EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP
STOCKHOLM, SVERIGE 2020

The Most Popular Programming Languages of GitHub's Trending Repositories

KRISTOFFER GUNNARSSON

OLIVIA HERBER

KTH
SKOLAN FÖR ELEKTROTEKNIK OCH DATAVETENSKAP

The Most Popular Programming Languages of GitHub's Trending Repositories

KRISTOFFER GUNNARSSON
OLIVIA HERBER

Bachelor of Science in Engineering - Computer Science and Technology
Date: June 8, 2020
Supervisor: Richard Glassey
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science (EECS)
Swedish title: GitHubs populäraste projekt och deras programmeringsspråk

Abstract

GitHub is one of the most popular hosting sites for software development, version control and code collaboration, often being used in open source development. The website has a trending page showing the current most popular projects, where popularity mostly is determined by the amount of users that have starred a given repository. This thesis aims to investigate the trends of the most used programming languages of these trending repositories, during a five year period from 2015 to 2020. This is done by scraping a daily newsletter containing the trending repositories, analyzing the data and comparing our results to other research on popular programming languages.

Our results somewhat correlate with the results of other studies on popular programming languages. Languages such as Java, Python, JavaScript and C++ are represented in the top of both our results and the results of other studies. JavaScript is by far the most popular programming language in our results. The languages that made it to the top of our results, but are not represented in other studies, are mostly languages related to web development, such as HTML, CSS, and TypeScript. This might suggest that web development projects are more likely to become popular on GitHub.

An interesting outcome of our results is that almost a fifth of the trending repositories did not contain any code. A random sample of these repositories was obtained and manually examined. This showed that these repositories are mostly educational computer science resources, or other resources related to computer science and technology, which corresponds with GitHub's user base of software developers.

Sammanfattning

GitHub är ett populärt verktyg för mjukvaruutveckling och används speciellt för projekt med öppen källkod. Hemsidan har en Trending-sida som visar de mest populära projekten för stunden, där populäriteten mäts genom hur många användare som stjärnmärkt det. Denna kandidatuppsats ämnar att undersöka vilka programmeringsspråk som är populärast bland projekten som hanmar på Trending-sidan på GitHub. För att göra detta analysers data från en femårsperiod mellan 2015 och 2020 som hämtats från ett nyhetsbrev som listar dessa projekt. Resultaten jämförs sedan med andra studier som har kollat på populäritet av programmingsspråk.

Våra resultat korrelerar till en viss del med andra rankningar av populära programmeringgspråk. Stora språk som t.ex. Java, Python, JavaScript och C++ toppar både våra resultat och resultaten av andra studier. JavaScript var det absolut mest representerade programmeringsspråket i vårt dataset. Detta språk, och andra språk som används i webbutveckling, var överrepresenterade i våra resultat jämfört med andra studier. Detta kan betyda att projekt inom webbutveckling har en större chans att bli populära på GitHub.

En intressant aspekt av våra resultat är att ungefär en femtedel av projekten inte innehöll någon kod alls. En manuell granskning av en delmängd av dessa projekt utfördes, vilket visade att nästan alla ändå är relaterade till programmering och teknologi. Urvalet innehöll bland bland annat information och genomgångar av programmeringsspråk och karriärstips för personer som vill arbeta inom utveckling.

Contents

Chapter 1

Introduction

Trends affect the way we make decisions, both positively and negatively. In the world of programming, one trend that can be analyzed involve programming languages. It is important to understand the reason(s) for the changes both when designing new languages and when choosing which to use for a project. A question such as: "why do people seem to choose language X, when Y has been the go-to for several years?" is important to be able to answer in order to motivate the usage of new languages. If a new language is to be designed it is essential to see which languages are currently in use and what features they contain. There is no point in designing a language similar to one that has been disfavored for years.

One example of a recent language that has been gaining popularity over the recent years is Rust. It was created in 2010 and has for the past five years been ranked as the most beloved language in Stack Overflow's yearly developer survey [1]. One potential reason for its popularity is that the developers saw the need for a low-level language with less expressive memory management than C or C++. Thus Rust was born and is currently climbing the ranks in terms of popularity.

