这个作业使你有机会练习在Python结构中编写涉及数据收集和分析的完整应用程序。重点是使用在本课程中获得的所有知识，从基本的Python到Pandas，使用JSON，创建图形等。

**内容**

我们希望通过分析存储库和开发人员来了解 GitHub 的背景。 你的程序必须使用 [GitHub REST API version 3](https://developer.github.com/v3/)来收集有关存储库、拉取请求和用户的信息。

由于对软件开发的社交方面感兴趣，我们还将从他们的 GitHub 个人资料中收集数据。

这项作业的第一个要求是你需要使用面向对象来处理正在收集和分析的数据。 最简单的方法是分析 json 字典结构。 当有嵌套元素时，可能需要一个新对象。

关于 GitHub 存储库，我们希望存储名称、所有者（用户名）、描述、主页、许可证（可以是另一个类）、分叉数、关注者数和 date\_of\_collection（收集数据的日期）。 当请求打印对象时，它应该是这样的：

●所有者/项目名称：描述（# of stars）

每个存储库还需要与拉取请求列表相关。 因此，对于每个存储库，收集在查询的第一页中返回的拉取请求，如下所示：

●https://api.github.com/search/issues?q=is:pr+repo:jabref/jabref（以存储库jabref/jabref为例）

对于每个拉取请求，需要保留：标题、编号、正文、状态、创建日期 (created\_at)、结束日期（如果状态与打开状态不同）、用户、提交次数、添加、删除、更改文件。

●对于最后4个字段，将需要使用以下格式进行另一个查询（使用之前找到的拉取请求的数量）：

●https://api.github.com/repos/JabRef/jabref/pulls/5531

然后，对于你在拉取请求中找到的每个作者（用户），需要保留：登录名和拉取请求数量（计算得出）。 你还需要从 GitHub 上的用户个人资料页面抓取（使用网络抓取库！）以下信息：存储库数量、关注者数量、关注数量、最近一年的贡献数量。 如果有重复用户，只需要更新pull requests（拉去请求）的数量

你需要有一个名为 to\_CSV 的函数，可以重用该函数将任何对象转换为 csv 条目（行）。 还需要提供文件名和对象。 如果该文件不存在，则需要创建该文件（带有标头）。 如果该文件存在，需要在 CSV 中添加一个包含该对象的新行。 为了实现这一点，你需要在每个类中都有一个具有相同名称的方法，该方法将返回一个字符串，其中包含已结构化为 CSV 的数据。

使用此函数创建/更新文件如下（无重复条目）：

●当从存储库收集数据时，需要将其添加到名为projects.csv的CSV中

●当收集存储库的拉取请求时，需要将它们存储在以所有者和存储库名称命名的文件中（projects/owner-project.csv）

●当从用户收集数据时，需要将其添加到名为users.csv的CSV中

**给用户的功能**

用户能够：

●请求系统收集特定存储库的数据（来自GitHub）。 通过提供所有者和存储库名称，程序需要开始收集所有内容（存储库、拉取请求、用户——包括抓取的数据）

●列出所有收集到的存储库

●列出来自存储库的所有拉取请求（请列出存储库以帮助用户提供现有选项）

●列出存储库的摘要，其中包含：

○处于打开状态的拉取请求数量

○处于关闭状态的拉取请求数量

○用户数

○最早请求拉取的日期

●根据存储库创建图形：

○箱型图，根据提交数量比较关闭和开放拉取请求

○箱型图，在添加和删除方面比较封闭式和开放式拉取请求

○箱线图，按作者关联分组的更改文件数

○散点图：添加x删除数

○直方图：每个拉取请求的提交数量

●考虑来自所有存储库的所有拉取请求创建图形：

○折线图，显示每天的拉取请求总数

○折线图，比较每天打开和关闭的拉取请求数量

○条形图，每个存储库的用户数量

○直方图，每个拉取请求的提交数量

●计算为用户收集的数据之间的相关性（关注者、关注者、拉取请求数量、贡献数量等）

●计算存储库拉取请求中所有数字数据之间的相关性

**测试**

你需要为代码编写至少 5 个单元测试。

This assignment will give you some practice with writing complete applications that deal with data collection and analysis in Python structures. The focus will be on using all the knowledge acquired in this course, from basic Python to Pandas, using JSON, creating graphics, etc.

