**1. Scala's real-world project repository data**

With almost 30k commits and a history spanning over ten years, Scala is a mature programming language. It is a general-purpose programming language that has recently become another prominent language for data scientists.

Scala is also an open source project. Open source projects have the advantage that their entire development histories -- who made changes, what was changed, code reviews, etc. -- are publicly available.

We're going to read in, clean up, and visualize the real world project repository of Scala that spans data from a version control system (Git) as well as a project hosting site (GitHub). We will find out who has had the most influence on its development and who are the experts.

The dataset we will use, which has been previously mined and extracted from GitHub, is comprised of three files:

1. pulls\_2011-2013.csv contains the basic information about the pull requests, and spans from the end of 2011 up to (but not including) 2014.
2. pulls\_2014-2018.csv contains identical information, and spans from 2014 up to 2018.
3. pull\_files.csv contains the files that were modified by each pull request.

Import the dataset into the notebook. All the relevant files can be found in the datasets subfolder.

* Import the pandas module.
* Load in 'datasets/pulls\_2011-2013.csv' and 'datasets/pulls\_2014-2018.csv'as pandas DataFrames and assign them to pulls\_one and pulls\_two respectively.
* Similarly, load in 'datasets/pull\_files.csv' and assign it to pull\_files.



## 2. Preparing and cleaning the data

First, we will need to combine the data from the two separate pull DataFrames.

Next, the raw data extracted from GitHub contains dates in the ISO8601 format. However, pandas imports them as regular strings. To make our analysis easier, we need to convert the strings into Python's DateTime objects. DateTime objects have the important property that they can be compared and sorted.

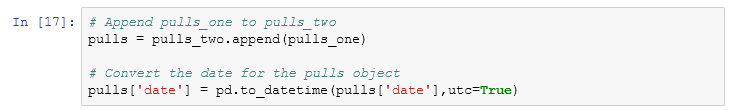
The pull request times are all in UTC (also known as Coordinated Universal Time). The commit times, however, are in the local time of the author with time zone information (number of hours difference from UTC). To make comparisons easy, we should convert all times to UTC.

Combine the two pulls DataFrames and then convert date to a DateTime object.

* Append pulls\_one to pulls\_two and assign the result to pulls.
* Convert the date column for the pulls object from a string into a DateTime object.

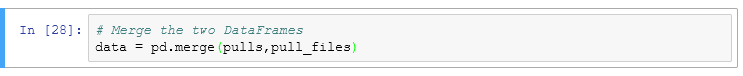
For the conversion, we recommend using pandas' to\_datetime() function. Set the utc parameter to True, as this will simplify future operations.

Coordinated Universal Time (UTC) is the basis for civil time today. This 24-hour time standard is kept using highly precise atomic clocks combined with the Earth's rotation.



## 3. Merging the DataFrames

The data extracted comes in two separate files. Merging the two DataFrames will make it easier for us to analyze the data in the future tasks.



## 4. Is the project still actively maintained?

The activity in an open source project is not very consistent. Some projects might be active for many years after the initial release, while others can slowly taper out into oblivion. Before committing to contributing to a project, it is important to understand the state of the project. Is development going steadily, or is there a drop? Has the project been abandoned altogether?

The data used in this project was collected in January of 2018. We are interested in the evolution of the number of contributions up to that date.

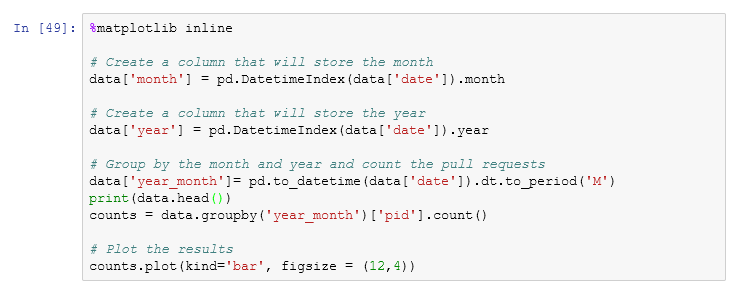
For Scala, we will do this by plotting a chart of the project's activity. We will calculate the number of pull requests submitted each (calendar) month during the project's lifetime. We will then plot these numbers to see the trend of contributions.

