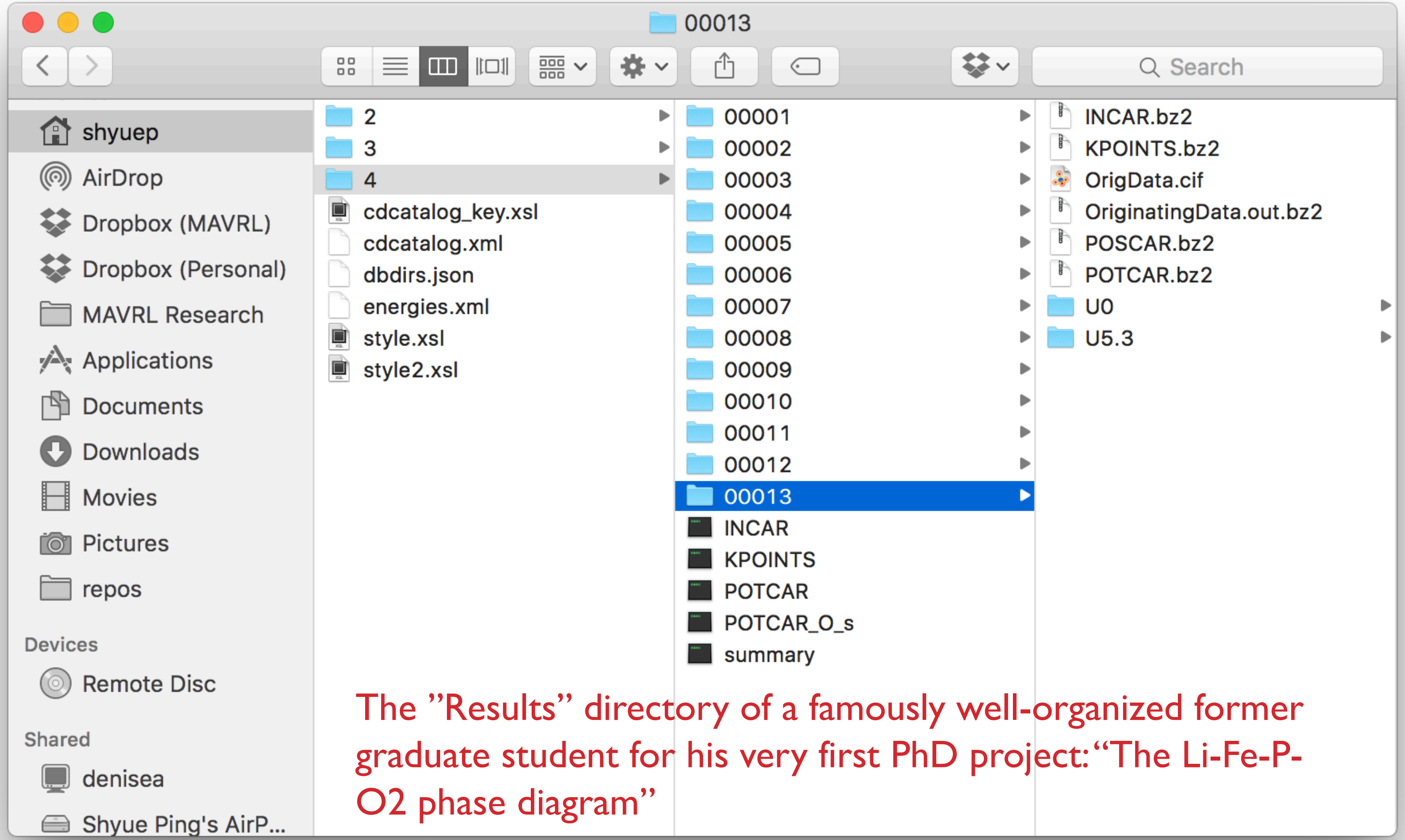
Shyue Ping Ong

August 16, 2016

UC San Diego
Jacobs School of Engineering

materials virtual lab

Why do I need a database?



2
3
4
cdcatalog_key.xsl
cdcatalog.xml
dbdirs.json
energies.xml
style.xsl
style2.xsl

00001
00002
00003
00004
00005
00006
00007
00008
00009
00010
00011
00012
00013
INCAR
KPOINTS
POTCAR
POTCAR_O_s
summary

INCAR.bz2
KPOINTS.bz2
OrigData.cif
OriginatingData.out.bz2
POSCAR.bz2
POTCAR.bz2
U0
U5.3

The ”Results” directory of a famously well-organized former graduate student for his very first PhD project: “The Li-Fe-P-O2 phase diagram”

Why do I need a database?

