Programming for IoT applications
Lab 3

SUGGESTION: Use **Postman**, a chrome plugin, for testing REST web services by managing HTTP requests

---

Exercise 1. Develop a **client** python application for invoking the RESTful calculator developed in *Exercise_1* during the *Lab 2* that:

- asks to end-users the operation to be performed and two operands
- invokes the web services published by the RESTful
- shows the results to end-users in a friendly way (NOT the full JSON)

SUGGESTION: use the library *requests* to open ULRs and read their contents (take care in handling exceptions)

Exercise 2. Extend *Exercise_3* proposed during the *Lab 1*, for designing a RESTful-style discography manager.
Identify what are the proper HTTP methods (among GET, POST, PUT and DELETE) and develop the web services to:

1. **Print the whole discography**
2. **Search by artist** and print the resulting entry (or entries) if exist(s) otherwise print an error message with proper HTTP-status code
3. **Search by title** and print the resulting entry (or entries) if exist(s) otherwise print an error message with proper HTTP-status code
4. **Search by publication** year and print the resulting entry (or entries) if exist(s) otherwise print an error message with proper HTTP-status code
5. **Search by total tracks** and print the resulting entry (or entries) if exist(s) otherwise print an error message with proper HTTP-status code
6. **Insert a new album**
7. **Delete an existing album**

Then, develop also the **client** python application for managing the discography by invoking the web services provided by the RESTful-style discography manager

Validate the JSON with http://jsonlint.com/

SUGGESTION: use the library *requests* to open ULRs and read their contents (take care in handling exceptions)

Exercise 3. Develop your REST web services for getting information from third-party web services for the real bike sharing in Barcelona (Spain).

Example of bike sharing JSON:


```
[
   {
      "id": "1",
      "district": "2",
      "lon": "2.180042",
      "lat": "41.397952",
      "bikes": "20",
      "slots": "2",
      "zip": "08013",
      "address": "Gran Via Corts Catalanes",
      "addressNumber": "760",
      "nearbyStations": "24,369,387,426",
      "status": "OPN",
      "name": "01 - C/ GRAN VIA CORTS CATALANES 760",
      "stationType": "BIKE"
   },{
      "id": "496",
      "district": "2",
      "lon": "2.175141",
      "lat": "41.404871",
      "bikes": "12",
      "slots": "12",
      "zip": "08025",
      "address": "C/ DE PROVENÇA",
      "addressNumber": "445",
      "nearbyStations": "452,475",
      "status": "OPN",
      "name": "496 - C/ DE PROVENÇA, 445 ",
      "stationType": "ELECTRIC_BIKE"
   }
]
```


In detail, the application needs to get in real-time the information from https://www.bicing.cat/availability_map/getJsonObject and provide web services to:

1. Order the bike-stations by available *"slots"* and display first N stations. The user can also choose to have the results in ascending or descending order (by default descending).
   The parameters for this web service are:
   - **N:** number of stations to display. It is optional and by default N=10.
   - **order:** It is optional and the default value is descending

2. Order the bike-stations by available *"bikes"* and display first N stations. The user can also choose to have the results in ascending or descending order (by default descending).

The parameters for this web service are:
    - **N:** number of stations to display. It is optional and by default N=10.
    - **order:** It is optional and the default value is descending

3. Get all the bike-stations with a zip-code provided by users.
   The parameter for this web service is:
    - **zip-code** (mandatory).

4. Get all the bike-stations with more than N available *"bikes"* for the *"stationType" = "ELECTRIC_BIKE"*.
   The parameter for this web service is:
    - **N:** number of available bikes. It is optional and by default N=10.

5. Count all the available *"bikes"* and all the *"slots"* in a *"district"*.
   The parameter for this web service is:
    - ***"district"***: it identifies the district with a number.

Each web service must verify if the parameters are correct otherwise rise an HTTP error with the right HTTP status code.

All the Web Services responses should be in JSON. Validate it with http://jsonlint.com/

SUGGESTION: use the library *requests* to open ULRs and read their contents (take care in handling exceptions)