TASK1:
SETUP BIMServer and UPLOAD AN IFC FILE VIA BIMsie API.

1) Open **Terminal** (for Mac or Ubuntu) or **Command Window** (for Windows)

2) Go to the file directory by typing in Terminal:
`cd (drag and drop the "PythonClient" folder)`
and hit Enter

3) Now you are in "PythonClient" directory. To setup/start the BIMServer type:
`python JAR_run.py`
This command should open the BIMserver starter. If you have an error like: `No module named subprocess` type `pip install subprocess` this command will install the required module. After the starter appears click the "Start" button and wait for BIMserver setup. In the log page when you see "Server started successfully", it means the server is ready to use. Click the "Launch Webbrowser" button to open the BIMServer page

4) After the installation BIMserver will ask an administrative username and password. For common activity during the demo, type username and password like:
`admin@bimserver.org`
`admin`
and click "Setup" button

5) Now we should connect to BIMserver with Python Client API. In Terminal type:
`cd json_REST`
Now you are in the API functions folder. After that type
`python LoginServer.py`
This command will connect the Python Client API with BIMserver. Please pay attention to the new created "*login.json"* file. In this file, you will see a long string. This is you unique login id, in other words a **Token**. This token will be used for other functionalities. And we will retrieve the token from this created "*login.json"* file.

6) Now we need to define a project name in BIMServer. To create a new project type:
`python addProject.py`
This command will define a new project to BIMServer called "*ClassDemo_Project"* by default. If you want to change the project name, simply edit the Line17 in the *addProject.py*. After running this command, please pay attention to the new created "*addProject.json"* file. This JSON file is the response of the API which includes a unique project object identification **(poid)** number which we will use for the next functions.

7) After adding the project, next step is uploading an IFC file to BIMserver. To upload IFC file, we will use "*checkinIFC.py*" file. Before running the function, it is better to see the content of this file. Open the "*checkinIFC.py*" in your text editor. As you can see in the function, it requires the **Token, poid** and **deserializerOID** (line 14-22). The **token** is retrieved from the returned JSON file which is generated before. But **deserializerOID and poid** should be generated. Deserializer means the file format converter in BIMserver. To generate the **deserializerOID** type in Terminal:
`python getAllDeserializers.py`
This command will generate the *"allDeserializers.json"* file. In this file, all defined deserializers in BIMServer are listed. We are going to upload an IFC2x3 file. Therefore, the deserializer should be the same. Line22 in "*checkinIFC.py*" selects the **deserializerOID** and pass this as a parameter.
To get the **poid** type
`python getAllProjects.py`
This command will generate "*allProjects.json*" file. In this file, all defined projects are listed in a JSON structure. Since we create only "*ClassDemo_Project*" , you will see only this project. The **oid** will be retrieved from this file. Line17-18 do this job.

8) Now we got all initial parameters **token, poid** and **deserializerOID** to upload our IFC file. In BIMserver, uploading an IFC file is a two-step process. First, we should pass the above parameters with the file. Line34-50 do this job. As a response from API, we receive a **TopicId** from generated "*checkinifc.json*" file. TopicId is generally used to upload, delete and/or retrieve data to/from BIMserver. This Id should be passed to another API function called "*getProgress".* From Line62, you can see how the **TopicId** is passed to upload the IFC file. At the end of the function. The IFC file should be uploaded to BIMServer. To check, open BIMServer Browser and simply refresh the page. Depends on the IFC file size, it may take 1-2 minutes. To start the uploading process type in Terminal:
`python checkinIFC.py`

Congratulations! You have uploaded an IFC file to BIMserver without using the user interface. Now you are able to write functions to retrieve some data from BIMServer.

TASK2:
GET THE DETAILED INFORMATION OF AN BUILDING ENTITY BASED-ON ITS GUID NUMBER

Think that your manager asked you to get the detailed information of a specific building entity. The only thing you know about this building entity is the GUID number of it. The **GUID Number** of our entity is "**2g$QZpOGbBUxSNx_ZgxGCP**" (ignore the quotation marks). How do you get the detailed information?

1. Open the "*getDataByGUID.py"* in the Sublime Text editor to see the source code.

2. To get the data by GUID, you need to use "*getDataObjectByGuid"* function of the API. This function **Revision Object Identification (roid)** and the **guid** parameters.
3. We already know the **guid** number of our entity which is **2g$QZpOGbBUxSNx_ZgxGCP**
4. Then we need to find the **roid** number of our IFC model. To get this number type following to Terminal:
   `python allRevisions.py`
   This command will return the "*allRevisions.json"* in which the **roid** parameter will be retrieved and passed.
5. After creating the **roid** number, we are ready to get the detailed data of our building element. To get this information, type following to Terminal:
   `python getDataByGUID.py`
   This command will generate the "*getByGUID.json"* file in which all the detailed information of our building entity is presented in a JSON structure.

   This task demonstrates one of the most basic functionality performed in Revit, ArchiCAD or any other BIM software applications.
   In these software applications, when the user clicks one element of the 3D model, the software captures the GUID number, run a function similar like above, and parse the data in their UI.