# Welcome

## INTRODUCTION TO MONGODB IN PYTHON

**Donny Winston**
Instructor

datacamp

# JavaScript Object Notation (JSON)

## Objects `{}`

- String keys & values
  `{'key1':value1, 'key2':value2,...}`
- Order of values is not important

```
{
'id': 12345,
'name': 'Donny Winston',
'instructor': true
},
```

## Arrays `[]`

- Series of values `[value1, value2,...]`
- Order of values is important

```
[
 "instructor_1",
 "instructor_2",
 ...
]
```

# JavaScript Object Notation (JSON)

```
{
  'people': [
    { 'id': 12345,
      'name': 'Donny Winston',
      'instructor': true,
      'tags': ['Python', 'MongoDB']
    },

    { 'id': 54321
      'name': 'Guido van Rossum'
      'instructor':false
      'tags': null
    },
  ]
}
```

**Values**

- Strings `'name':'Donny Winston'`
- Numbers `'id': 12345`
- `true` / `false`
- `null`
- Another array

  ```
  'tags': ['Python', 'MongoDB']
  ```

- Another object

  ```
  [{ 'id': 12345,  ...},...]
  ```

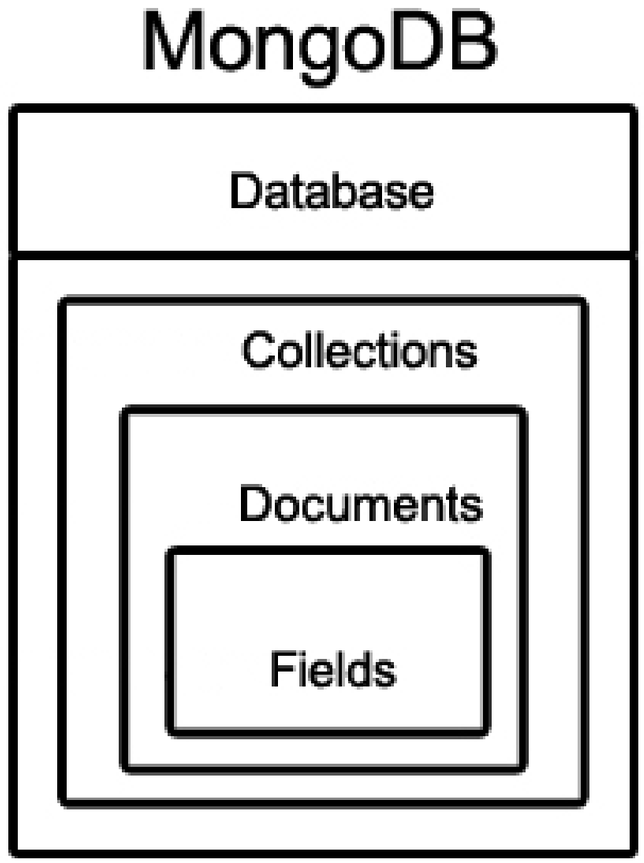
# JSON <> Python

| JSON | Python |
| --- | --- |
| Objects | Dictionaries `dict` |
| Arrays | Lists `list` |
| **Values:** | |
| · *strings* | `str` |
| · _numbers _ | `int` , `float` |
| · `true` / `false` | `True` / `False` |
| · `null` | `None` |
| · *other objects/arrays* | other `dict` / `list` |

--

# JSON <> Python <> MongoDB

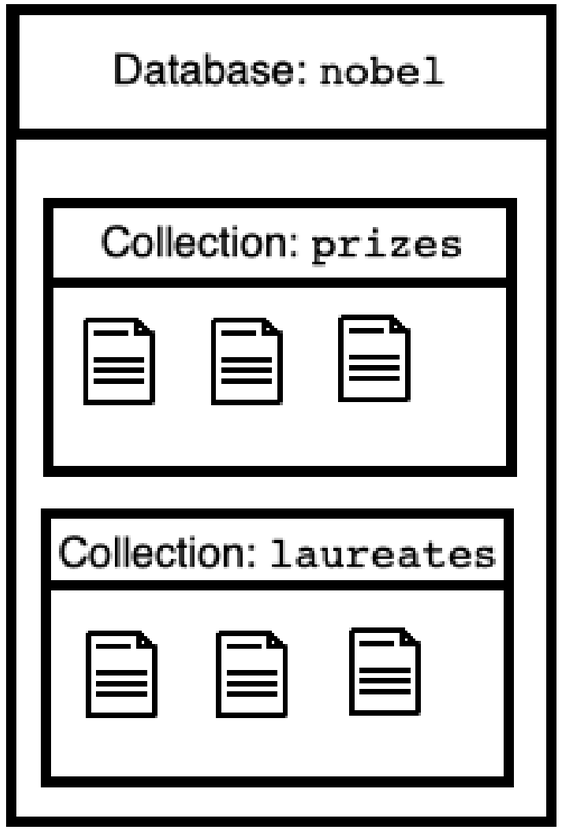
| MongoDB | JSON | Python |
|---|---|---|
| Databases | Objects | Dictionaries |
| ↳Collections | Arrays | Lists |
| ↳ ↳Documents | Objects | Dictionaries |
| ↳ ↳ ↳Subdocuments | Objects | Dictionaries |
| ↳ ↳ ↳ Values | Value types | Value types + `datetime`, regex... |



