MongoKIT

A python ODM for MongoDB

```
class MyDocument(Document) :
```

Structure

Descriptors

Options

```
class MyDocument(Document) :

    structure = {
        'foo': int,
        'bar': float,
        'spam':{
            'eggs': [unicode],
            'blah': None,
        }
    }
```

Descriptors

Options

```
class MyVeryNestedDoc(Document):
    structure = {
        '1':{
            '2':{
                '3':{
                    '4':{
                        '5':{
                            '6':{
                                '7':int,
                                '8':{
                                    '9':float,
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

```
class MyDocument(Document) :

structure = {
    'foo': unicode,
    'bar': int,
    'spam':{
        'eggs': [unicode],
        'blah': float,
    }
}
```

Descriptors

Options

```
class MyDocument(Document) :

    structure = {
        'foo': unicode,
        'bar': int,
        'spam':{
            'eggs': [unicode],
            'blah': float,
        }
    }

    required = ['foo', 'spam.eggs']
    default_values = {'spam.blah' : 1.0}
    validators = {'bar':lambda x >0}
```

Options

```
class MyDocument(Document) :

    structure = {
        'foo': unicode,
        'bar': int,
        'spam':{
            'eggs': [unicode],
            'blah': float,
        }
    }

    required = ['foo', 'spam.eggs']
    default_values = {'spam.blah' : 1.0}
    validators = {'bar':lambda x >0}

    use_dot_notation = True
    skip_validation = True
```

Advantages

- **Great** readability
- **Simple** python dict
- **Pure** python types
- **Nested** and complex schema declaration
- **Fast** : don't instanciate objects
- **Live** update via instrospection
- **Dynamic** keys

Dynamic keys

```
class MobilePhones(Document) :
    structure = {
        'os' : {
            unicode:[{'version': float, 'name':unicode}],
        }
```

