ANANDARASA Milojan
BICHAT Guillaume

REPORT MongoDB

API Velib' : https://opendata.paris.fr/api/records/1.0/search/?dataset=velib-disponibilite-en-temps-reel&rows=1400&facet=station_state&facet=kioskstate&facet=creditcard&facet=overflowactivation

First of all please connect MongoDB to your database. You can run the code on the python command prompt (Anaconda prompt for my case) or directly on the terminal of your IDE.

Menu:

A simple menu with 4 features, you have to enter your choice (between 1:4)

```
                        ##################################

                                  Data Velib':

                        ##################################


        1-Number of bikes available.
        2-Find data about a station.
        3-Top stations
        4-Exit

Your choice:
```

Number of bikes available:

Features that return the number of bikes currently available. We did this thanks to the aggregate function of pymongo.
We also display a graph to better visualize the data thanks to the matplotlib function.

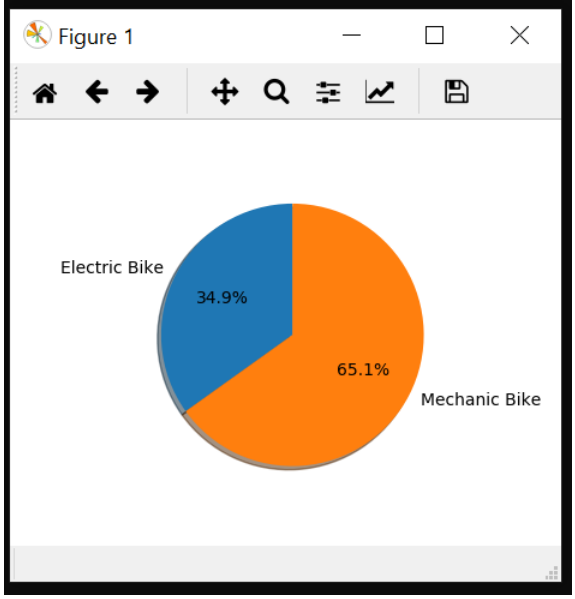
```
nb_velib=col.aggregate([{'$group':{'_id':"$datasetid",'total':{'$sum':"$fields.ebike"}}}])

nb_velib=col.aggregate([{'$group':{'_id':"$datasetid",'total':{'$sum':"$fields.mechanical"}}}])

plt.pie(bikes, labels=name, autopct='%1.1f%%', startangle=90, shadow=True)
plt.show()
```

```
Number of bikes avaible at :

  -Electric Bike: 5122
  -Mechanic Bike: 9559
  -Total: 14681
```



Find data about a station:

```
Enter the station name please: Charles de Gaulle

  City: Neuilly-sur-Seine
     Station name: Charles de Gaulle - Madrid
           Capacity: 22
           Docks avaible: 10
           Bikes avaible: 12
           Last update: 2020-03-19T13:06:13+00:00


  City: Boulogne-Billancourt
     Station name: Transvaal - Charles de Gaulle
           Capacity: 27
           Docks avaible: 3
           Bikes avaible: 24
           Last update: 2020-03-19T13:04:47+00:00


  City: Pantin
     Station name: Charles de Gaulle - Jean Lolive
           Capacity: 26
           Docks avaible: 24
           Bikes avaible: 2
           Last update: 2020-03-19T13:06:05+00:00
```

The functionality allows the user to enter the name of a station and to obtain the necessary data on this station. We use the function find in this case.

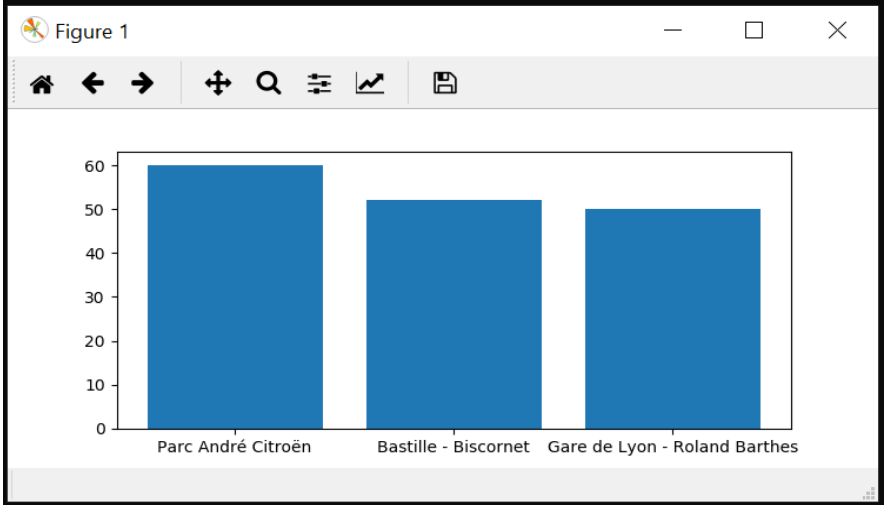
```
station=col.find({'fields.name':{'$regex': name_station}})
```

Top stations:

```
How many stations do you want to see? 3

  The top stations which the number of bikes avaible:

     -Parc André Citroën: 60
     -Bastille - Biscornet: 52
     -Gare de Lyon - Roland Barthes: 50
```



This function allows you to view the best stations with the highest number of bikes available. You must first indicate the number of stations you want to see and the program displays the data with a graph.

```
top=col.aggregate([{'$project':{'fields.name':1,'total':{'$add':["$fields.ebike","$fields.mechanical"]}}},{'$sort':{'total':-1}},{'$limit':int(nb)}])
```

```
plt.bar(name,bikes)
plt.show()
```

Update our database:

```
def Update():
    client = MongoClient()
    db = client.velib
    col = db.data
    for x in data_json['records']:   #New data
        y=col.find({'fields.stationcode': x['fields']['stationcode']})    #Old data
        new_date=datetime.datetime.strptime(x['fields']['duedate'],'%Y-%m-%dT%H:%M:%S%z')    #New date
        old_date=datetime.datetime.strptime(y[0]['fields']['duedate'],'%Y-%m-%dT%H:%M:%S%z')
        if (new_date>old_date):
            col.replace_one({'fields.stationcode':y[0]['fields']['stationcode']},x)
    client.close()


class MonThread (threading.Thread):
    def __init__(self,jusqua):
        threading.Thread.__init__(self)
        self.jusqua = jusqua

    def run(self):
        for i in range(0, self.jusqua):
            Update()
            time.sleep(30)   # attend 30 secondes sans rien faire

m = MonThread(10)          # crée le thread
m.start()                  # démarre le thread,
```

Our program updates the data automatically every 30 seconds without disturbing user requests. The function to recover the data via the API then compare them with the MongoDB database, to update only the necessary data.