


Effectuer des requêtes depuis python

pymongo

- **PyMongo** est une **bibliothèque Python** native contenant des outils pour travailler avec MongoDB,
- Pour installer la bibliothèque, il faut saisir la commande suivante dans un terminal, la commande c'est

pip install pymongo

python -m pip install pymongo



Invite de commandes

Microsoft Windows [version 10.0.19044.1826]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\> **pip install pymongo**

Collecting pymongo
 Downloading pymongo-4.2.0-cp310-cp310-win_amd64.whl (374 kB)
 | 374 kB 726 kB/s
Installing collected packages: pymongo
Successfully installed pymongo-4.2.0
C:\Users\>

Connexion via python au serveur MongoDB

La première étape consiste à créer une connexion avec le serveur MongoDB,

- Pour effectuer cette connexion on utilise la classe **MongoClient** de la librairie **pymongo**
- Puis passer comme paramètre à l'objet MongoClient la chaîne de connexion vers le serveur MongoDB. Pour cela vous avez plusieurs façons :
 1. `client = pymongo.MongoClient("mongodb://localhost:27017")`
 2. `client = pymongo.MongoClient(host='localhost', port=27017)`
 3. `client = pymongo.MongoClient()` # se connecte par défaut au serveur local
 4. `client=pymongo.MongoClient("mongodb://[user:password@]localhost:27017/"`
`)`

Exemple de connexion au serveur MongoDB

- Voici un exemple de code permettant de se connecter au serveur MongoDB local :

```
from pymongo import *  
client=MongoClient()  
print(client)
```

- Résultat:

```
MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)
```

- Autre exemple :

```
from pymongo import *  
client=MongoClient(host=['localhost'],port=27017)  
print(client)
```

Afficher la liste des bases de données du serveur

- Pour afficher la liste des base de données :

```
from pymongo import MongoClient  
client = MongoClient("mongodb://localhost:27017")  
print(client.list_database_names())
```

Connexion à une base de données

- Pour se connecter à la base de données Exemples du serveur objet de la connexion

```
from pymongo import *  
client=MongoClient(host=['localhost'],port=27017)  
Labase=client["bdsportif"]
```

Nom de la base à laquelle on se connecte

- Ou bien : Labase= `client.bdsportif`

Afficher les collections de la base de données

- Dans python on peut afficher les collections d'une base de données MongoDB on utilise la méthode `list_collection_names()` de l'objet Database.

```
from pymongo import MongoClient
client = MongoClient("mongodb://localhost:27017")
Labase= client['bdsportif']
print(Labase.list_collection_names())
```

Accéder à une collection de la base de données

- Pour accéder à une collection on peut utiliser les deux syntaxes suivantes :

```
Macollection = LaBase['NomDecollection']  
Macollection = LaBase.NomDecollection  
Macollection = LaBase.get_collection("nomDecollection")
```


Créer une base MongoDB

- Pour créer une base de données dans MongoDB, on utilise la même syntaxe pour accéder à une base de données MongoDB avec l'objet MongoClient, et MongoDB va créer la base de données si elle n'existe pas.

```
from pymongo import MongoClient
client = MongoClient("mongodb://localhost:27017")
Nombd = input("Donnez le nom de la base à créer: ")
listbases = client.list_database_names()
if Nombd in listbases:
    print("Cette base existe déjà .")
else:
    exemple = client[Nombd]
    print("La base a été créée !")
```

Créer une collection dans la base MongoDB

- Pour créer une collection dans une base de données MongoDB, on utilise la méthode :

```
ObjetBase.createCollection("NomDeCollection")
```

Requêtes de mise à jour (insertion)

- Pour insérer un document dans une collection on utilise les deux méthodes :
 - `NomCollection.insert_one ({object_1})` : pour insérer un seul document
 - `NomCollection.insert_many([{object_1},{object_2}, ...])` : pour insérer plusieurs documents.
- Exemple :

```
from pymongo import *
client=MongoClient(host=['localhost'],port=27017)
db=client["bdsportif"]
s={"_id":"sp9","nom":"said","prenom":"Aouita","genre":"homme","sport":{"description":"maraton"},
d=db.get_collection("sportif").insert_one(s)
```

Requêtes simple

- Pymongo utilise la même syntaxe que l'interface MongoDB:

```
db.nomDeLaCollection.requete()    // ou db.get_collection("nomDecollection").requete()
```

```
from pymongo import *
client=MongoClient(host=['localhost'],port=27017)
db=client["bdsportif"]
d=db.get_collection("sportif").find()
for x in d[:]:
    print(x)
```

```
from pymongo import *
client=MongoClient(host=['localhost'],port=27017)
db=client["bdsportif"]
d=db.get_collection("sportif").find()
for x in d[1:3]:
    print(x)
```

