

Jour-27 Python avec MongoDB

Python est une technologie backend et peut être connectée à différentes applications de base de données. Il peut être connecté aux bases de données SQL et NOSQL. Dans cette section, nous connectons Python à la base de données MongoDB qui est la base de données NoSQL.

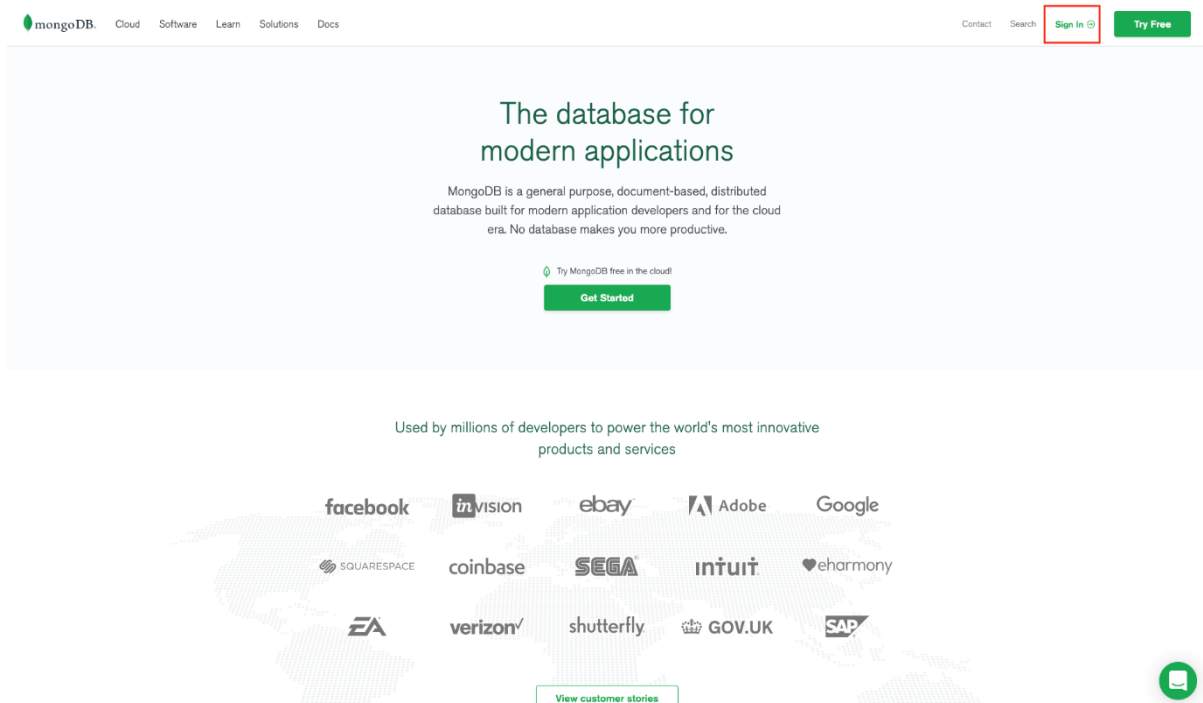
Mongodb

MongoDB est une base de données NoSQL. MongoDB stocke les données dans un document de type JSON qui rend MongoDB très flexible et évolutif. Voyons les différentes terminologies des bases de données SQL et NOSQL. Le tableau suivant fera la différence entre les bases de données SQL contre NOSQL.




SQL contre Nosql

SQL	NoSQL
Database	Database
Tables	Collections
Rows	Documents
Columns	Fields
Index	Index
Join	Embedding and Linking
Group by	Aggregation
Primary Key	_id field

Dans cette section, nous nous concentrerons sur une base de données NOSQL MongoDB. Permet de vous inscrire sur MongoDB en cliquant sur le bouton Connexion, puis cliquez sur Inscrivez-vous sur la page suivante.



Complétez les champs et cliquez sur Continuer


MetLife  BuzzFeed 

Sign Up

Email Address

Password

✓ 8 characters minimum

First Name

Last Name

Phone Number

Company Name

Job Function

None Selected

Country


Select Country

☐ I agree to the [terms of service](#) and [privacy policy](#)

Already have an account? [Login](#)

Continue

Sélectionnez le plan gratuit


MONGODB ATLAS
Choose a path. Adjust anytime.
Available as a fully managed service across 60+ regions on AWS, Azure, and Google Cloud

Starter Clusters
For teams learning MongoDB or developing small applications.

- ✓ Highly available auto-healing cluster
- ✓ End-to-end encryption
- ✓ Role-based access control
- ✗ No downtime scaling
- ✗ Network isolation
- ✗ Realtime performance metrics

Starting at
FREE

Create a cluster

Single-Region Clusters
For teams building applications that need advanced development and production-ready environments.

- ✓ Includes all features from Starter Clusters
- ✓ No downtime scaling
- ✓ Network isolation
- ✓ Realtime performance metrics

Starting at
\$0.08/hr*
*estimated cost \$50.94/month

Create a cluster

Multi-Region Clusters
For teams developing world-class applications that require multi-region resiliency or ultra-low latency.

- ✓ Includes all features from Starter and Single-Region Clusters
- ✓ Replicate data across multiple regions
- Global Clusters ☐
- ✓ Globally distributed read and write operations
- ✓ Control data residency at the document level

Starting at
\$0.13/hr*
*estimated cost \$58.55/month

Create a cluster

[Skip](#) [Advanced Configuration Options](#)

Choisissez la région gratuite immédiate et donnez n'importe quel nom pour votre cluster.

