Python and R LAB
University LUISS Guido Carli
Master of Science in Data Science and Management

-Art Institute of Chicago API-

Task 1

Group members: *Cantelmo Carlotta*, *Imperatore Claudia*, *Lona Masriera Marian*

This project is built for the course of *Python and R LAB* at University LUISS Guido Carli.
For the first task we have been asked to collect data by accessing the API (Application Programming Interfaces) of third-party services and create a full dataset.

For the task, we decided to work with the API provided by the Art Institute of Chicago. The documentation for the API can be accessed at the following link: Documentation | Art Institute of Chicago API (artic.edu). The API provides JSON-formatted data as a REST-style and it is a collector of all the possible information regarding the physical and virtual collection and information of the Art Institute.
For the project we decided to work with the artworks' collection data. The collection is composed by 115506 objects and each artworks has 93 variables associated. We decided to retrieve 1100 artworks. In order to do so, we used the main endpoint **/artworks,** resulting in the following API: https://api.artic.edu/api/v1/artworks.

Note: API'S listing and search endpoint are paginated, in the next section we are going to explain how we overcame it. More about the pagination at: Documentation | Art Institute of Chicago API (artic.edu)

Description of the script

As a first step, in the first three lines of the script we have imported the three main libraries needed for the purpose of task 1 (data collection task).

```
1. Import json
```

The first library is the **JSON** (JavaScript Object Notation) module which is mainly used to convert the Python dictionary into a JSON string that can be written into a file. It is a lightweight data format used for data interchange between different languages.

2. `Import pandas as pd`

The second library is the **Pandas** library which is used for data manipulation and analysis. It is a very powerful and versatile data analysis library that expedites the pre-processing steps of data science projects.

3. `Import request`

The last library is the **Request** module which allows to send HTTP requests and returns a Response Object with all the response data.

We have chosen the Art Institute of Chicago API's. As stated before, the API is built with pagination, controlled by the query parameters *page* and *limit.* The maximum number of records to retrieve per page allowed is set to 100 and this is stated by the code '=limit 100'.

https://api.artic.edu/api/v1/artworks?page=1&limit=100: example of API, it retrieves 100 records for page 1.

We are asked to build a data set with at least 1000 observations and for this purpose we proceeded in the following way:

```
4.  Num_of_pages = 12
5.  datafr = []
6.   for n in range(1, num_of_pages):
7.      url = "https://api.artic.edu/api/v1/artworks?page=" + str(n) + "&limit=100"
8.      response = requests.get(url)
9.      json_data = response.json()
10.     for d in json_data['data']:
11.         datafr.append(d)
```

We have created a variable 'Num_of_pages' assigning the integer 12 for the reason stated above. [4]
We then proceeded building a nested 'for loop' in order to retrieve the data from the API, stated in [7] as a string named *url*, and placing them into an empty list called 'datafr' [5].

We did so by iterating the page numbers within the URL exactly 11 times (using an in *range() function*), which corresponds to the number of pages necessary for the number of observations to be 1100. The values in the *range* function are integers, in order to implement them into the *url* we transform them in strings. [7]
We then use the command *get* from the library *request* so that all form data is encoded into the URL, appended to the action URL as query string parameters. [8]
Subsequently, the data retrieved from the URL are red through the *.json* function and are assigned to the variable *json_data. [9]*
At this stage, our data of interest are stored in a dictionary format in the list *'data'* in the *json_data* dictionary (the structure of the JSON corresponds to a python dictionary).
In order to extract them, we built another '*for loop*' that extracts the dictionaries in the *'data*' list and adds them through the function *"append()"* to our empty list *'datafr'.*
This process is repeated for every page of the URL.

At the end of the loops, *datafr* will be a list containing 1100 dictionaries.

In the last part of our code, we had to transform our list ‘datafr’ in a csv file:

```
12. csv_file_path = "art1000.csv"
13. df = pd.DataFrame(datafr)
14. df.fillna(0,inplace = True)
15. df.to_csv(csv_file_path, header=True, index=False)
```

First, we called ‘art1000.csv’ the final .csv document [12]; then, we used the Pandas library and the function pd.DataFrame() to construct a DataFrame from the 'datafr’ list.
In [14] we have also cleaned the dataset by inserting a zero in the boxes with 'not available values' trough the *.fillna* function.
Eventually, we transformed the resulted data frame into a csv file through the *.to_csv* function.

The resulting csv file is a dataset of 1100 x 93 where the rows represent the single artworks (1100) and the columns represent the variables (93) as retrieved from the API. The meaning of the variables and their type can be found at the following link: https://api.artic.edu/docs/#collections-2.
We would suggest to carefully inspect the dataset and the documentation before carrying out any analyses.