```
{'os' : {
    'android':[
        {'version': 2.2, 'name' :'froyo'},
        {'version': 2.1, 'name' :'eclair'}
    ],
    'iphone':[{'version': 4, 'name' :'iOS'}],
}
```

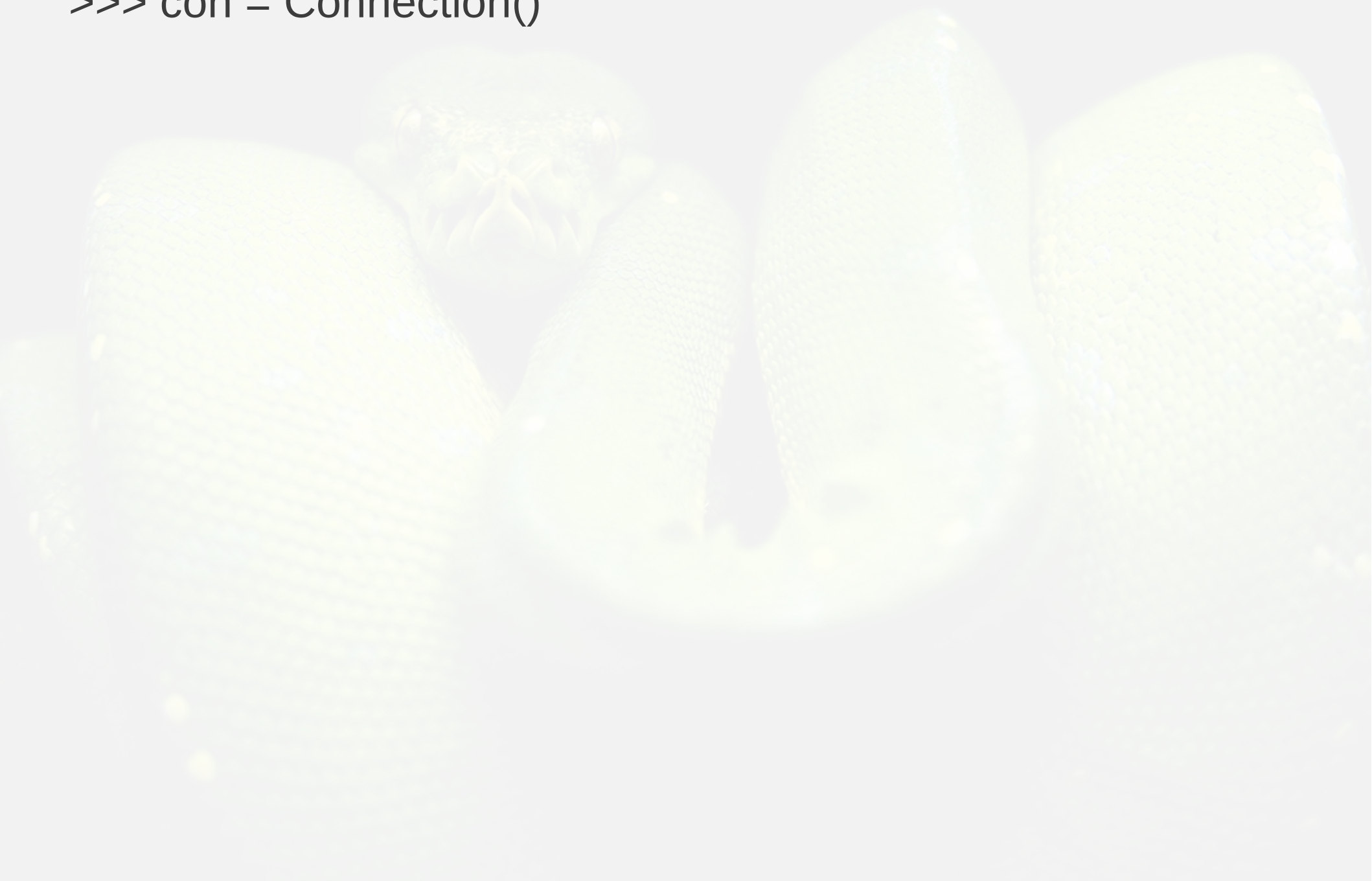
It's all about Pymongo

- Built on top of Pymongo
- Use the same syntax

- Built on top of Pymongo
- Use the same syntax

- ✔ Easy code migration (pymongo → MongoKit)
- ✔ Learn fast
- ✔ One syntax to rule them all

```
>>> from mongokit import *
>>> con = Connection()
```



```
>>> from mongokit import *
>>> con = Connection()
```

Pymongo's way

```
>>> doc = con.mydb.mycol.find_one() # very fast !
# doc is a dict instance
```

```
>>> from mongokit import *
>>> con = Connection()
```

Pymongo's way

```
>>> doc = con.mydb.mycol.find_one() # very fast !
# doc is a dict instance
```

Mongokit's way

```
>>> doc = con.mydb.mycol.MyDocument.find_one()
# doc is a MyDocument instance
>>> doc.spam.eggs.append(u'foo')
>>> doc.save()
```

Features

Inheriance / Polymorphism

```
class A(Document) :
    structure = {
        'a': {
            'foo' : unicode,
        }
    }
```

```
class B(Document) :
    structure = {
        'b': {
            'bar' : [float],
        }
    }
```

Inheriance / Polymorphism

```
class A(Document) :
    structure = {
        'a': {
            'foo' : unicode,
        }
    }
```

```
class B(Document) :
    structure = {
        'b': {
            'bar' : [float],
        }
    }
```

```
class C(A,B) :
    structure = {
        'c': {
            'spam' : int,
        }
    }
```

Inheriance / Polymorphism

```
class A(Document) :
    structure = {
        'a': {
            'foo' : unicode,
        }
    }
```

```
class B(Document) :
    structure = {
        'b': {
            'bar' : [float],
        }
    }
```

```
class C(A,B) :
    structure = {
        'c': {
            'spam' : int,
        }
    }
```