Create a Starter Cluster

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

Cloud Provider & Region

AWS, N. Virginia (us-east-1) ▾



Create a **free tier cluster** by selecting a region with **FREE TIER AVAILABLE** and choosing the **M0** cluster tier below.

★ Recommended region ⓘ

NORTH AMERICA	EUROPE	ASIA
<div> N. Virginia (us-east-1) ★ FREE TIER AVAILABLE</div> <div> Oregon (us-west-2) ★ FREE TIER AVAILABLE</div>	<div> Ireland (eu-west-1) ★ FREE TIER AVAILABLE</div> <div> Frankfurt (eu-central-1) ★ FREE TIER AVAILABLE</div>	<div> Singapore (ap-southeast-1) ★ FREE TIER AVAILABLE</div> <div> Mumbai (ap-south-1) FREE TIER AVAILABLE</div>
AUSTRALIA		
<div> Sydney (ap-southeast-2) ★ FREE TIER AVAILABLE</div>		

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage) >
Encrypted

Additional Settings

MongoDB 4.0, No Backup >

Cluster Name

30DaysOfPython ▾

One time only: once your cluster is created, you won't be able to change its name.

Cluster names can only contain ASCII letters, numbers, and hyphens.

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Back

Create Cluster

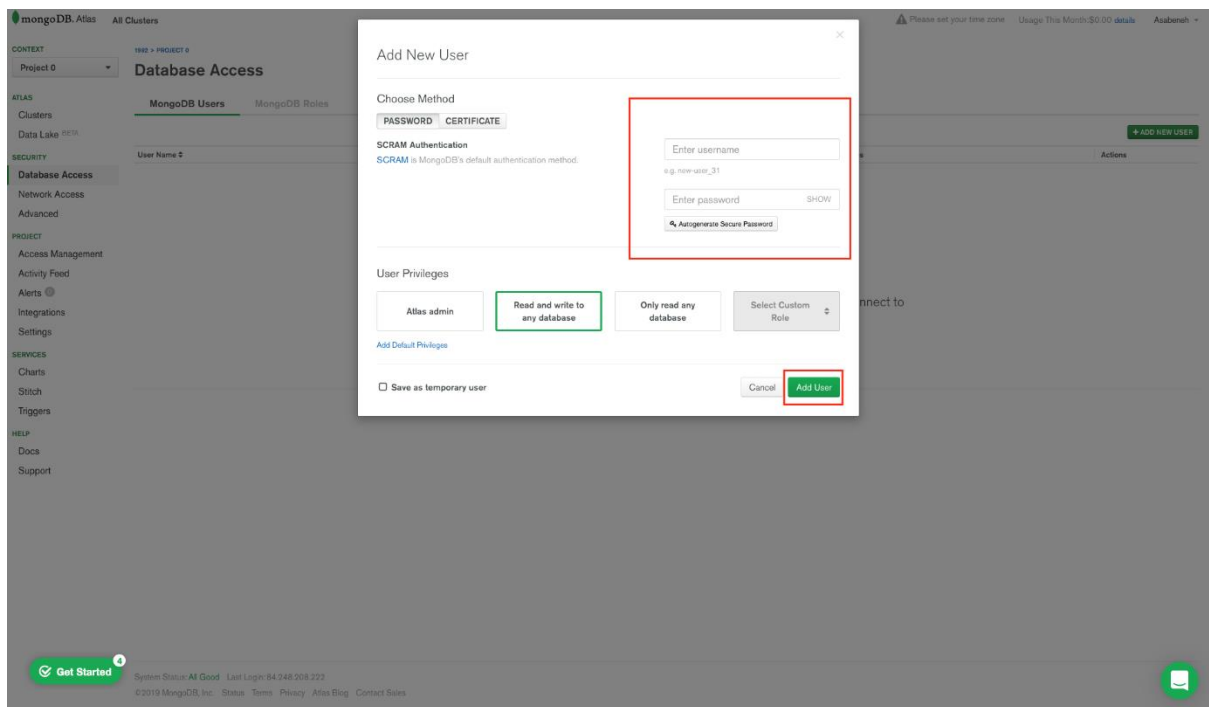
Maintenant, un bac à sable gratuit est créé

The screenshot shows the MongoDB Atlas interface. On the left, a sidebar contains navigation links for Context, Atlas, Security, Project, and Services. The main area displays the 'Clusters' page for a specific cluster. A 'Connect to Atlas' dialog box is open, showing a checklist of steps to get started. Red arrows highlight the 'CONNECT' tab and the 'COLLECTIONS' link. A 'Get Started' button is located at the bottom left.