There are a few common approaches to measure popularity which will be further explained in the Background chapter. The approach this thesis will focus on is gathering data from open-source projects. In order to get access to such projects a source-code hosting platform has to be chosen. To get a representative view of what languages can be considered popular, some criteria must be fulfilled.

First of all the platform has to have a big enough user-base. Looking at the most popular platforms, GitHub reports around 40 million users[2], BitBucket around 10 million[3] and SourceForge a few millions[]. GitHub has a clear advantage in this aspect with almost 3 times as many users than BitBucket and SourceForge combined. Finally it is important to have easily accessible data. Briefly reading the API documentation for each of these platforms gives the impression that GitHub has the most extensive one in terms of statistical data. Our final choice of platform is therefore GitHub.

Finding a suitable dataset on GitHub's platform can seem like a daunting task since 16 repositories are created every ten seconds on average [4]. A solution was however quite apparent once we found their Trending page. This page contains the most popular public repositories (projects) over a time period of a day, week or month. Although, what created a problem was the fact that there is no functionality to go back to a certain date and see that day's trending repositories. This requirement is a must since we need data over a certain time-span to observe changes in trends.

Through research we found a newsletter called Changelog Nightly which sends out daily updates on GitHub's trending page. We then found out that Changelog archive their newsletters, thus enabling us to build a scraper in order to extract the necessary data. This led us to form the following aim of our thesis:

1.1 Research Question

This thesis aims to find the most prevalent programming languages in GitHub's trending page. Furthermore, we aim to compare our findings to other measurements of programming language popularity.

1.2 Scope

Because of the limited amount of data available in regards to Changelog Nightly's newsletters this study will only consider the past five years. That is, 2015 to Q1 2020.

Chapter 2

Background

The purpose of this chapter is to give the reader some insight into some relevant studies as well as a description of GitHub and Changelog Nightly. The first section describes GitHub's own study and two web search based indexes. The second section goes into detail about GitHub, what is to be expected is the explanation of repositories, the trending page and GitHub's star system. The third and final section explains the contents of the Changelog Nightly newsletter.

2.1 Previous Studies

There have been some previous surveys of the most popular programming languages during the last few years. They have all used different methodologies and resulted in different conclusions. This section will go more into depth about a few that seem relevant to this thesis.

The most common approaches when looking at programming language trends seems to be to either analyze projects on open-source platforms such as GitHub [5] or SourceForge [6], or analyzing web-search results by looking for a target search phrase [7][8]. The first approach is often found in academic papers, one example of this is a study conducted by T. F. Bissyandé et al. where 100 000 GitHub repositories were analyzed [5]. Another example is a study by Leo A. Meyerovich and Ariel S. Rabkin, analyzing 200 000 SourceForge projects and multiple programmer surveys [6]. The second approach of analyzing web-search results seems to be more common when constructing indexes, such as the PYPL index [7] and the TIOBE index [8]. This is quite logical since it is easier to automate the process of scraping search results than analyzing huge project datasets.

The motivation for conducting such studies according to T. F. Bissyandé et al. as well as Leo A. Meyerovich and Ariel S. Rabkin, is that a study of such a big dataset had not been performed at the time of writing, therefore they aimed to fill that gap. Another motivation is that studying programming language trends, is of importance to understand the underlying factors in whether a language is successful or not.

2.1.1 GitHub - The State of the Octoverse

GitHub conducts a yearly analysis of the most popular programming languages. They look at the primary programming language of all private and public repositories and rank the languages by the number of unique collaborators [2]. This was done for the first time in 2014, which gives us six years of data. The results can be seen in figure 2.1.

In this study JavaScript came out on top and has consistently been the most popular programming language each year since 2014. Python was the second most popular language until 2019, when it was overtaken by Java. Java started at fourth place in 2014 and rose to third place the following year where it stayed until rising to second place in 2019.

2.1.2 IEEE Spectrum

Most notable are the annual rankings by IEEE Spectrum that started in 2014 [9]. The ranks are calculated using a combination of 8 sources, such as Google, Stack Overflow, Twitter, GitHub and CarreerBuilder (a website for job openings). They, however, almost only count mentions of the language and not which are actually being used in projects.

