Name : Dhaval Gogri
Course : Advanced Application Programming
Quest No : Quest 7 (Mongo 2)

Basic Import and support methods

```
import pymongo
from pymongo import MongoClient

# Only for formatting purposes
def putSpaces(len):
    space = ""
    for i in range(0, 15 - len, 1):
        space = space + " "
    return space
```

Connecting to MongoDB

```
# Connect to MongoDB Client
client =
MongoClient('mongodb://dgogri:jc35uDs5@smgo7db01.smu.edu:27017/?authSource=dgogridb')

# Choose my database
db = client.dgogridb
```

- **PART A**

Printing Collections that I have in my Database

```
# Part : A
print("--------------------------PART A---------------------------------")
# Get collections from dgogridb database and print them
collections = db.collection_names(include_system_collections=False)
print("Printing Collections Below")
for col in collections:
    print(col)
print("\n")
```

```
Printing Collections Below
quest6
zipcodes
zipstates
zorro
```

Indexing the collection and printing the result

```
# With Index
print("-------------------------INDEXED---------------------------------")
#create index for zip codes
db.zipstates.create_index([("zip_code", pymongo.ASCENDING)])
zipCodeQuery = { "zip_code": 10463}

#Get Statistics info for the query for indexed.
withIndexStats = db.zipstates.find(zipCodeQuery, {"_id":
0}).explain()["executionStats"]
withIndexExecutionTime = withIndexStats["executionTimeMillis"]
withIndexDocumentSearched = withIndexStats["totalDocsExamined"]



# Get data from MongoDB based on query and print the results
for dataInDB in db.zipstates.find(zipCodeQuery, {"_id": 0}):
    print("City --> " + dataInDB["city"])
    print("State --> " + dataInDB["state"])


# Execution Time and Document Searched for Indexed
print("Execution Time : With Index --> " + str(withIndexExecutionTime))
print("Document Searched : With Index --> " + str(withIndexDocumentSearched))
print("\n")
```

```
-------------------------INDEXED---------------------------------
City --> Bronx
State --> NY
Execution Time : With Index --> 0
Document Searched : With Index --> 1
```

Without-Indexed collection and printing the result

```
#Without Index
print("-------------------------NOT INDEXED---------------------------------")
db.zipstates.drop_index([("zip_code", pymongo.ASCENDING)])
zipCodeQuery = { "zip_code": 10463}
withoutIndexStats = db.zipstates.find(zipCodeQuery, {"_id":
0}).explain()["executionStats"]
withoutIndexExecutionTime = withoutIndexStats["executionTimeMillis"]
withoutIndexDocumentSearched = withoutIndexStats["totalDocsExamined"]


#Get data from MongoDB based on query and print the results
for dataInDB in db.zipstates.find(zipCodeQuery, {"_id": 0}):
    print("City --> " + dataInDB["city"])
    print("State --> " + dataInDB["state"])


# Execution Time and Document Searched for Non-Indexed
print("Execution Time : Without Index --> " + str(withoutIndexExecutionTime))
print("Document Searched : Without Index --> " + str(withoutIndexDocumentSearched))
print("\n\n")
```

```
-------------------------NOT INDEXED---------------------------------
City --> Bronx
State --> NY
Execution Time : Without Index --> 40
Document Searched : Without Index --> 42741
```

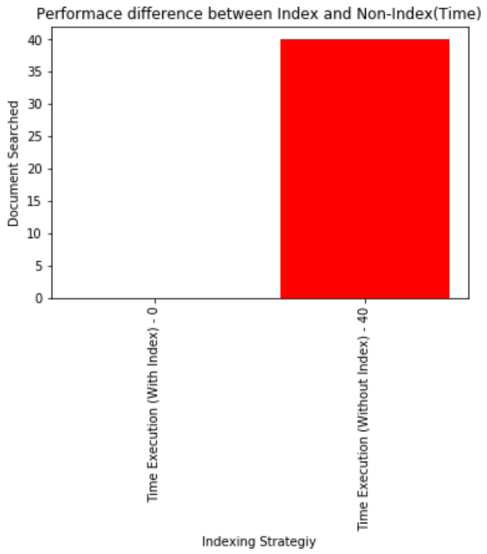
FINAL OUTPUT OF PART A

```
-------------------------PART A---------------------------------
Printing Collections Below
quest6
zipcodes
zipstates
zorro


-------------------------INDEXED---------------------------------
City --> Bronx
State --> NY
Execution Time : With Index --> 0
Document Searched : With Index --> 1


-------------------------NOT INDEXED---------------------------------
City --> Bronx
State --> NY
Execution Time : Without Index --> 40
Document Searched : Without Index --> 42741
```