Store pertinent results and analyses

- Anything more than O(10) calculations, a DB is useful.
- Some analyses are expensive (e.g., AIMD or charge density)

Sharing

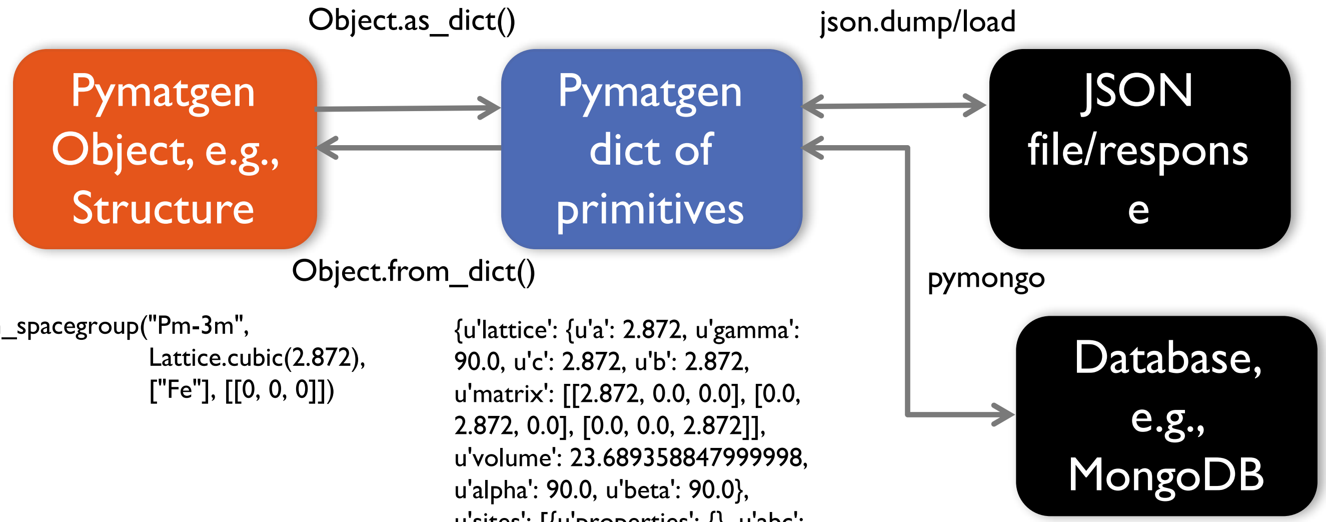
- Dropbox is great, but not really a long-term solution for sharing lots of numeric data.
- Pre-requisite for building web applications for team.

Querying and analysis

- Years into the future. Even today, MP can recall data calculated during the dark ages of 2011.

The as_dict() and from_dict() protocol

Almost all non-trivial objects in pymatgen support the as_dict() and from_dict() serialization protocol.



Object.as_dict()

json.dump/load

Pymatgen Object, e.g., Structure

Pymatgen dict of primitives

JSON file/respons e

Object.from_dict()

pymongo

Database, e.g., MongoDB

```
Structure.from_spacegroup("Pm-3m",
                          Lattice.cubic(2.872),
                          ["Fe"], [[0, 0, 0]])
```

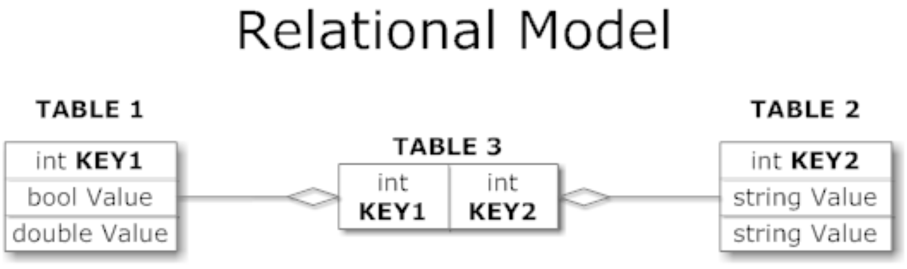
```
{u'lattice': {u'a': 2.872, u'gamma':
90.0, u'c': 2.872, u'b': 2.872,
u'matrix': [[2.872, 0.0, 0.0], [0.0,
2.872, 0.0], [0.0, 0.0, 2.872]],
u'volume': 23.689358847999998,
u'alpha': 90.0, u'beta': 90.0},
u'sites': [{u'properties': {}, u'abc':
[0.0, 0.0, 0.0], u'xyz': [0.0, 0.0,
0.0], u'species': [{u'occu': 1,
u'element': u'Fe'}], u'label': u'Fe'}],
u'@class': 'Structure',
u'@module':
'pymatgen.core.structure'}
```