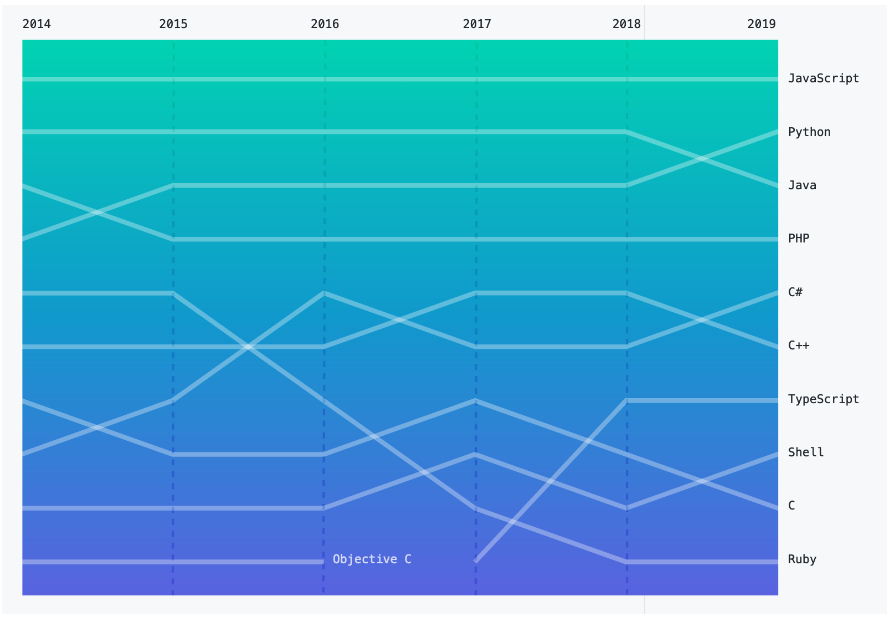


Figure 2.1: The most used programming languages in GitHub repositories ranked by number of unique collaborators. Source: GitHub The State of the Octoverse 2019 [2]

2.1.3 The PYPL PopularitY of Programming Language Index

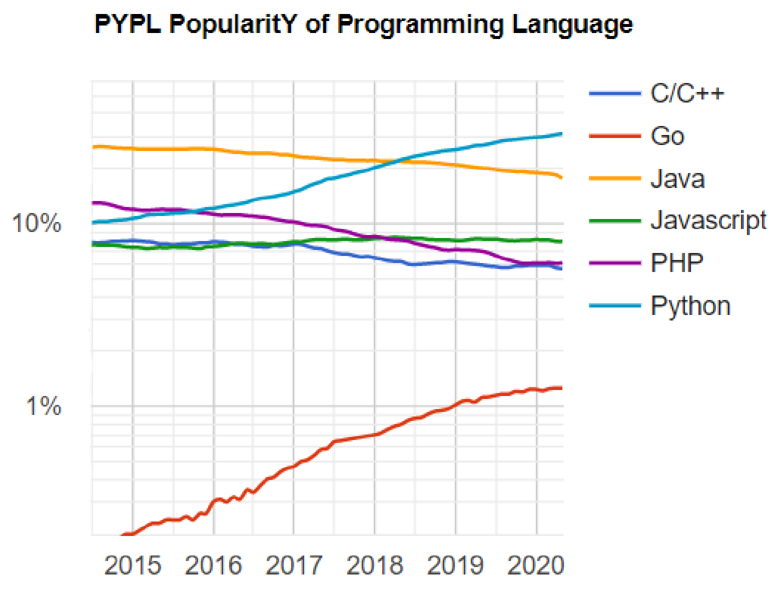


Figure 2.2: Semi-logarithmic graph representing the popularity of programming languages according to the PYPL index. For reference the peak of Python is at 31.2%.

"The PYPL PopularitY of Programming Language Index" [7] is an index using Google searches in order to measure the popularity of a programming language. The way it is done is by using raw data from Google Trends, and looking at how many times a programming language tutorial is searched for. If a language tutorial is searched for more, the language is considered more popular.

