

# Data management

pymongo essentials

# Imports

```
import pymongo
```

# Initializing

- You will be working with the collection object, mycol

```
# connect to mongoDB server, db and collection
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["user_database"]
mycol = mydb["users"]
```

# Resetting the database

- When we are testing we will want to reset the DB

```
mycol.drop()
```

# Inserting a document

- Insert a document and inspect the result

```
# insert a user and print inserted id
data1 = {'username': 'joe', 'password': 'schmoe'}
x = mycol.insert_one(data1)
print(x)
```

# ObjectIds

- When you print `_id`, it will look like a string
- But it is actually an ObjectId

```
_id = x.inserted_id  
print(_id)  
print(type(_id))
```

# Forcing a doc to have a str \_id

- You can change the default behavior by creating an \_id before you insert
- This will make things simpler later on
- You will need "from bson import ObjectId"

```
data1 = {'username': 'joe', 'password': 'schmoe'}  
data1['_id'] = ObjectId()  
x = mycol.insert_one(data1)  
print(x)
```

## Reading by id

- The `_id` field is part of the document now
- You can query with an `ObjectId` or `str`
  - this depend on whether you created a `str _id` on creation.

```
doc = mycol.find_one({"_id":_id})  
print(doc)
```



# Reading all users

- `find()` will return all documents in a collection
- `find()` will return a Cursor object

```
data = mycol.find()  
print(type(data))
```

# Working with Cursors

- a Cursor object is an iterable:
  - iterate with a for loop
  - get the next one with next() - will consume one
  - convert to a list - will consume all

```
docs = list(data)
for doc in docs:
    print(doc)
```

# Reading users with a query

- We've already done this when we read by id
- We can also query many with find
- find will return a Cursor, even if there are zero or one results.

```
doc = mycol.find_one({"_id":_id})
print(doc)

docs = mycol.find({'organization':'UNH'})
doc_list = list(docs)
print(doc_list)
```

# Updating

- you can change a value using the \$set operator

```
# update
myquery = {'username':'joe'}
newvalues = {'password':'asdf'}
result = mycol.update_one(myquery, {'$set':newvalues})
print(result)
```

## Embedded document: address

```
# embedded doc
address = {'line1': '33 Academic Way',
           'city': 'Durham',
           'state': 'NH',
           'zip': '03809'}

# update
myquery = {'username': 'joe'}
newvalues = {'address': address}
result = mycol.update_one(myquery, {'$set': newvalues})
```

# Deleting

- inspect the result

```
result = mycol.delete_one({'_id':user_id})  
print(result)
```