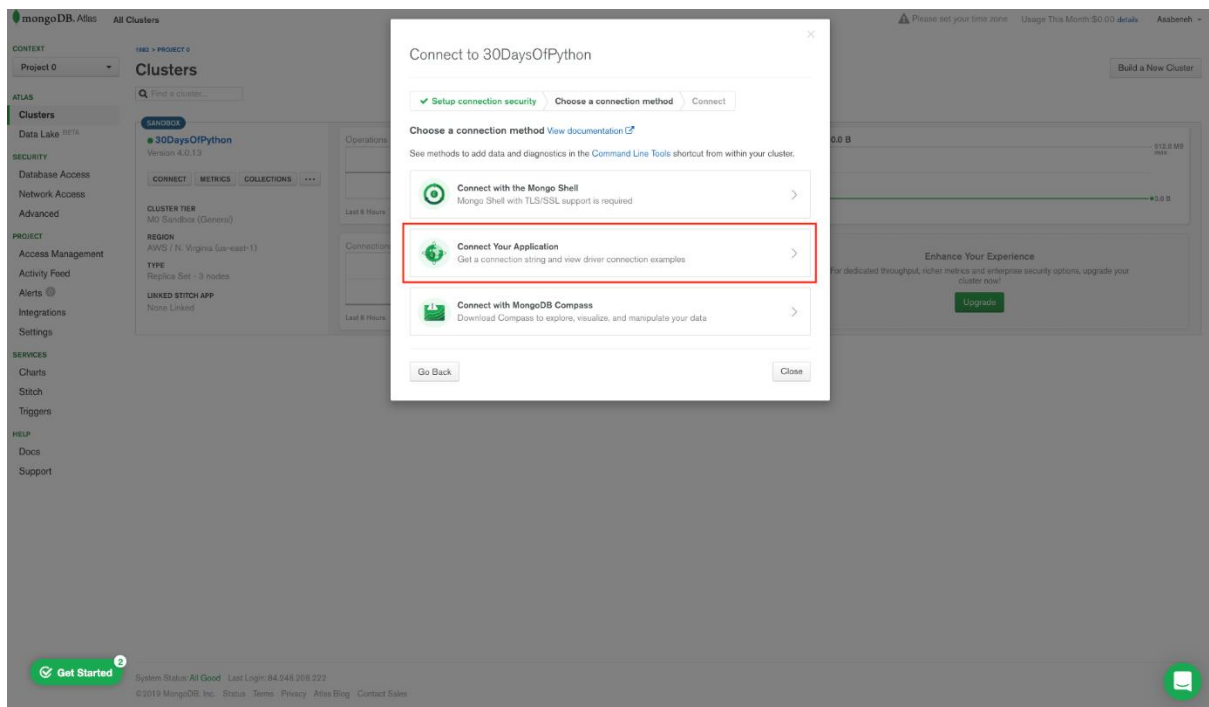
All local host access

The screenshot shows the MongoDB Atlas 'Network Access' page. The 'IP Whitelist' tab is selected. A 'Add Whitelist Entry' dialog box is open, showing options to 'Add Current IP Address' or 'Allow Access From Anywhere'. The 'Confirm' button is highlighted. The dialog box also includes fields for 'Whitelist Entry' and 'Comment'.

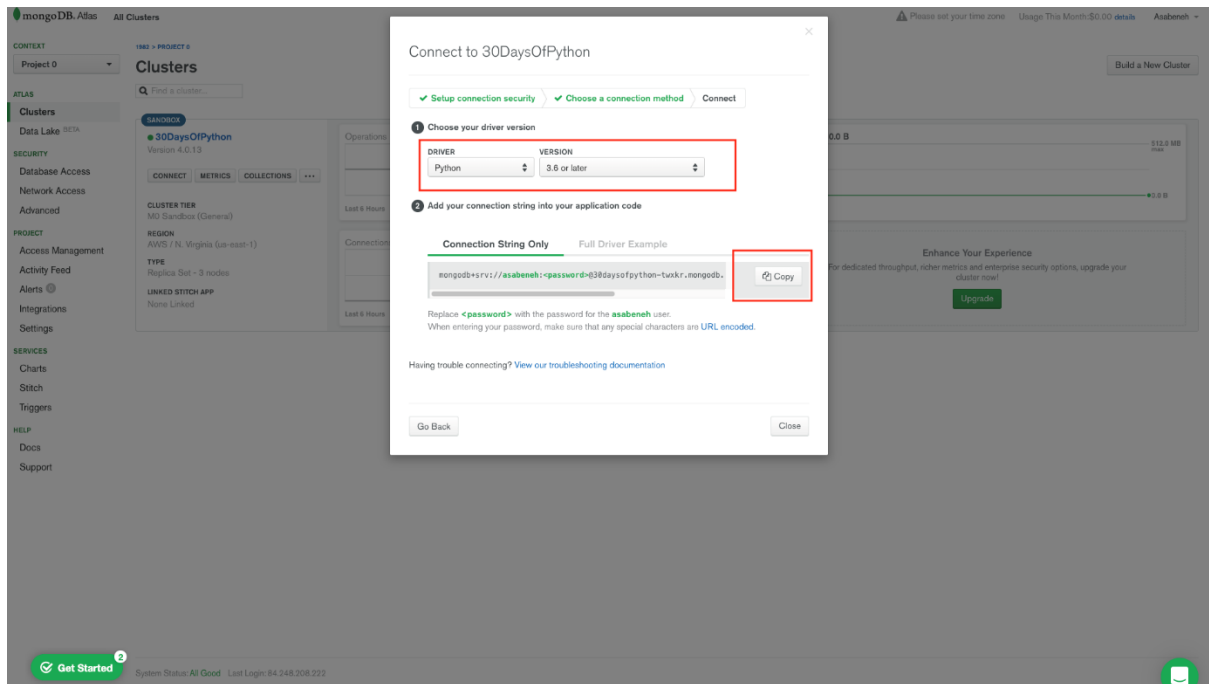
Ajouter l'utilisateur et le mot de passe



Create a mongoDB uri link



Sélectionnez Python 3.6 ou au-dessus du pilote



Obtenir une chaîne de connexion (MongoDB URI)

Copiez le lien de chaîne de connexion et vous obtiendrez quelque chose comme ceci:

```
MongoDB + srv: // asabeneh: < mot de passe > @ 30daysofpython- twxkr.mongodb.net/test?retryWrites =True&w=majority
```

Ne vous inquiétez pas de l'URL, c'est un moyen de connecter votre application à MongoDB. Remplacement de l'espace réservé du mot de passe par le mot de passe que vous avez utilisé pour ajouter un utilisateur.