In figure 2.2 a logarithmic graph can be seen representing the data collected by PYPL from 2015 to 2020. The percentage on the Y-axis represents the share of searches each language has accumulated. The languages represented on the graph are some of the more popular ones and others have been filtered in order to avoid clutteredness. The graph still provides enough information to get an insight in the index's measurements.

2.1.4 TIOBE Programming Community Index

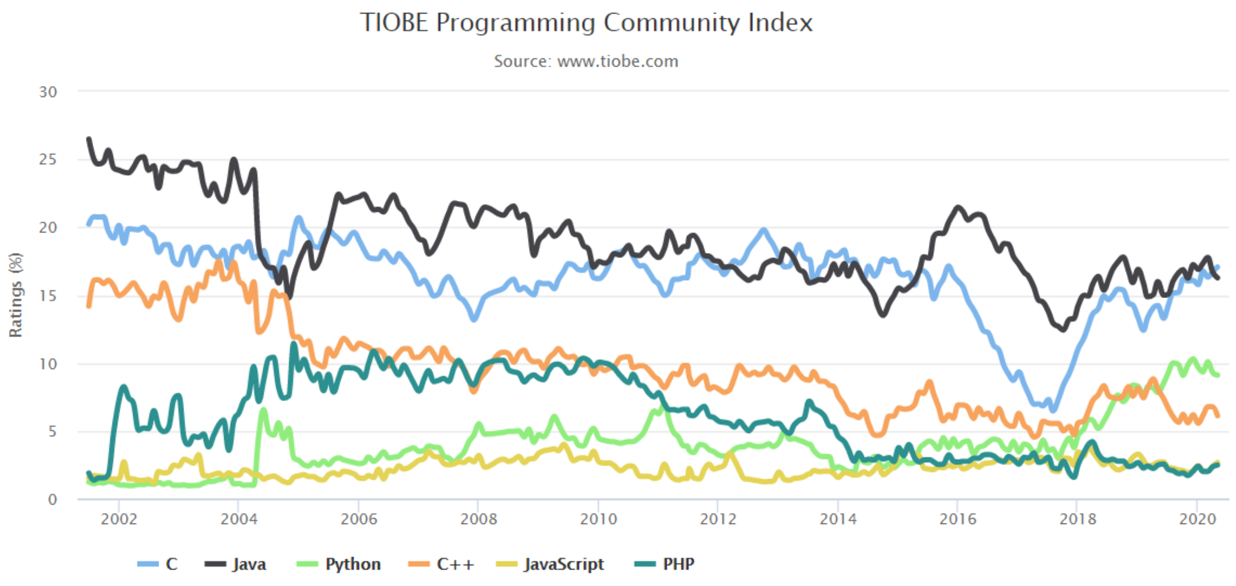


Figure 2.3: Graph representing the popularity of programming languages according to the TIOBE Programming Community Index

The TIOBE Programming Community Index [8] is another index which can be useful to compare the results against. Instead of looking for tutorial searches, this index counts the hits of the search query "[language] programming". In comparison to the PYPL index which only uses Google searches, TIOBE gather their results from additional search engines as well. Some of those are (in order of number of search results): Google.com, Baidu.com, Yahoo.com and Wikipedia.com. What is interesting to note is the second result Baidu.com, which is the most popular search engine in China. By including various search engines the final measurement from the TIOBE Index can be seen as more inclusive.

2.2 GitHub

GitHub is an online platform for code collaboration and hosting. Users create containers called “repositories” (or “repos” for short) that can be shared with other users. These repositories are either private (only visible to the owner and users it has been shared with) or public (accessible to everybody online). Users

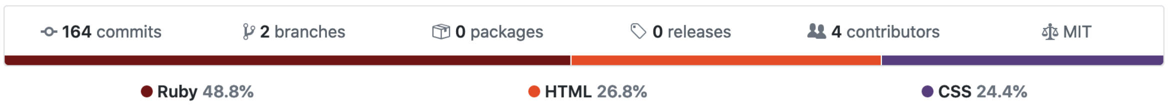


Figure 2.4: Some of the metadata for the public repository Changelog Nightly [12] (accessed April 21st 2020). The programming languages used in the project and their respective ratios can be seen at the bottom. In this case the primary programming language is tagged as "Ruby", since most of the code is written in this language.

can also create separate copies of the code (branches) to independently work on specific features of the project and then merge them into the main (master) branch when they are ready for deployment. All of this makes GitHub a great tool for software collaboration and open-source projects.