## **Context**

We want to understand the context of GitHub by analyzing repositories and developers who contribute. Your program will have to consume the [GitHub REST API version 3](https://developer.github.com/v3/), to collect information about the repo, pull requests, and users.

Since we are interested in the social side of software development, we will also collect data from their GitHub profile.

The first requirement for this assignment is that **you need to use object orientation** to deal with the data you are collecting and analyzing. The easiest way to do this is by analyzing the json dictionary structure. When you have nested elements, a new object *may* be necessary.

About the GitHub repositories, we would like to store the name, owner (username), description, homepage, license (can be another class), number of forks, watchers, and date\_of\_collection (the date you collected the data). When you request to print the object it should be like this:

* Owner/ProjectName: Description (# of stars)

Each repository also needs to be related to a list of pull requests. Thus, for each repos, collect the pull requests that are returned in the first page of a query like this:

* https://api.github.com/search/issues?q=is:pr+repo:jabref/jabref (using repository jabref/jabref as an example)

For each pull request you need to keep: title, number, body, state, date of creation (created\_at), closing date (if the state is different than open), user, number of commits, additions, deletions, changed\_files.

* For the last 4 fields, you will need to make another query using the following format (using the number of the pull requests you found before):
* https://api.github.com/repos/JabRef/jabref/pulls/5531

Then, for each author (user) you find in the pull requests you need to keep: login and number of pull requests (calculated). You are also required to ***scrape*** (using the web scraping library!) the following information from the user profile page on GitHub: Number of Repositories, Number of Followers, Number of Following, Number of contributions in the last year. If you have repeated users, you only need to update the number of pull requests

You need to have a function called to\_CSV that can be reused to convert any object to a csv entry (row). You need to provide the file name and the object. If the file does not exist, you need to create the file (with the header). If the file exists, you need to append a new line with the object in the CSV. To make it possible, you will need to have a method in each of your classes with the very same name, which will return a string with the data already structured as a CSV.

Use this function to create/update the files as following (NO REPEATED ENTRIES):

* when you collect data from a repo, you need to add it to a CSV called projects.csv
* when you collect the pull requests of a repo, you need to store them in a file named after the owner and the name of repo (projects/owner-project.csv)
* when you collect data from users, you need to add it to a CSV called users.csv

## **Functions to the user**

A user may be able to:

* request the system to collect data for a specific repository (from GitHub). By providing the owner and repository name, your program needs to start the collection of everything (repository, pull request, users -- including scraped data)
* list all repos collected
* list all pull requests from a repo (please list the repos to help the user giving an existing option)
* list the summary of a repo, containing:
  + number of pull requests in open state
  + number of pull requests in closed state
  + number of users
  + date of the oldest pull requested
* create graphics given a repo:
  + boxplot comparing closed and open pull requests in terms of number of commits
  + boxplot comparing closed and open pull requests in terms of additions and deletions
  + boxplot comparing the number of changed files grouped by the author association
  + scatterplot: additions x deletions
  + histogram: number of commits per pull request
* create graphics considering ALL pull requests from ALL repos:
  + line graph showing the total number of pull requests per day
  + line graph comparing number of open and closed pull requests per day
  + bars comparing the number of users per repo
  + histogram: number of commits per pull request
* calculate the correlation between the data collected for a user (following, followers, number of pull requests, number of contributions, etc.)
* calculate the correlation between all the numeric data in the pull requests for a repo

## **Tests**

You need to write at least 5 unit tests for your code.