Exemple:

```
MongoDB + srv: // asabeneh: 123123123 @ 30daysofpython- twxkr.mongodb.net/test?retryWrites =True&w=majority
```

Maintenant, j'ai tout remplacé et le mot de passe est 123123 et le nom de la base de données est *thirty_days_python*. Ce n'est qu'un exemple, votre mot de passe doit être plus fort que l'exemple de mot de passe.

Python a besoin d'un pilote MongoDB pour accéder à la base de données MongoDB. Nous utiliserons *pymongo* avec *dnspython* pour connecter notre application avec la base MongoDB. À l'intérieur de votre répertoire de projet, installez Pymongo et Dnspython.

```
pip install pymongo dnspython
```

Le module "dnspython" doit être installé pour utiliser `mongodb + srv: // uris`. Le Dnspython est une boîte à outils DNS pour Python. Il prend en charge presque tous les types d'enregistrements.

Connexion de l'application Flask au cluster MongoDB

```
# Importons le ballon
```

```
De Flask Import Flask, render_template Import Os # Importation du système de système d'exploitation MongoDB_URI = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daySofpython -twxkr.mongodb.net/Test?retryWrites {v3 » = pymongo.mongoclient (mongodb_uri) print (client.list_database_names ())
```

```
app = flask (__ name__) si __name__  
== '__main__':  
    # Pour le déploiement, nous utilisons l'environnement  
    # pour le faire fonctionner pour la production et le développement  
    port = int (os.environ.get ("port", 5000))  
    app.run (debug = true, host = '0.0.0.0', port =)
```

Lorsque nous exécutons le code ci-dessus, nous obtenons les bases de données MongoDB par défaut.

```
['admin', 'local']
```

Création d'une base de données et d'une collection

Créons une base de données, une base de données et une collection dans MongoDB seront créées si elles n'existent pas. Créons un nom de base de données *thirty_days_of_python* et *students* Collection.

Pour créer une base de données:

```
db = client.name_of_databse # Nous pouvons créer une base de données comme celle-ci o  
u la deuxième voie db = client ['name_of_database']
```

```
# Importons le ballon
```

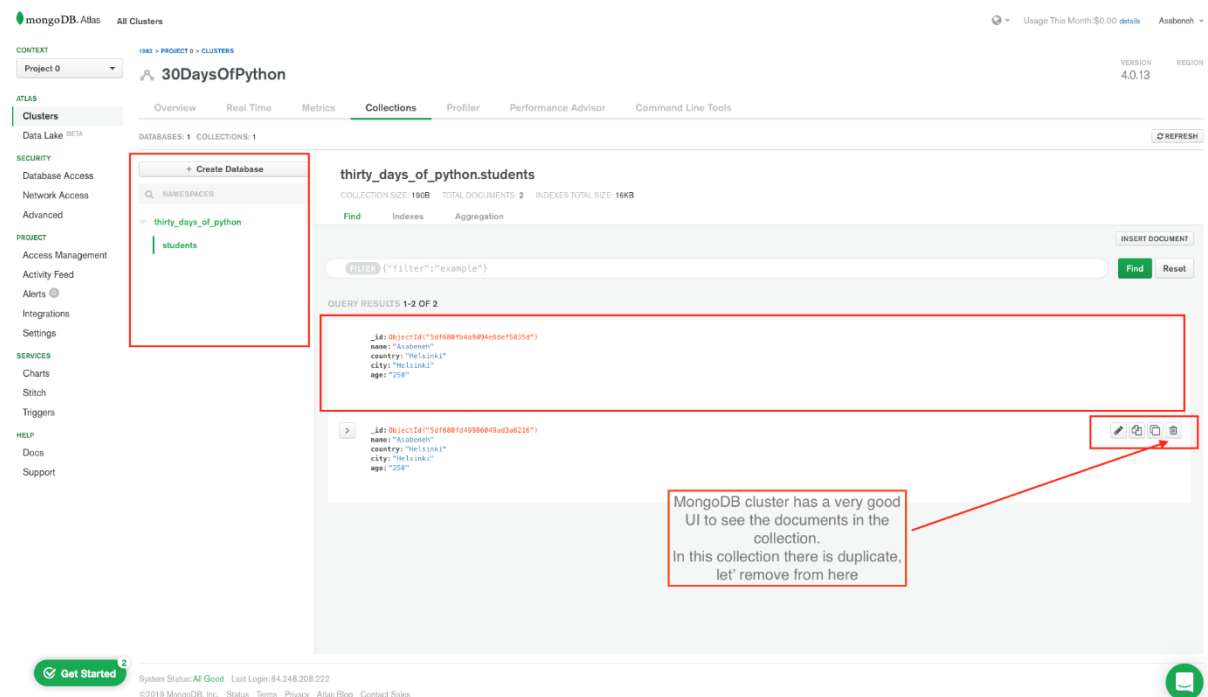
```
From Flask Import Flask, render_template Import Os # Importation du module du système d'exploitation mongodb_uri = 'mongodb + srv: // asabeneh: your_password_goes_here @ 30daysofpython -twxkr.mongodb.net/test?retryWrites =True&w=majority 'client = pymongo.mongoclient (mongodb_uri) # Création de la base de données DB = client.thirty_days_of_python # création d'étudiants Collection et insertion a un document A db.students.insert_one ({'name': 'asabeneh', 'country': 'Finland', 'City': 'Helsinki', 'Age': 250}) imprimer (client.list_database_names ()) App = FLASK (__ name__) Si __name__ == '__main__': # pour le déploie__ == " __MAIN__
```

```
# pour le faire fonctionner pour la production et le développement
port = int(os.environ.get("port", 5000))
app.run(debug = True, host = '0.0.0.0', port = port)
```

Après avoir créé une base de données, nous avons également créé une collection d'étudiants et nous avons utilisé la méthode `insert_one()` pour insérer un document. Maintenant, la collection de base de données `thirty_days_of_python` et `students` a été créée et le document a été inséré. Vérifiez votre cluster MongoDB et vous verrez à la fois la base de données et la collection. À l'intérieur de la collection, il y aura un document.

```
['trente_days_of_python', 'admin', 'local']
```

Si vous voyez cela sur le cluster MongoDB, cela signifie que vous avez réussi à créer une base de données et une collection.