# The Nobel Prize API data(base)

```
import requests
from pymongo import MongoClient
# Client connects to "localhost" by default
client = MongoClient()
# Create local "nobel" database on the fly
db = client["nobel"]

for collection_name in ["prizes", "laureates"]:
    # collect the data from the API
    response = requests.get(
        "http://api.nobelprize.org/v1/{}.json".\
      format(collection_name[:-1] ))
    # convert the data to json
    documents = response.json()[collection_name]
    # Create collections on the fly
    db[collection_name].insert_many(documents)
```



# Accessing databases and collections

- Using `[]`

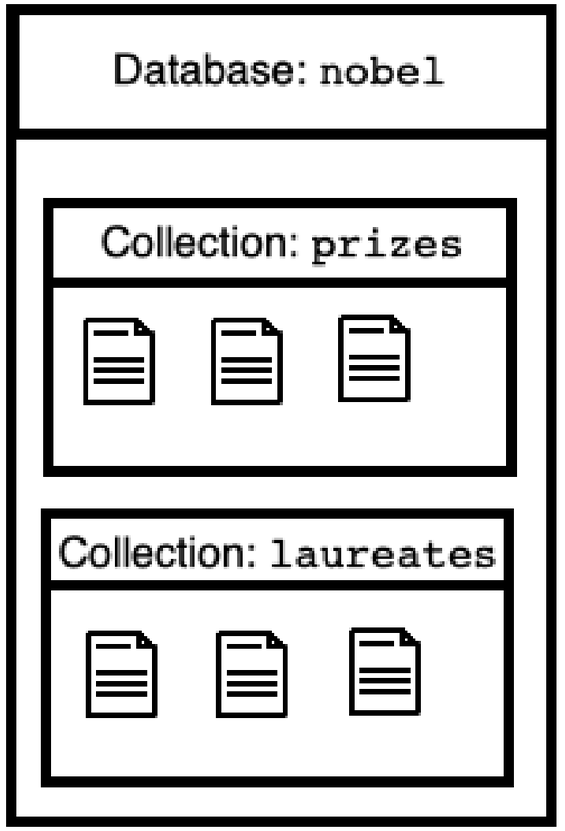
```
# client is a dictionary of databases
db = client["nobel"]

# database is a dictionary of collections
prizes_collection = db["prizes"]
```

- Using `.`

```
# databases are attributes of a client
db = client.nobel

# collections are attributes of databases
prizes_collection = db.prizes
```



# Count documents in a collection

```
# Use empty document {} as a filter
filter = {}

# Count documents in a collection
n_prizes = db.prizes.count_documents(filter)
n_laureates = db.laureates.count_documents(filter)
```

```
590
934
```

```
# Find one document to inspect
doc = db.prizes.find_one(filter)
```

```
{'_id': ObjectId('5bc56145f35b634065ba1996'),
 'category': 'physics',
 'laureates': [{'firstname': 'Arthur',
   'id': '960',
   'motivation': '"for the optical tweezers and their
   application to biological systems"',
   'share': '2',
   'surname': 'Ashkin'},
  {'firstname': 'Gérard',
   'id': '961',
   'motivation': '"for their method of generating
   high-intensity, ultra-short optical pulses"',
  ...
```