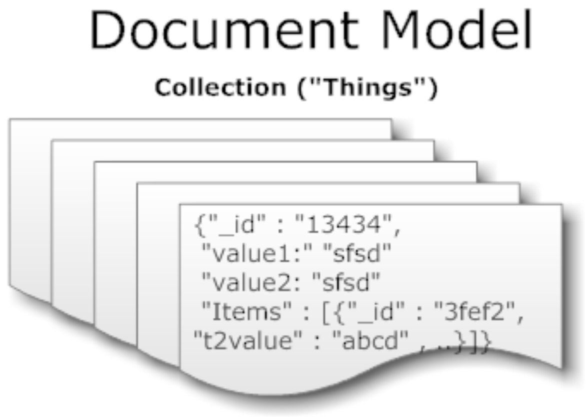
What is MongoDB?

MongoDB is an open-source NoSQL document database.

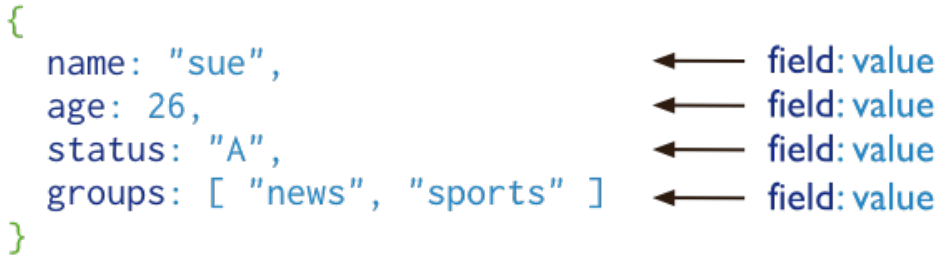
- JSON-style documents with dynamic schemas offer simplicity and power.
- Full Index Support
- Replication & High Availability by mirroring across LANs and WANs for scale and peace of mind.
- Rich, document-based queries.
- Fast In-Place Updates
- Map/Reduce for flexible aggregation and data processing.

Document-based vs SQL databases





```
{"_id" : "13434",
 "value1:" "sfsd"
 "value2: "sfsd"
 "Items" : [{"_id" : "3fef2",
"t2value" : "abcd" , ...}]}
```



```
{
   name: "sue",
   age: 26,
   status: "A",
   groups: [ "news", "sports" ]
}
```

How is MongoDB used in the Materials Project?

MongoDB databases contains collections of documents.

Each logical unit of calculation or analysis or material is a document in a collection.

Collection name	Document description
tasks	Results of a single first principles calculation, be it a relaxation, static or bandstructure calculation
materials	Summary of computed information about a material formed by aggregating all tasks of a crystal structure.
(bandstructures, phasediagrams, batteries, etc.)	Documents representing results of various types of analyses. E.g., a battery document is formed by combining several materials (e.g., FePO4 and LiFePO4) to represent an intercalation system with associated properties like voltage, capacity, etc.

MongoDB resources

Official MongoDB docs and tutorials:
https://docs.mongodb.com

How do you interact with MongoDB?

- ”mongo” shell command
- Pymongo (Python Mongo Driver)

Pymatgen-db (http://pythonhosted.org//pymatgen-db/)

Database add-on for pymatgen. Enables the creation of Materials Project-style MongoDB (www.mongodb.org) databases for management of materials data. Key features:

- Query engine for easy translation of MongoDB docs to useful pymatgen objects for analysis purposes.
- Includes a clean and intuitive web ui (the Materials Genomics UI) for exploring Mongo collections.

Pymatgen-db is **primarily used for the generation of the "tasks" collection**, i.e., it parses VASP calculations, convert the results into a pymatgen dict, and inserts it into a MongoDB collection as a document. It also facilitates the retrieval of these documents as pymatgen objects.

How does it help you?

If you are only doing a few calculations, it is probably not worth your time to deal with the overhead of learning MongoDB, pymatgen-db, etc.

