Using a Web API with Python (Remote Procedure Invocation)

The four main service integration styles are

1. File Transfer Integration,
2. Integration through a Shared Database,
3. Remote procedure invocation (Web API), and
4. Message Queue.

This exercise shows how you can use a Web API with Python. Most Web APIs use a REST (REpresentational State Transfer) style, communicate over HTTP (Hypertext Transfer Protocol), and are using JSON (JavaScript Object Notation) as a message format.

1. https://randomuser.me/ is a random user generator
2. It has an API. Go to https://api.randomuser.me/ with your browser
3. You get a JSON back. Refresh the browser
4. You can add query parameters. There are added to the URL with a `?` . You then add the parameter with `parameter=value`. If you have more than one parameter, then the parameters are connected with a `&` e.g.
   ```
   http://api.open-notify.org/iss-pass.json?lat=37.78&lon=-122.41
   ```
5. Go the documentation https://randomuser.me/documentation and check how you can get multiple users and how to specify constraints on the output
6. Go to your browser and add parameters to https://api.randomuser.me/ so that you get 5 results of only males from the US
7. You can also get the data from the command line. Open the command line and write

   ```
   curl -s https://api.randomuser.me/
   ```

8. Now we want to analyze this data with Python
9. Open a Jupyter Notebook and create a new notebook
10. Import the two libraries `requests` and `json`
11. You can find the documentation for the requests package here: http://www.python-requests.org/en/latest/
12. With the requests package you can call a Web API with the URL and the method get

    ```
    response = requests.get("https://api.randomuser.me/")
    ```

13. Print the status code of the request
    ```
    print(response.status_code)
    ```

14. The meanings of the status codes are:

    - 200 – everything went okay, and the result has been returned (if any)
    - 301 – the server is redirecting you to a different endpoint. This can happen when a company switches domain names, or an endpoint name is changed.
    - 401 – the server thinks you're not authenticated. This happens when you don't send the right credentials to access an API.
    - 400 – the server thinks you made a bad request. This can happen when you don't send along the right data, among other things.
    - 403 – the resource you're trying to access is forbidden – you don't have the right permissions to see it.
    - 404 – the resource you tried to access wasn't found on the server.

15. Get with the request method 10 results of only males from the US. You can specify the query parameters with a Python dictionary like this:

```
parameters = {"lat": 37.78, "lon": -122.41}
```

16. And pass the parameter to the request like this

```
response = requests.get("http://api.open-notify.org/iss-pass.json",
params=parameters)
```

17. Alternatively, you could build the URL also by yourself
18. You can show the result of the request with the method text

```
response.text
```

19. You can convert the data from JSON to a Python dictionary with the package JSON

```
data = json.loads(response.text)
```

20. Check the type of variable data
21. Loop through the dictionary and print all first names
22. Print out all the names of the astronauts who are right now in space. You get the information about the Web APU from here http://open-notify.org/Open-Notify-API/People-In-Space/
23. Print the number of people that are right now in space
24. Print all names
25. Create a small **Streamlit** application, that creates a dynamic website, that shows the total number of people in space as well as all the names of the people. The app should use the API to get the names and display the names. Add a descriptive title and a short description (you can use the Streamlit markdown method).
26. Add to your Streamlit app the following functionality:
    a. Get the current location of the International Space Station (ISS) http://open-notify.org/Open-Notify-API/ISS-Location-Now/
    b. Create a geographical map in you Streamlit app that visualizes the current ISS location. There are a lot of different Python packages for geo mapping. E.g.
        i. PyDeck (included in Streamlit) https://docs.streamlit.io/en/latest/api.html#streamlit.pydeck_chart
        ii. Folium http://python-visualization.github.io/folium/
        iii. Cartopy https://pypi.org/project/Cartopy/
    c. Add a short description for your map
27. Deploy your Streamlit app with CapRover to your VPS. Create for your application a new subdomain **iss** so that your URL is something like iss.example.com. Enable HTTPS
28. Copy-paste the URL of your deployed Streamlit app to Moodle
29. What we have not covered in this exercise is authentication:
    a. A lot of Web APIs require a key for interacting with them (like Twitter, Facebook, …). You find at http://www.python-requests.org/en/latest/user/authentication/ more information for Authentication for Web APIs with the request package
    b. There are also special Python packages for interacting with services. E.g. for Twitter: https://pypi.python.org/pypi/twitter that handle also the authentication