* A helpful reminder of how to access various components of a date can be found in [this exercise of Data Manipulation with pandas](https://campus.datacamp.com/courses/data-manipulation-with-pandas/slicing-and-indexing?ex=12)
* Additionally, recall that you can group by multiple variables by passing a list to groupby(). This video from [Data Manipulation with pandas](https://campus.datacamp.com/courses/data-manipulation-with-pandas/aggregating-data-ad6d4643-0e95-470c-8299-f69cc4c83de8?ex=9) should help!

Calculate and plot project activity in terms of pull requests.

* Group data by month and year (i.e. '2011-01', '2011-02', etc), and count the number pull requests (pid). Store the counts in a variable called counts.
  + There are a number of ways to accomplish this.
  + One way would be to create two new columns containing the year and month attributes of the date column, and then group by these two variables.
* Plot counts using a bar chart (this has been done for you).

*Note, the scaffolding exists to help you create the two columns as suggested above. However, this exercise will only check whether you create counts correctly. Thus, alternate solutions are more than welcome!*



## 5. Is there camaraderie in the project?

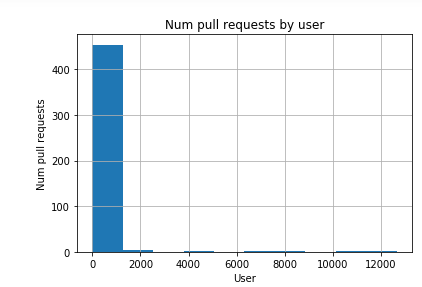
The organizational structure varies from one project to another, and it can influence your success as a contributor. A project that has a very small community might not be the best one to start working on. The small community might indicate a high barrier of entry. This can be caused by several factors, including a community that is reluctant to accept pull requests from "outsiders," that the code base is hard to work with, etc. However, a large community can serve as an indicator that the project is regularly accepting pull requests from new contributors. Such a project would be a good place to start.

In order to evaluate the dynamics of the community, we will plot a histogram of the number of pull requests submitted by each user. A distribution that shows that there are few people that only contribute a small number of pull requests can be used as in indicator that the project is not welcoming of new contributors.

Plot pull requests by user.

* Group the pull requests by each user and count the number of pull requests they submitted. Store the counts in a variable called by\_user.
* Plot the histogram for by\_user.





## 6. What files were changed in the last ten pull requests?

Choosing the right place to make a contribution is as important as choosing the project to contribute to. Some parts of the code might be stable, some might be dead. Contributing there might not have the most impact. Therefore it is important to understand the parts of the system that have been recently changed. This allows us to pinpoint the "hot" areas of the code where most of the activity is happening. Focusing on those parts might not the most effective use of our times.

Identify the files changed in the last ten pull requests.

* Select the last ten pull requests and name the resulting DataFrame last\_10.
* Merge last\_10 with the pull\_files DataFrame on pid, assigning the result to joined\_pr.
* Identify the unique files in joined\_pr (via the file column) using set().

Python's DateTime objects are comparable and sortable. A more recent date is larger than an older date. In task 2, we converted the date column into DateTime objects. Therefore, the largest ten values in the date column are the most recent ones.

pandas' nlargest method ([documentation](https://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.nlargest.html)) is helpful for the first bullet.

[Here](https://stackoverflow.com/questions/39551566/create-a-set-from-a-series-in-pandas) is an example of using set() on Stack Overflow



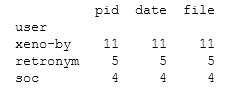
## 7. Who made the most pull requests to a given file?