However, if you perform hundreds or thousands of similar calculations on different materials, and you need some proper way to manage and analyze this data (note: compiling calculations in a spreadsheet does not constitute proper data management!), it is well worth taking the time to learn MongoDB and pymatgen-db.

Basic tutorial using the command line

A quick way of playing around with pymatgen-db without going through the hassle of installing MongoDB on your own machine is to sign up for a free MongoLab (https://mongolab.com) “sandbox” account. The default 512Mb should be sufficient for you to insert a few VASP calculations and play around with some queries to decide if this is suitable for your needs. If you decide to get serious, it is highly recommended that you go through the MongoDB tutorial to learn how to use MongoDB effectively. There is a learning curve, but it rapidly pays dividends when you have to manage a lot of calculations.

The mgdb CLI

Pymatgen-db has a comprehensive command-line tool called mgdb.

- Perform many common tasks without ever looking at the pymatgen-db source code or writing a single piece of code.

After installing pymatgen-db, you can call up the help by typing:

```
mgdb --help
```

on the command line.

Initial setup

Step 1: Set up your MongoDB server (either with MongoLab, or starting your own server)

Step 2: Run

```
mgdb init -c db.json
```

to create a database credentials file called db.json. The file stores your database connection details and authentication information, which is used by other commands to access the database.

Inserting your first VASP calculation(s)

We have run a few vasp calculations for Cu, Au and CuAu. The output is in the "CuAu" directory.

To insert all three calculations into the MongoDB, you simply need to type:

```
mgdb insert -c /path/to/db.json CuAu
```

Exploring the MongoDB collection

To explore the MongoDB collection, you have several options:

- Use the mongo command line given by MongoDB itself.
- Use a GUI MongoDB manager such as MongoHub (use the actively maintained version at https://github.com/jeromelebel/MongoHub-Mac)
- Use the Flamyngo (we will cover this later).
- You can do simple queries using mgdb

```
mgdb query -c db.json --crit '{"pretty_formula": "CuAu"}' --props task_id energy_per_atom
```

- Write python code using the matgendb.QueryEngine class to perform queries.

Please consult the pymatgen-db documentation for details of the last two options.

More advanced usage

mgdb CLI tool is simply a way to facilitate usage of pymatgen-db for common calculations and use cases. Pymatgen-db is capable of a lot more.

Two key classes here that most developers would need to be aware of:

- **matgendb.creator.VaspToDBTaskDrone:** Drone class that can be used with pymatgen's borg framework to assimilate all calculations within a directory structure and insert them into the database.
- **matgendb.query_engine.QueryEngine**: Interface layer (utilizing pymongo) between pymatgen objects and the MongoDB database. Facilitates conversions to pymatgen objects, as well as various syntax enhancements for materials applications, e.g., standardizing formula strings for queries.