Si vous avez vu sur la figure, le document a été créé avec un long identifiant qui agit comme une clé principale. Chaque fois que nous créons un document MongoDB, création et ID unique pour cela.

Insertion de nombreux documents à la collection

La méthode `insert_one()` insère un élément à la fois si nous voulons insérer de nombreux documents à la fois, nous utilisons la méthode `insert_many()` ou pour la boucle. Nous pouvons utiliser pour Loop pour insérer de nombreux documents à la fois.

```
# let's import the flask
```



```

De Flask Import Flask, render_template Import Os # Importation du système de système d'ex
ploitation MongoDB_URI = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30d
aySofpython -twxkr.mongodb.net/Test?retryWrites {v3 » = pymongo.mongoclient (mongodb
_uri)

```

```

étudiants = [
{'name': 'david', 'country': 'uk', 'ville': 'london', 'Âge': 34}, {'name': 'John', 'country': 'suédois', '
City': 'Stockholm', 'Age': '28'}, {'name': 'Sami', 'Country': 'Finland', 'City': 'Helsinki', 'Age
': 'Finland', 'City': 'Helsinki', 'Age': 'Country', pour les étudiants, pour les étudiants: 'Helsin
ki', 'Age': 25. db.students.insert_one (étudiant)

```

```

app = flask (__ name__) if __name__ == '__main__': # Pour le déploiement, nous utilis
ons l'environnement pour le faire fonctionner pour la production et le port de développe
ment = int (os.environ.get ("port", 5000)) app.run (debug = true, hôte = '0.0.0.0.0.0.0.
0.0.0.0.0.0.0., port =)

```

MONGODB FIND

Les méthodes *find()* et *findOne()* sont une méthode courante pour trouver des données dans une collection dans la base de données MongoDB. Il est similaire à l'instruction SELECT dans une base de données MySQL. Utilisons la méthode *find_one()* pour obtenir un document dans une collection de bases de données.

- * `find_one ({ "_id": ObjectId ("id") })`: obtient la première occurrence si un identifiant n'est pas fourni

```

# Importons le flacon à partir du flacon d'importation Flask, render_template Import OS # Im
portation du module du système d'exploitation mongodb_uri = 'mongodb + srv: // asabeneh: y
our_password_goes_here @ 30daysofpython -twxkr.mongodb.net/test?retryWrites =True&w
=majority 'client = pymongo.mongoclient (mongodb_uri) db = client ['Thirty_days_of_pytho
n'] # accédant à l'étudiant de la données = db.students.find_one () imprimer (étudiant)

```

```

App = Flask (__ Name__)

```

```
Si __name__ == '__main__': # Pour le déploiement, nous utilisons l'environnement # p
our le faire fonctionner pour le port de production et de développement = int (os.enviro
n.get ("port", 5000)) app.run (debug = true, host = '0.0.0', port = port))
```

```
{'_id': ObjectId ('5df68a21f106fe2d315bbc8b'), 'name': 'asabeneh', 'country': 'Helsinki', 'City':
'Helsinki', 'Age': 250}
```

La requête ci-dessus renvoie la première entrée, mais nous pouvons cibler un document spécifique à l'aide de `_id` spécifique. Faisons un exemple, utilisez l'ID de David pour obtenir un objet David. `'_id': ObjectId ('5DF68A23F106FE2D315BBC8C')`

```
# Importons le ballon à partir du flacon d'importation Flask, render_template Import OS # Mo
dule du système d'exploitation d'importation de BSON.ObjectId Import ObjectId # ID Object
Mongodb_uri = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daysofpython
-twokr.mongodb.net/test?retryWrites =True&w=majority 'client = pymongo.mongoclient (m
ongodb_uri) db = client ['Thirty_days_of_python '] # accédant à l'étudiant de la données = db
.students.find_one ({'_id': ObjectId ('5df68a23f106fe2d315bbc8c')}) imprimer (étudiant) app
= Flask (__ name__) Si __name__ == '__main__': # pour le déploiement, nous utilisons l'envi
ronnement # pour faire du travail informatique pour les deux pour la production et le port de d
éveloppement et le port de développement et le port de développement pour le déploiement. =
int (os.environ.get ("port", 5000)) app.run (debug = true, host = '0.0.0.0', port = port)
```

```
{'_id': ObjectId ('5DF68A23F106FE2D315BBC8C'), 'NAME': 'David', 'Country': 'UK', 'City': '
London', 'Age': 34}
```

Nous avons vu, comment utiliser `find_one()` en utilisant les exemples ci-dessus. Passons-en un à `find()`