# Let's practice!

INTRODUCTION TO MONGODB IN PYTHON

# Finding documents

## INTRODUCTION TO MONGODB IN PYTHON

**Donny Winston**
Instructor

datacamp

# An example "laureates" document

```
{'_id': ObjectId('5b9ac94ff35b63cf5231ccb1'),
 'born': '1845-03-27',
 'bornCity': 'Lennep (now Remscheid)',
 'bornCountry': 'Prussia (now Germany)',
 'bornCountryCode': 'DE',
 'died': '1923-02-10',
 'diedCity': 'Munich',
 'diedCountry': 'Germany',
 'diedCountryCode': 'DE',
 'firstname': 'Wilhelm Conrad',
 'gender': 'male',
 'id': '1',
 'prizes': [{'affiliations': [{'city': 'Munich',
                               'country': 'Germany',
                               'name': 'Munich University'}],
             'category': 'physics',
             'motivation': '"in recognition of the extraordinary services '
                           'he has rendered by the discovery of the '
                           'remarkable rays subsequently named after him"',
             'share': '1',
             'year': '1901'}],
 'surname': 'Röntgen'}
```

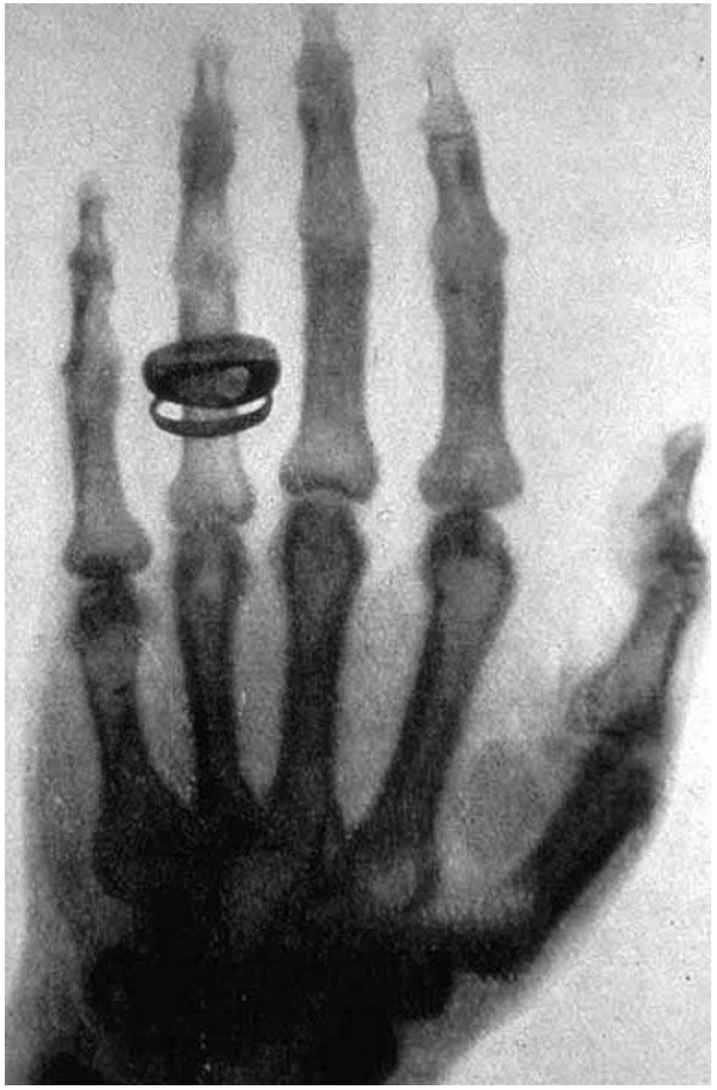
# Filters as (sub)documents

Count documents by providing a filter document to match.

```
filter_doc = {
 'born': '1845-03-27',
 'diedCountry': 'Germany',
 'gender': 'male',
 'surname': 'Röntgen'
}

db.laureates.count_documents(filter_doc)
```

```
1
```



Jimi

Amelia

Charlie

Wally

Levi

# Simple filters

```
db.laureates.count_documents({'gender': 'female'})
```

```
48
```

```
db.laureates.count_documents({'diedCountry': 'France'})
```

```
50
```

```
db.laureates.count_documents({'bornCity': 'Warsaw'})
```

```
2
```