GRAPH FOR COMPARISION

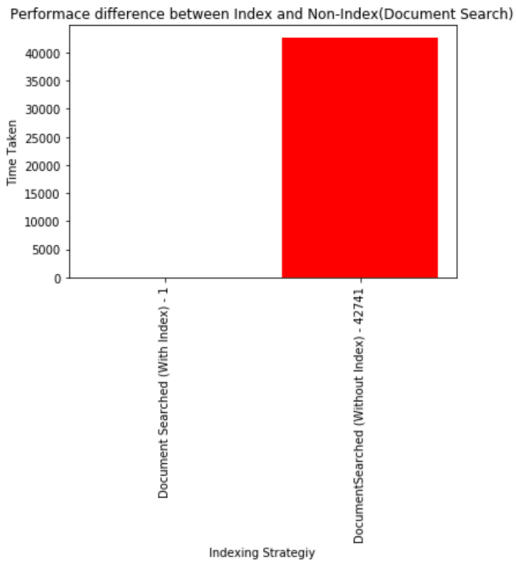
Time Comparision

With Index – 0 millisecond
Without Index – 40 millisecond



Document Searched

With Index – 1
Without Index – 42741



- **PART B**

```
#Part : B
print("--------------------------PART B---------------------------------")
print("------ finds the city and states for each of the zip codes in the list ------")

# Reading a Text File in Python
zipCodeTextFile = open("zipcodes.txt", "r")
# Split the data in the file based on ','
zipCodeTextFileText = zipCodeTextFile.read().replace(" ", "").split(",")


# Get all the city, state from database based on the the lat,long given in the
# txt file
print("\nZip" + putSpaces(3) + "City" + putSpaces(4) + "State")
for eachZipCode in zipCodeTextFileText:
    zipCityStateQuery = {"zip_code": int(eachZipCode)}
    queryResult = db.zipstates.find(zipCityStateQuery, {"_id": 0})
    if(queryResult.count() == 0):
        print("" + eachZipCode + putSpaces(len(str(eachZipCode))) + "Not a Valid Zip
Code")
    for dataInDB in db.zipstates.find(zipCityStateQuery, {"_id": 0}):
        print(""+eachZipCode+ putSpaces(len(str(eachZipCode))) + dataInDB["city"]+
putSpaces(len(str(dataInDB["city"]))) + dataInDB["state"])
print("\n\n")
```

OUTPUT

```
--------------------------PART B---------------------------------
------ finds the city and states for each of the zip codes in the list ------

Zip             City           State
10463           Bronx          NY
06520           New Haven      CT
00603           Aguadilla      PR
75225           Dallas         TX
90210           Beverly Hills  CA
99999           Not a Valid Zip Code
```

Q. Explaination of how I solved the challenge with related code snippets
Answer : I read the zip codes of the file as int when querying the database. Below is the code snippet.

```
zipCityStateQuery = {"zip_code": int(eachZipCode)}
```