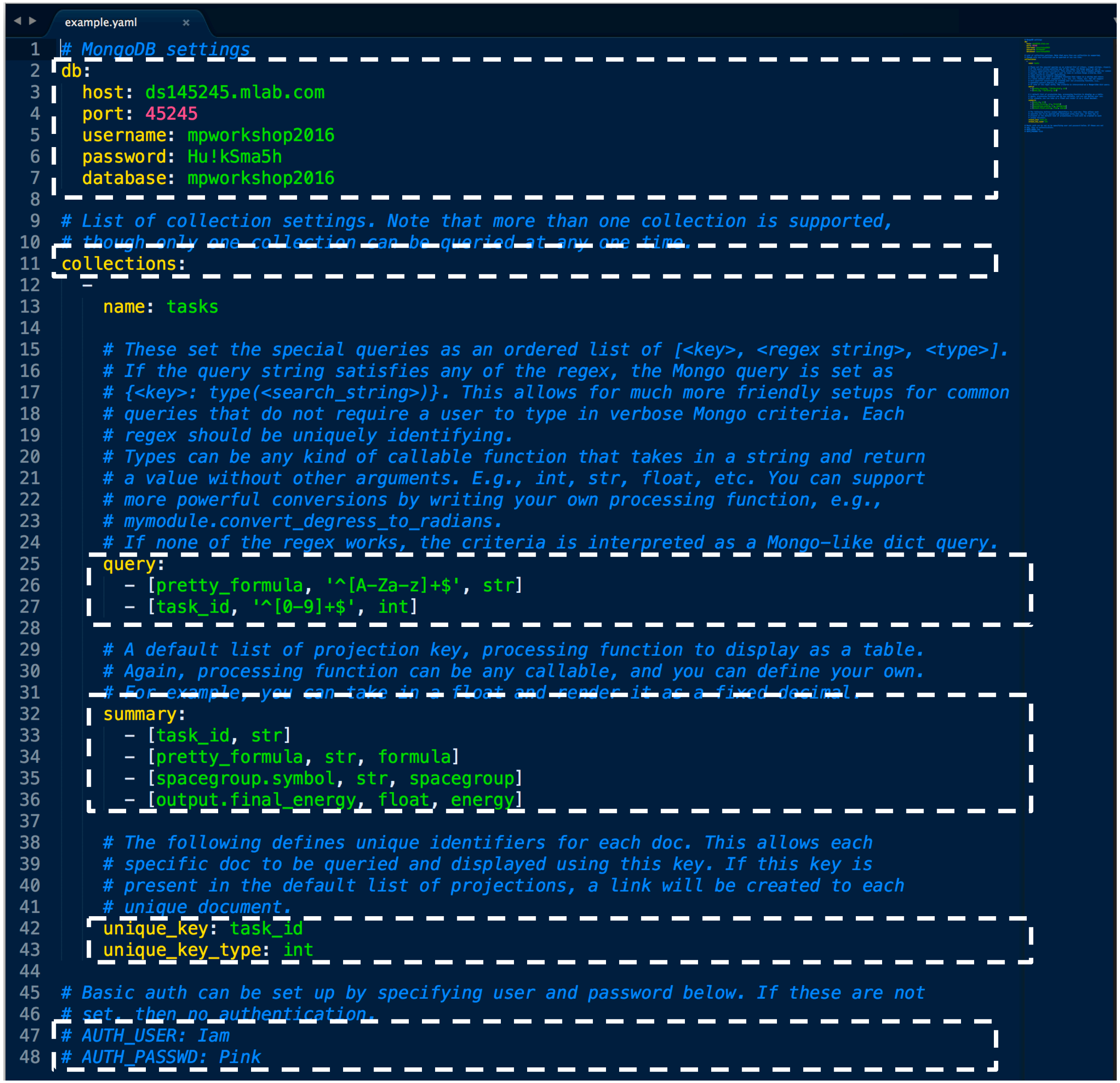
Flask + pymongo = *Flamyngo*

Flamyngo

Flamyngo is a Flask-based (a micro web framework) web interface for MongoDB databases.

Features:

- Most common use scenarios can be completely controlled via a YAML configuration file.
- Extensible to support complex data formatting and types.
- Flexible plotting capabilities (beta!).
- Very customizable.
- Starting point for full fledged web applications.



```yaml
# MongoDB settings
db:
  host: ds145245.mlab.com
  port: 45245
  username: mpworkshop2016
  password: Hu!kSma5h
  database: mpworkshop2016

# List of collection settings. Note that more than one collection is supported,
# though only one collection can be queried at any one time.
collections:
  -
    name: tasks

    # These set the special queries as an ordered list of [<key>, <regex string>, <type>].
    # If the query string satisfies any of the regex, the Mongo query is set as
    # {<key>: type(<search_string>)}. This allows for much more friendly setups for common
    # queries that do not require a user to type in verbose Mongo criteria. Each
    # regex should be uniquely identifying.
    # Types can be any kind of callable function that takes in a string and return
    # a value without other arguments. E.g., int, str, float, etc. You can support
    # more powerful conversions by writing your own processing function, e.g.,
    # mymodule.convert_degress_to_radians.
    # If none of the regex works, the criteria is interpreted as a Mongo-like dict query.
    query:
      - [pretty_formula, '^[A-Za-z]+$', str]
      - [task_id, '^[0-9]+$', int]

    # A default list of projection key, processing function to display as a table.
    # Again, processing function can be any callable, and you can define your own.
    # For example, you can take in a float and render it as a fixed decimal.
    summary:
      - [task_id, str]
      - [pretty_formula, str, formula]
      - [spacegroup.symbol, str, spacegroup]
      - [output.final_energy, float, energy]

    # The following defines unique identifiers for each doc. This allows each
    # specific doc to be queried and displayed using this key. If this key is
    # present in the default list of projections, a link will be created to each
    # unique document.
    unique_key: task_id
    unique_key_type: int

# Basic auth can be set up by specifying user and password below. If these are not
# set, then no authentication.
# AUTH_USER: Iam
# AUTH_PASSWD: Pink
```

Database settings

Multiple collections can be specified

Options for simplified querying

Summary of results

Unique document identifier

Optional auth settings

Demo

Bonus tip – Free deployment via Heroku

Sign up for an account on Heroku.com.