# Composing filters

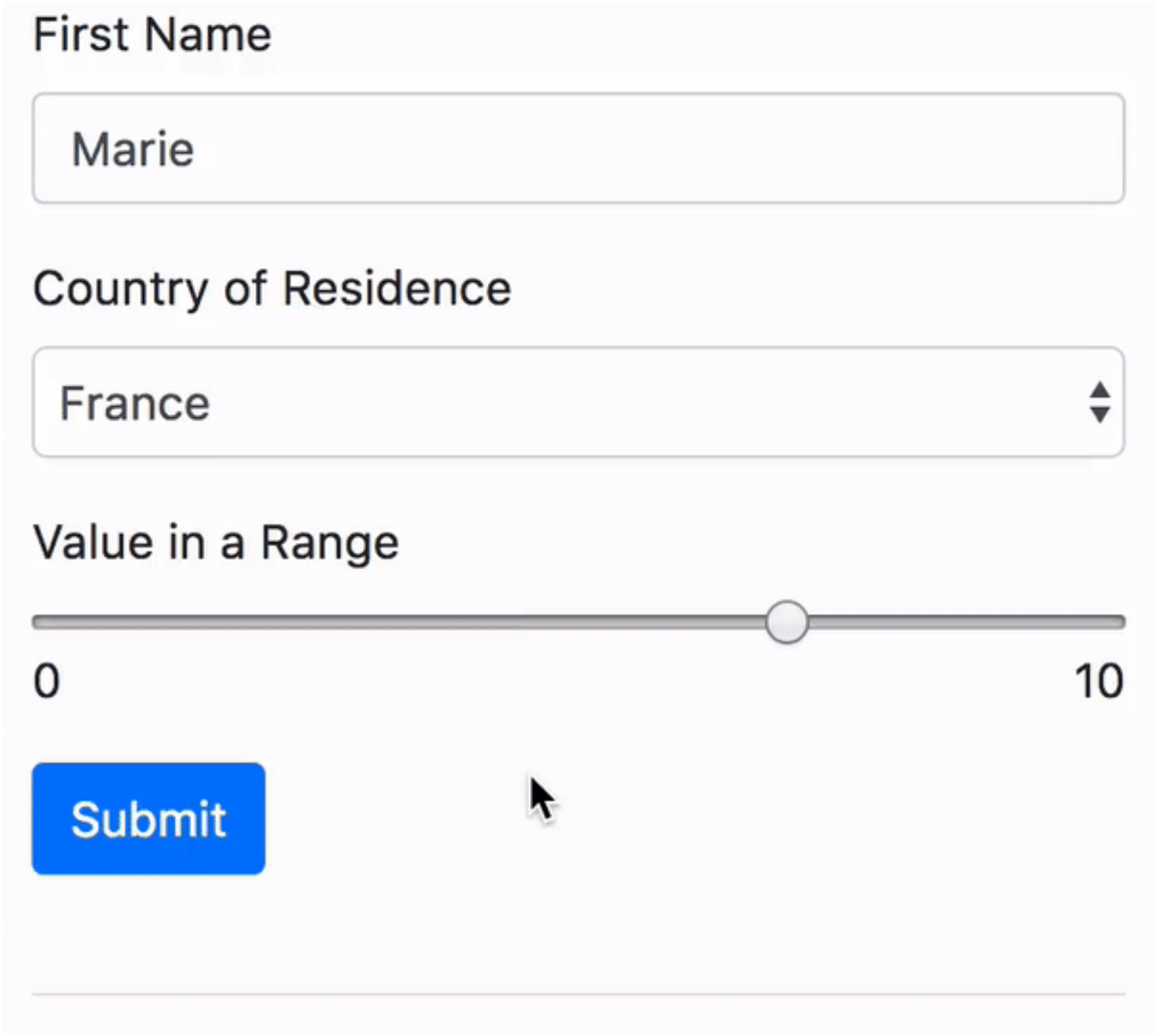
```
filter_doc = {'gender': 'female',
              'diedCountry': 'France',
              'bornCity': 'Warsaw'}
db.laureates.count_documents(filter_doc)
```

```
1
```

```
db.laureates.find_one(filter_doc)
```

```
{'_id': ObjectId('5bc56154f35b634065ba1be9'),
 'born': '1867-11-07',
 'bornCity': 'Warsaw',
 'bornCountry': 'Russian Empire (now Poland)',
 'bornCountryCode': 'PL',
 'died': '1934-07-04',
 'diedCity': 'Sallanches',
 'diedCountry': 'France',
 'diedCountryCode': 'FR',
 'firstname': 'Marie',
 ...
```