- **PART C**

```
#Part : C
print("--------------------------PART C----------------------------------")
print("----- Use Mongo to find the closest zipcode, city and state for each of the
mystery gps locations. -----")

zipCodeMysteryTextFile = open("mysteryLatLong.txt", "r")
zipCodeMysteryTextFile = zipCodeMysteryTextFile.read().replace(" ", "").split(",")

# Variable initialization
# Store mystery lat and long
mysteryLatitude = []
mysteryLongitude = []
# Store closest place and dict in the below declared array
maxClosestPlace = []
maxClosestPlaceDataForCityAndState = []


# import math
from math import radians, sin, cos, asin, sqrt, fabs

# python function that computes the distance in km
def haversine(lon1, lat1, lon2, lat2):
    lon1, lat1, lon2, lat2 = map(radians, [float(lon1), float(lat1), float(lon2),
float(lat2)])
    dlon = lon2-lon1
    dlat = lat2-lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    km = 6371*c
    return km



# Store the mystery lat and long in the array
for i in range(0, len(zipCodeMysteryTextFile), 2):
    mysteryLatitude.append(zipCodeMysteryTextFile[i])
    mysteryLongitude.append(zipCodeMysteryTextFile[i + 1])

# initialize the array
for i in range(0, len(mysteryLatitude), 1):
    maxClosestPlace.append(100000000.0)
    maxClosestPlaceDataForCityAndState.append({})


# Find the closest city and state for the mystery lat, long.
for eachDataInZipState in db.zipstates.find({}):
    # Check if the lat long exist for the data
    if(eachDataInZipState["longitude"] and eachDataInZipState["latitude"]):
        for i in range(0, len(mysteryLatitude), 1):
            if (haversine(mysteryLongitude[i], mysteryLatitude[i],
eachDataInZipState["longitude"], eachDataInZipState["latitude"]) <
maxClosestPlace[i]):
                maxClosestPlaceDataForCityAndState[i] = eachDataInZipState
                maxClosestPlace[i] = haversine(mysteryLongitude[i],
mysteryLatitude[i], eachDataInZipState["longitude"], eachDataInZipState["latitude"])
```

```
# Print the data we get by comapring thr mystery lat, long and database information
print("\nMysteryLat1" + putSpaces(11) + "MysteryLon1" + putSpaces(11) + "ClosestCity"
+ putSpaces(11) + "State" + putSpaces(5) + "Lat" + putSpaces(3) + "Lon")
counter = 0
for eachPlace in maxClosestPlaceDataForCityAndState:
    mystLon = str(mysteryLongitude[counter]) +
putSpaces(len(str(mysteryLongitude[counter])))
    mystLat = str(mysteryLatitude[counter]) +
putSpaces(len(str(mysteryLatitude[counter])))

    city = str(eachPlace["city"]) + putSpaces(len(str(eachPlace["city"])))
    state = str(eachPlace["state"]) + putSpaces(len(str(eachPlace["state"])))

    lat = str(eachPlace["latitude"]) + putSpaces(len(str(eachPlace["latitude"])))
    lon = str(eachPlace["longitude"]) + putSpaces(len(str(eachPlace["longitude"])))

    print(mystLat + mystLon + city + state + lat + lon)
    counter = counter + 1
```

OUTPUT

```
-------------------------PART C---------------------------------
----- Use Mongo to find the closest zipcode, city and state for each of the mystery gps locations. -----

MysteryLat1    MysteryLon1    ClosestCity    State          Lat            Lon
40.8276        -73.92614      Bronx          NY             40.819329      -73.920355
41.94887       -87.65778      Chicago        IL             41.997247      -87.716621
32.7582813     -97.1105791    Arlington      TX             32.77408       -97.131689
```

Q. What the locations have in comman
Answer : All the locations are cities in USA.

ALL OUTPUT TOGETHER

```
-------------------------PART A---------------------------------
Printing Collections Below
quest6
zipcodes
zipstates
zorro


-------------------------INDEXED---------------------------------
City --> Bronx
State --> NY
Execution Time : With Index --> 0
Document Searched : With Index --> 1


-------------------------NOT INDEXED---------------------------------
City --> Bronx
State --> NY
Execution Time : Without Index --> 40
Document Searched : Without Index --> 42741



-------------------------PART B---------------------------------
------ finds the city and states for each of the zip codes in the list ------

Zip             City            State
10463           Bronx           NY
06520           New Haven       CT
00603           Aguadilla       PR
75225           Dallas          TX
90210           Beverly Hills   CA
99999           Not a Valid Zip Code



-------------------------PART C---------------------------------
----- Use Mongo to find the closest zipcode, city and state for each of the mystery gps locations. -----

MysteryLat1     MysteryLon1     ClosestCity     State           Lat             Lon
40.8276         -73.92614       Bronx           NY              40.819329       -73.920355
41.94887        -87.65778       Chicago         IL              41.997247       -87.716621
32.7582813      -97.1105791     Arlington       TX              32.77408        -97.131689
```