When contributing to a project, we might need some guidance. We might find ourselves needing some information regarding the codebase. It is important direct any questions to the right person. Contributors to open source projects generally have other day jobs, so their time is limited. It is important to address our questions to the right people. One way to identify the right target for our inquiries is by using their contribution history.

We identified src/compiler/scala/reflect/reify/phases/Calculate.scala as being recently changed. We are interested in the top 3 developers who changed that file. Those developers are the ones most likely to have the best understanding of the code.

Identify the top 3 developers that submitted pull requests to src/compiler/scala/reflect/reify/phases/Calculate.scala.

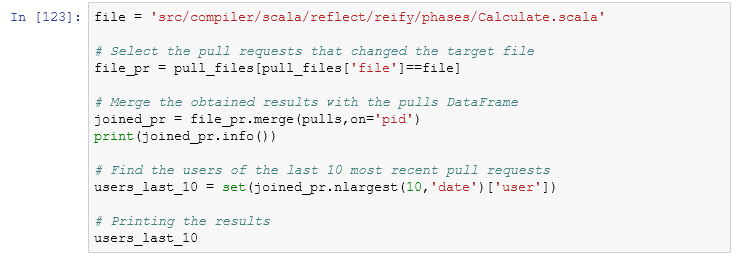
* Select the pull requests that changed that file and name the resulting DataFrame file\_pr.
* Count the number of changes made by each developer and name the resulting DataFrame author\_counts.
* Print the top 3 developers.



## 8. Who made the last ten pull requests on a given file?

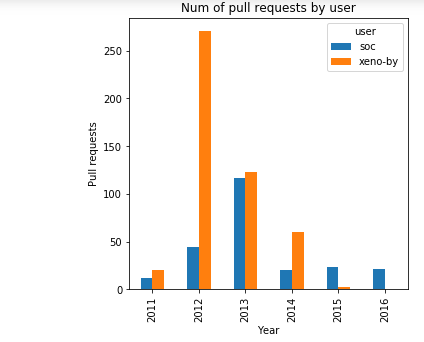
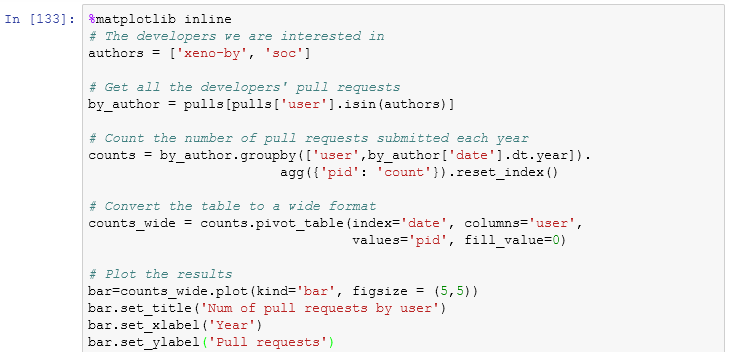
Open source projects suffer from fluctuating membership. This makes the problem of finding the right person more challenging: the person has to be knowledgeable and still be involved in the project. A person that contributed a lot in the past might no longer be available (or willing) to help. To get a better understanding, we need to investigate the more recent history of that particular part of the system.

Like in the previous task, we will look at the history of src/compiler/scala/reflect/reify/phases/Calculate.scala.



## 9. The pull requests of two special developers

Now that we have identified two potential contacts in the projects, we need to find the person who was most involved in the project in recent times. That person is most likely to answer our questions. For each calendar year, we are interested in understanding the number of pull requests the authors submitted. This will give us a high-level image of their contribution trend to the project.



## 10. Visualizing the contributions of each developer

As mentioned before, it is important to make a distinction between the global expertise and contribution levels and the contribution levels at a more granular level (file, submodule, etc.) In our case, we want to see which of our two developers of interest have the most experience with the code in a given file. We will measure experience by the number of pull requests submitted that affect that file and how recent those pull requests were submitted.