# Query operators



# Query operators

- Value in a range `$in: <list>`

```
db.laureates.count_documents({
 'diedCountry': {
  '$in': ['France', 'USA']}})
```

```
258
```

- Not equal `$ne : <value>`

```
db.laureates.count_documents({
 'diedCountry': {
  '$ne': 'France'}})
```

```
872
```

**Query syntax:**

```
{
  # Match a single value exactly:
  'field_name1': value1,

  # Use operators:
  'field_name2': {
      $operator1: value1,
      $operator2: value2,
      ... # more operators
    },
    ... # more fields
}
```

# Query operators

- Comparison:
  - $>$ : `$gt` , $\geq$ : `$gte`
  - $<$ : `$lt` , $\leq$ : `$lte`

```
db.laureates.count_documents({
 'diedCountry': {
  '$gt': 'Belgium',
  '$lte': 'USA'}})
```

```
453
```

```
453
```

(Strings are compared lexicographically)

**Query syntax:**

```
{
  # Match a single value exactly:
  'field_name1': value1,

  # Use operators:
  'field_name2': {
      $operator1: value1,
      $operator2: value2,
      ... # more operators
    },
    ... # more fields
}
```

# Let's Practice!

INTRODUCTION TO MONGODB IN PYTHON

# Dot notation: reach into substructure

INTRODUCTION TO MONGODB IN PYTHON

**Donny Winston**
Instructor

datacamp

# A functional density

```
db.laureates.find_one({
  "firstname": "Walter",
  "surname": "Kohn"})
```

```
{'born': '1923-03-09',
 'bornCity': 'Vienna',
 'bornCountry': 'Austria',
 'firstname': 'Walter',
 'prizes': [
  {'affiliations': [
    {'city': 'Santa Barbara, CA',
     'country': 'USA',
     'name': ('University of '
              'California')
    }],
   'category': 'chemistry',
   'motivation': (
    '"for his development of the '
    'density-functional theory"'),
   'share': '2',
   'year': '1998'
  }],
 'surname': 'Kohn',
 ...} # showing partial document
```

```
db.laureates.count_documents({
 "prizes.affiliations.name": (
  "University of California")})
```

```
34
```

```
db.laureates.count_documents({
 "prizes.affiliations.city": (
  "Berkeley, CA")})
```

```
19
```

# No Country for Naipaul

```
db.laureates.find_one({'surname': 'Naipaul'})
```

```
{'_id': ObjectId('5b9ec791f35b63093c3d98b7'),
 'born': '1932-08-17',
 'died': '2018-08-11',
 'diedCity': 'London',
 'diedCountry': 'United Kingdom',
 'diedCountryCode': 'GB',
 'firstname': 'Sir Vidiadhar Surajprasad',
 'gender': 'male',
 'id': '747',
 'prizes': [{'affiliations': [[]],
   'category': 'literature',
   'motivation': ('"for having united perceptive narrative and '
                  'incorruptible scrutiny in works that compel us '
                  'to see the presence of suppressed histories"'),
   'share': '1',
   'year': '2001'}],
 'surname': 'Naipaul'}
```

```
db.laureates.count_documents({"bornCountry": {"$exists": False}})
```

```
31
```

# Multiple prizes

```
db.laureates.count_documents({})
```

```
922
```

```
db.laureates.count_documents({"prizes": {"$exists": True}})
```

```
922
```

```
db.laureates.count_documents({"prizes.0": {"$exists": True}})
```

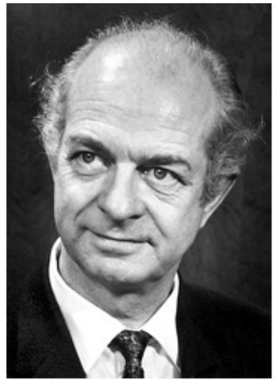
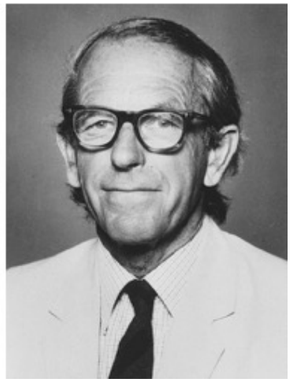
```
922
```

```
db.laureates.count_documents({"prizes.1": {"$exists": True}})
```

```
6
```

# Multiple-prize winners



# On to exercises!

INTRODUCTION TO MONGODB IN PYTHON