- `find()`**: renvoie tout l'occurrence d'une collection si nous ne transmettons pas un objet de requête. L'objet est l'objet `Pymongo.Cursor`.

```
# Importons le ballon à partir du flacon d'importation FLASK, Rende
r_Template Importer OS # Module du système d'exploitation d'impor
tation
Mongodb_uri = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daysofpython
-twxkr.mongodb.net/test?retryWrites =True&w; DB = client ['trente_days_of_python'] # Acc
ès aux étudiants de la base de données = db.students.find () pour les étudiants en étudiants: im
primer (étudiant) app = flask (__ name__) if __name__ == '__main__':
```

```
# Pour le déploiement, nous utilisons l'environnement
# pour le faire fonctionner pour la production et le développement
port = int (os.environ.get ("port", 5000))
app.run (debug = true, host = '0.0.0.0', port = port)
```

```
{'_id': ObjectId ('5df68a21f106fe2d315bbc8b'), 'name': 'asabeneh', 'country': 'Finland', 'City': '
Helsinki', 'Age': 250} {'_id': ObjectId ('5df68a23f106fe2d315bbc8c'), '5df68a23f106f 'David',
```

```
'Country': 'UK', 'City': 'London', 'Age': 34} {'_id': ObjectId ('5DF68A23F106FE2D315BBC8
D'), 'name': 'John', 'Country': 'Sweden', 'City': 'Stockholm', 'Age': 28} {'_id': ObjectId ('5DF6
8A23F106FE2D315BBC8E'), 'name': 'Sami', 'Country': 'Finlande', 'City': 'Helsinki', 'Age': 25
}
```

Nous pouvons spécifier les champs à retourner en passant le deuxième objet dans le *find({}, {})*. 0 signifie ne pas inclure et 1 signifie inclure mais nous ne pouvons pas mélanger 0 et 1, sauf pour *_ID*.

```
# Importons le ballon à partir du flacon d'importation Flask, render_template Import OS # Im
portation du module du système d'exploitation mongodb_uri = 'mongodb + srv: // asabeneh: y
our_password_goes_here @ 30daysofpython -twxkr.mongodb.net/test?retrywrites =True&w
=majority 'client = pymongo.mongoclient (mongodb_uri) db = client [' thirty_days_of_pytho
n '] # accédant aux étudiants de la données = db.students.find ({}, {"_id": 0, "nom": 1, "pays"
: 1}) # 0 signifie ne pas inclure et 1 signifie inclure pour les étudiants dans les étudiants: impr
imer (étudiant)
```

```
app = flask(__name__) if __name__ == '__main__': # Pour le déploiement, nous utilis
ons l'environnement pour le faire fonctionner pour la production et le développement de
la production = int(os.environ.get("port", 5000)) app.run(debug = true, hôte = '0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.DIC port =)
```

```
{'name': 'asabeneh', 'country': 'Finlande'}  
{ «Nom»: «David», «Country»: «UK» }  
{'name': 'John', 'country': 'Suède'}  
{'name': 'sami', 'country': 'Finlande'}
```

Trouver avec une requête

Dans MongoDB, trouver un objet de requête. Nous pouvons passer un objet de requête et nous pouvons filtrer les documents que nous aimons filtrer.

```
# Importons le flacon à partir du flacon d'importation Flask, render_template Import OS # Im
portation du module du système d'exploitation mongodb_uri = 'mongodb + srv: // asabeneh: y
our_password_goes_here @ 30daysofpython -twxkr.mongodb.net/test?retryWrites =True&w
=majority 'client = pymongo.mongoclient (mongodb_uri) db = client [' Thirty_days_of_pytho
n '] # accédant à la dotabase Query = {"Country": "" = db.students.find (query) for Student in
Student: print (étudiant) app = flask (__ name__) if __name__ == '__main__': # pour le déplo
iement, nous utilisons l'environnement pour le faire fonctionner pour la production et le port d
e développement = int (OS.enViron.get ("", 5000)) app.run (debug = true, hôte = '0.0.0.0', por
t =)
```

```
{'_id': ObjectId('5df68a21f106fe2d315bbc8b'), 'name': 'Asabeneh', 'country': 'Finland', 'city': 'Helsinki', 'age': 25} {'_id': ObjectId('5df68a23f106fe2d315bbc8e'), 'name': 'Sami', 'Country': 'Finlande', 'City': 'Helsinki', 'Age': 25}
```

Requête avec des modificateurs

```
# Importons le ballon à partir du flacon d'importation FLASK, Rendre_Template Importation OS # Importation du module du système d'exploitation Importer Pymongo
```

```
MongoDB_URI = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daysofpython -twxkr.mongodb.net/test?retrywrites =True&w =majority' client = pymongo.mongoclient (mongoDB_URI) DB {v6. Client ['Thirty_Days_Of_Python'] # Accès à la requête de base de données = {"City": "Helsinki"} Students = db.students.find (Query) pour les étudiants dans les étudiants: imprimer (étudiant) App = Flask (__ Name__) Si __Name__ == "__main__": # pour la déploie. # pour le faire fonctionner pour le port de production et de développement = int (os.environ.get ("port", 5000)) app.run (debug = true, host = '0.0.0.0', port = port)
```

```
{'_id': ObjectId ('5df68a21f106fe2d315bbc8b'), 'name': «Asabeneh», «Country»: «Finlande», «City»: «Helsinki», «Age»: 250} {'_id': objectId ('5df68a23f106fe2d315bbc8e'), 'name': 'sami', 'Country': 'Finlande', 'City': 'Helsinki', 'Age': 25}
```