```
>>> con.mydb.mycol.C()
{
    'a' : {'foo' : None},
    'b' : {'bar' : []},
    'c' : {'spam' : None}
}
```

Dot notation

```
class MyDocument(Document) :
    structure = {
        'foo': unicode,
        'bar': int,
        'spam':{
            'eggs': [unicode],
            'blah': float,
        }
    }
```

Dot notation

```
class MyDocument(Document) :
    structure = {
        'foo': unicode,
        'bar': int,
        'spam':{
            'eggs': [unicode],
            'blah': float,
        }
    }
    use_dot_notation = True
```

Dot notation

```
class MyDocument(Document) :
    structure = {
        'foo': unicode,
        'bar': int,
        'spam':{
            'eggs': [unicode],
            'blah': float,
        }
    }
    use_dot_notation = True
```

```
>>> doc = con.mydb.mycol.MyDocument()
>>> doc.foo = u'the foo'
>>> doc.spam.eggs.append(u'bla', u'toto')
```

Document reference

```
class User(Document) :
    structure = {
        'login': unicode,
        'name': unicode,
    }
```

```
class Comment(Document) :
    structure = {
        'title': unicode
        'body': unicode,
        'author' : ObjectId,
    }
```

Document reference

```
class User(Document) :
    structure = {
        'login': unicode,
        'name': unicode,
    }
```

```
class Comment(Document) :
    structure = {
        'title': unicode
        'body': unicode,
        'author' : User,
    }
    use_autorefs = True
```

Document reference

```
class User(Document) :
    structure = {
        'login': unicode,
        'name': unicode,
    }
```

```
class Comment(Document) :
    structure = {
        'title': unicode
        'body': unicode,
        'author' : User,
    }
    use_autorefs = True
```

```
>>> con.mydb.mycol.find_one()
```

```
{
    'title' : 'Hello world !',
    'body' : 'My first blog post',
    'author' : DBRef(...),
}
```

Document reference

```
class User(Document) :
    structure = {
        'login': unicode,
        'name': unicode,
    }
```

```
class Comment(Document) :
    structure = {
        'title': unicode
        'body': unicode,
        'author' : User,
    }
    use_autorefs = True
```

```
>>> con.mydb.mycol.find_one()
{
    'title' : 'Hello world !',
    'body' : 'My first blog post',
    'author' : DBRef(...),
}
```

```
>>> con.mydb.mycol.BlogPost.find_one()
{
    'title' : 'Hello world !',
    'body' : 'My first blog post',
    'author' : {
        'login' : 'timy',
        'name' : 'Timy Donzy'
    }
}
```

GridFS support

```
class MyDocument(Document) :
    structure = {
        'foo': unicode,
        'bar': int,
    }
    grid_fs = {
       'files':['source', 'template'],
       'containers': ['images'],
    }
```

```
>>> doc = con.mydb.mycol.MyDocument()
>>> doc.fs.source = '...'
>>> doc.fs.images['image1.png'] = '…'
```

i18n

```
class MyDocument(Document) :
    structure = {
        'foo': unicode,
        'bar': int,
    }
    i18n = ['foo']
    use_dot_notation = True
```

```
>>> doc = con.mydb.mycol.MyDocument()
>>> doc.set_lang('fr')
>>> doc.foo = u'Salut'

>>> doc.set_lang('en')
>>> doc.foo = u'Hello'
>>> doc.save()
```

- Inheritance / Polymorphism
- Dot notation
- Document auto-reference support (DBRef)
- GridFS
- I18n support

- Inheritance / Polymorphism
- Dot notation
- Document auto-reference support (DBRef)
- GridFS
- I18n support
- Schema migration
- Json export/import

- Inheritance / Polymorphism
- Dot notation
- Document auto-reference support (DBRef)
- GridFS
- I18n support
- Schema migration
- Json export/import

CAN BE DISABLED