Create new app called <appname>. The free dyno option will do for now. Go through the steps of installing the heroku toolbelt and login.

```
cd <appname>
git init
heroku git:remote -a <appname>
```

Copy the “requirements.txt”, “example.yaml” and “Procfile” into the repo. Commit and then push.

```
git add .
git commit -am ”initial deployment”
git push heroku master
```

The website will then be deployed to heroku. Go to https://<appname>.herokuapp.com.

Example at https://mpworkshop2016.herokuapp.com

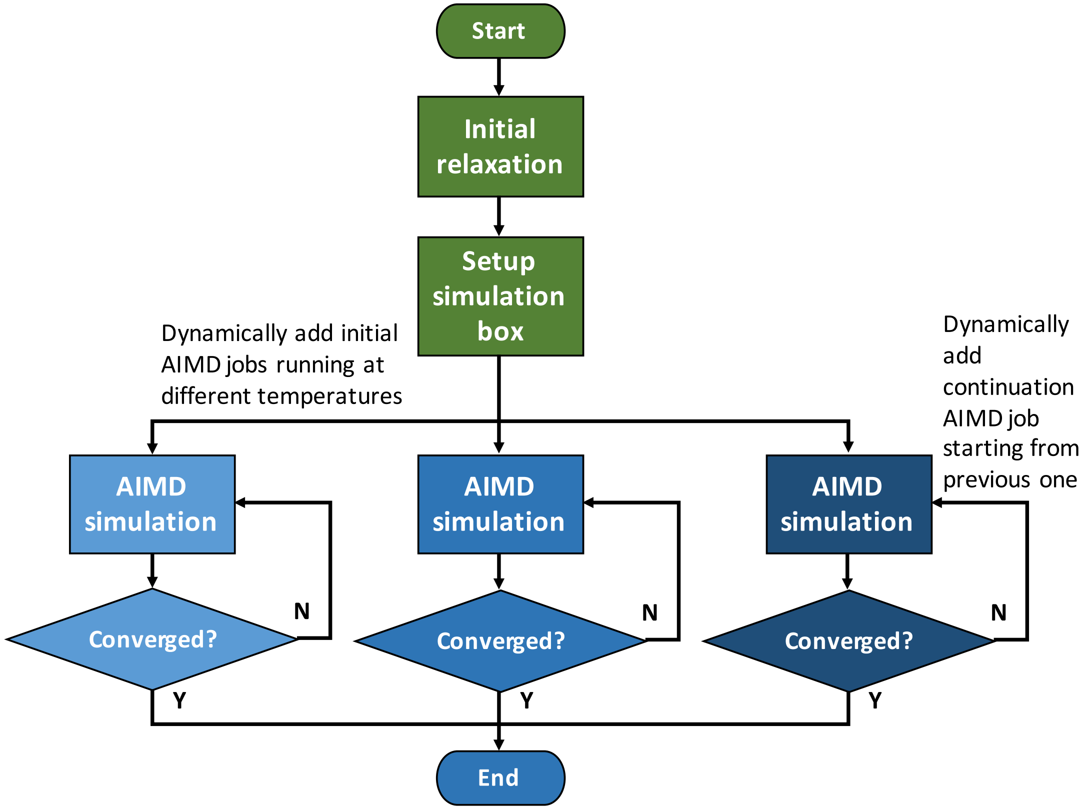
Examples of pymatgen-db & flamyngo powered databases and websites

Note: Some of these are private databases and websites, and I am showing them via live demo.

- http://crystalium.materialsvirtuallab.org
- https://spydy.herokuapp.com
- https://mqm-eie.herokuapp.com

AIMD @ The Materials Virtual Lab

Combining pymatgen + pymatgen-db + custodian + fireworks


Start
Initial relaxation
Setup simulation box
Dynamically add initial AIMD jobs running at different temperatures
Dynamically add continuation AIMD job starting from previous one
AIMD simulation
AIMD simulation
AIMD simulation
Converged?
Converged?
Converged?
N
N
N
Y
Y
Y
End

Pymatgen

- Analysis and input file generation

Pymatgen-db

- Database insertion

Custodian

- Stops job before wall time

Fireworks

- Manage workflow