On peut
préciser les
indices de
départ et
d'arrivé

Requêtes simple

- Si on préfère travailler avec les listes, on peut convertir le résultat de `find()` en liste

```
from pymongo import *
client=MongoClient(host=['localhost'],port=27017)
db=client["bdsportif"]
d=db.get_collection("sportif").find()
l=list(d)

for x in l:    #parcourir toute la liste
    print(x)

for x in l[2:10]:    #parcourir une partie de la liste
    print(x)
```

Requêtes simple

- Utiliser la méthode find() avec une restriction et une projection comme suit:

```
db=client["bdsportif"]
restriction={"_id":"sp3"}
projection={"nom":1,"prenom":1}
d=db.get_collection("sportif").find(restriction,projection)

for x in d[:]:
    print(x)
```

Requête de mise à jour

- La syntaxe reste la même que l'interface **MongoDB**:

```
Client.BaseDeDonnee.Collection.requete()
```

Requête pymongo	Fonctionnement
insert_one()	Insertion d'un seul document
insert_many()	Insertion d'une liste de documents
delete_one()	Suppression d'un document
delete_many()	Suppression d'une liste de documents
update_one()	Modification d'un document
update_many()	Modification d'une liste de documents

Requêtes de mise à jour modification

- Il existe deux méthodes pour modifier les documents dans une base de données MongoDB :
 - **NomCollection.update_one({conditionselection}, {"\$set" : {donneesAmodifier}})** : modifier un seul document dans la collection.
 - **NomCollection.update_many({conditionselection}, {"\$set" : {donneesAmodifier}})** : modifier plusieurs documents dans la collection.

```
from pymongo import *
client=MongoClient(host=['localhost'],port=27017)
db=client["bdsportif"]
cond={"_id":"sp3"}
mod={"nom":"ELGOURCH","prenom":"MOHAMED","nbMedails":3}
db.get_collection("sportif").update_one(cond,{"$set":mod})

d=db.get_collection("sportif").find()

for x in d[:]:
    print(x)
```


Requête de mise à jour suppression

- Exemple :

```
from pymongo import *
client=MongoClient(host=['localhost'],port=27017)
db=client["bdsportif"]
cond={"_id":"sp3"}

db.get_collection("sportif").delete_one(cond)

d=db.get_collection("sportif").find()

for x in d[:]:
    print(x)
```

Requête d'agrégation

- Les requêtes d'agrégation ont pour but de faire des calculs simples (agrégats) sur toute la collection ou seulement sur certains groupes,
- La syntaxe reste la même que l'interface MongoDB

```
Client.BaseDeDonnee.Collection.aggregate()
```

Requête d'agrégation

- Exemple :
- Afficher le nombre maximum de médailles par genre.

```
from pymongo import *
client=MongoClient(host=['localhost'],port=27017)
db=client["bdsportif"]
operation1={"$group":{"_id":"$genre","PGMedails":{"$max":"$nbMedails"}}}
d=db.get_collection("sportif").aggregate([operation1])

dd=list(d)
print(dd)
```

Importer/exporter des données

- MongoDB propose deux manières d'import/export de la base de données:
 1. **mongoexport/mongoimport**
 2. **Mongodump/mongorestore**
- On s'intéresse plutôt à **mongoexport/mongoimport**:
 - mongoexport: utilisé pour exporter des données dans une Collection à un fichier (json, csv,..)
 - mongoimport: utilisé pour importer des données dans une Collection à partir d'un fichier (json, csv,..)
 - Pour utiliser mongoexport ou mongoimport, on doit installer les outils mongod

Exporter des données

- On peut exporter les données d'une collection à un fichier JSON ou un fichier CSV
- La syntaxe est assez simple :

```
mongoexport -d nomBaseDonnees -c nomDeLaCollection -o c://fichier.json
```

- Exemple:

```
mongoexport -d DBsportif -c sportif -o fichier.Sportjson
```

Importer les données

- On peut importer les données à partir d'un fichier JSON
- La syntaxe est assez simple :

```
mongoimport -d nomBaseDonnees -c nomDeLaCollection --file fichier.json
```

- Exemple :

```
mongoimport -d DBsportif -c sportif --file d://fichier.json
```