Trouvez la requête avec modificateur

```
# Importons le ballon à partir du flacon d'importation FLASK, Rende
r_Template Importer OS # Module du système d'exploitation d'impor
tation
import pymongo mongodb_uri = 'mongodb + srv: // asabeneh: your_password_goes_here @
30daysofpython -twxkr.mongodb.net/test?retrywrites {v3rue pymongo.mongoclient (mongod
b_uri) db = client ['the trente_days_of_python'] # accéder à la requête de base de données = {
"country": "Finland", "ville": "Helsinki"} Students = db.students.find (Query) for Student in
Student FLASK (__ name__) Si __ name__ == '__main__': # Pour le déploiement, nous utilis
ons l'environnement # pour le faire fonctionner pour la production et le développement du por
t = int (os.environ.get ("port", 5000)) app.run (debug = true, host = '0.0.0.0', port {V14
```

```
{'_id': ObjectId ('5df68a21f106fe2d315bbc8b'), 'name': 'asabeneh', 'country': 'Finlande', 'City': 'Helsinki', 'Age': 250}
```

```
{'_id': ObjectId ('5df68a23f106fe2d315bbc8e'), 'name': 'sami',
'Country': 'Finlande', 'City': 'Helsinki', 'Age': 25}
```

Requête avec des modificateurs

```
# Importons le ballon à partir du flacon d'importation FLASK, Rende
r_Template Importation OS # Importation du module du système d'ex
ploitation Importer Pymongo
```

```
MongoDB_URI =
'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daysofpython -twxkr.mongod
b.net/test?retrywrites =True&w =majority' client = pymongo.mongoCLIENT (Mongodb_uri)
Client ['Thirty_Days_Of_Python'] # Accès à la requête de base de données = {"Age": {"$ gt":
30}} Students = db.students.find (Query) pour les étudiants dans les étudiants:
```

Impression (étudiant)

```
app = flask(__name__) if __name__ == '__main__': # Pour le déploiement, nous utilis  
ons l'environnement pour le faire fonctionner pour la production et le développement de  
la production = int(os.environ.get("port", 5000)) app.run(debug = true, hôte = '0.0.0.0  
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.DIC port =)
```

```
{'_id': ObjectId('5df68a21f106fe2d315bbc8b'), 'name': 'asabeneh', 'country': 'Finlande', 'City': 'Helsinki', 'Age': 250}
```

```
{'_id': objectId('5df68a23f106fe2d315bbc8c'), 'name': 'david',
«Country»: «UK», «City»: «Londres», «Age»: 34}
```

```
# Importons le ballon à partir du flacon d'importation FLASK, Rende
r_Template Importation OS # Importation du module du système d'ex
ploitation Importer Pymongo
```

```
Mongodb_uri = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daysofpython
-twxkr.mongodb.net/test?retryWrites} Pymongo.Mongo.Mongo DB = Client ["Thirty_days_O
f_Python"] # Accès à la requête de base de données = {"Age": {"$ gt": 30}} Students = db.stu
dents.find (Query) pour les étudiants dans les étudiants: print (étudiant)
```

```
{'_id': ObjectId('5df68a23f106fe2d315bbc8d'), 'name': 'John', 'Country': 'Sweden', 'City': 'Stockholm', 'Age': 28} {'_id': ObjectId('5df68a23f106fe2d315bbc8e'), 'name': 'SAMI', 'Country': 'Finland', 'City': 'Helsinki', 'Age': 25}
```

Limiter les documents

Nous pouvons limiter le nombre de documents que nous renvoyons en utilisant la méthode *limit()*.

```
# Importons le ballon à partir du flacon d'importation FLASK, Rende
r_Template Importation OS # Importation du module du système d'ex
ploitation Importer Pymongo
```

```
Mongodb_uri = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daysofpython
-twxkr.mongodb.net/test?retryWrites = pymongo.mongoclient (mongodb_uri) db = client ['tre
nte_days_of_python'] # accéder à la base de données
```

```
db.students.find().limit(3)
```

Trouver avec tri

Par défaut, le tri est dans l'ordre croissant. Nous pouvons modifier le tri en ordre décroissant en ajoutant -1 paramètre.

```
# let's import the flask
from flask import Flask, render_template
import os # importing operating system module
import pymongo

MONGODB_URI =
'mongodb+srv://asabeneh:your_password_goes_here@30daysofpython
-twxkr.mongodb.net/test?retryWrites=true&w=majority'
client = pymongo.MongoClient(MONGODB_URI)
db = client['thirty_days_of_python'] # accessing the database
students = db.students.find().sort('name')
for student in students:
    print(student)

students = db.students.find().sort('name',-1)
for student in students:
    print(student)

students = db.students.find().sort('age')
for student in students:
    print(student)

students = db.students.find().sort('age',-1)
for student in students:
    print(student)

app = Flask(__name__)
if __name__ == '__main__':
    # for deployment we use the environ
    # to make it work for both production and development
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
```

Commande ascendante

```
{'_id': ObjectId('5df68a21f106fe2d315bbc8b'), 'name': 'asabeneh', 'country': 'Finland', 'City': 'Helsinki', 'Age': 250} {'_id': ObjectId('5df68a23f106fe2d315bbc8c'), 'name': 'David', 'Country': 'UK', 'City': 'London', 'Age': 34}
```



```
{'_id': ObjectId('5df68a23f106fe2d315bbc8d'), 'name': 'John', 'Country': 'Sweden', 'City': 'Stockholm', 'Age': 28} {'_id': ObjectId('5df68a23f106fe2d315bbc8e'), 'name': 'Sami', 'Country': 'Finland', 'City': 'Helsinki', 'Age': 25}
```

Ordre descendant

```
{'_id': ObjectId('5df68a23f106fe2d315bbc8e'), 'name': 'Sami', 'country': 'Finland', 'City': 'Helsinki', 'Age': 25} {'_id': ObjectId('5df68a23f106fe2d315bbc8d'), 'name': 'John', 'Country': 'Sweden', 'City': 'Stockholm', 'Age': 28} {'_id': ObjectId('5df68a23f106fe2d315bbc8c'), 'name': 'David', 'Country': 'UK', 'City': 'London', 'Age': 34} {'_id': ObjectId('5df68a23f106fe2d315bbc8b'), 'name': 'Asabeneh', 'Country': 'Finland', 'City': 'Helsinki', 'Age': 250}
```

Mise à jour avec la requête

Nous utiliserons la méthode `update_one()` pour mettre à jour un élément. Il faut deux objets que l'une est une requête et la seconde est le nouvel objet. La première personne, Asabeneh a eu un âge très invraisemblable. Mettons à jour l'âge d'Asabeneh.

```
# Importons le ballon à partir du flacon d'importation FLASK, Rendre
r_Template Importation OS # Importation du module du système d'ex
ploitation Importer Pymongo
```

```
Mongodb_uri = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daysofpython
-twokr.mongodb.net/test?retryWrites =true&w; DB = client ['trente_days_of_python'] # Accè
s à la base de données
```

```
Query = {'Age': 250} new_value = {'$ set': {'age': 38}} db.students.update_one (query,
new_value) # if if issing (student) application application = flask () __Name__ == '__
main__': # Pour le déploiement, nous utilisons l'environnement # pour le faire fonctionn
er pour le port de production et de développement = int (os.environ.get ("port", 5000))
```

```
app.run(debug = true, host = '0.0.0.0', port = port)
```

```
{'_id': ObjectId('5df68a21f106fe2d315bbc8b'), 'name': 'Asabeneh', 'country': 'Finland', 'City': 'Helsinki', 'Age': 38} {'_id': ObjectId('5df68a23f106fe2d315bbc8c'), '5df68a23f106fe2d315bbc8c'), '5df68a23f106fe2d315bbc8c'), '5df68a23f106fe2d315bbc8c'), '5df68a23f106fe2d315bbc8c'), '5df68a23f106f 'David', 'Country': 'UK', 'City': 'London', 'Age': 34} {'_id': ObjectId('5DF68A23F106FE2D315BBC8D'), 'name': 'John', 'Country': 'Sweden', 'City': 'Stockholm', 'Age': 28} {'_id': ObjectId('5DF68A23F106FE2D315BBC8E'), 'name': 'Sami', 'Country': 'Finlande', 'City': 'Helsinki', 'Age': 25}
```

Lorsque nous voulons mettre à jour de nombreux documents à la fois, nous utilisons la méthode `update_many()`.

Supprimer le document

La méthode `delete_one()` supprime un document. Le `delete_one()` prend un paramètre d'objet et de requête. Il supprime seulement la première occurrence. Retirons un John de la collection.

```
# Importons le ballon à partir du flacon d'importation Flask, render_template Import OS # Im
portation du module du système d'exploitation Importation Pymongo Mongodb_uri = 'Mongo
DB + srv: // asabeneh: your_password_goes_here @ 30daysofpython -twxkr.mongodb.net/tes
t?retryWrites =True&w=majority 'client = pymongo.mongoclient (mongodb_uri) db = client
[' Thirty_days_of_python ' ] # john' # '#' 'ESC a accès à la database
Query = { ' nom ' db.students.delete_one (query) pour l'étudiant dans db.students.find (): print
(étudiant) # Vérifiez le résultat si l'âge est modifié pour l'étudiant dans db.students.find (): im
primer (étudiant) App = flask (__ name__) Si __Name__ == '__main__': # pour le degré de tr
avail. à la fois le port de production et de développement = int (os.environ.get ("port", 5000))
app.run (debug = true, host = '0.0.0.0', port = port)
```

```
{'_id': ObjectId('5df68a21f106fe2d315bbc8b'), 'name': 'Asabeneh', 'country': 'Finland', 'City': 'Helsinki', 'Age': 38} {'_id': ObjectId('5df68a23f106fe2d315bbc8c'), '5df68a23f106fe2d315bbc8c'), '5df68a23f106fe2d315bbc8c'), '5df68a23f106fe2d315bbc8c'), '5df68a23f106fe2d315bbc8c'), '5df68a23f106f 'David', 'Country': 'UK', 'City': 'London', 'Age': 34} {'_id': ObjectId('5DF68A23F106FE2D315BBC8E'), 'name': 'Sami', 'Country': 'Finland', 'City': 'Helsinki', 'Age': 25}
```

Comme vous pouvez le voir, John a été retiré de la collection.

Lorsque nous voulons supprimer de nombreux documents que nous utilisons la méthode `delete_many()`, il faut un objet de requête. Si nous passons un objet de requête vide à `delete_many({})`, il supprimera tous les documents de la collection.

Faire tomber une collection

En utilisant la méthode `drop()`, nous pouvons supprimer une collection d'une base de données.

```
# Importons le ballon à partir du flacon d'importation FLASK, Rendre  
r_Template Importation OS # Importation du module du système d'ex  
ploitation Importer Pymongo
```

```
Mongodb_uri = 'MongoDB + srv: // asabeneh: your_password_goes_here @ 30daysofpython  
-twxkr.mongodb.net/test?retryWrites =true&w; db = client ['trente_days_of_python'] # accéder  
à la base de données db.students.drop ()
```

Maintenant, nous avons supprimé la collection des étudiants de la base de données.

Exercices: Jour 27

Félicitations!