GitHub collects various different metadata for each repository. One of these is the "primary programming language". Since a project can utilise multiple different programming languages GitHub chooses the one the represents the highest percentage of code in the repository as the primary programming language. GitHub uses the Linguist Library to determine the programming languages that are used [10]. The library looks at different aspects, including file-extensions, to determine the used programming languages and updates the ratios each time new code is added to the master branch [11]. If the repository does not contain any code, it is not tagged with a primary programming language.

2.2.1 Trending Repositories and Stars on GitHub

GitHub users can "star" repositories that they are interested in. This acts as a bookmark and saves the repository to a special page where the user easily can easily access their starred repositories. GitHub lists the total number of stars each repository has as part of its metadata, the amount of stars can therefore be used as a popularity metric for public repositories.

GitHub has a “Trending” page that shows the current 25 most popular public repositories on the website. The page can show the most popular repositories of the last day, last week or last month as well as apply filters for the primary programming language and spoken language. GitHub has never released an official explanation of how the algorithm for the trending page works, most likely in order to prevent users to cheat their way onto the list. The user has the option to see the trending repositories for the current day, the past week or the past month. GitHub does not publish any archive of the repositories that

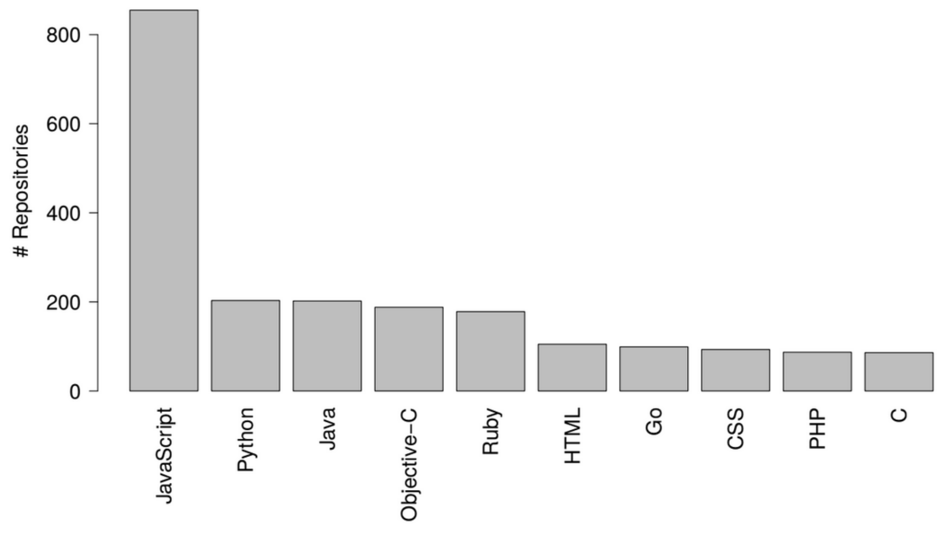


Figure 2.5: Programming language by amount of repositories from GitHub's 2500 most starred repositories. From the study by Hudson Borges et al. [13]

have been on the trending page reaching further back than that.

2.2.2 Previous Studies on Popular Repositories

The programming languages of popular GitHub repos has been investigated before in the study "Understanding the Factors that Impact the Popularity of GitHub Repositories" [13]. They gathered the top 2 500 repos with the most stars in 2016 and investigated factors that could have influenced the repo's popularity, including primary programming language. Their results on the most popular programming languages can be seen in figure 2.5

A study by Hu et al. looked at the most popular repositories on GitHub using the amount of stars a repository received [14]. They found that most of the popular repositories were web-applications written in either JavaScript or HTML. The repositories written in JavaScript were overwhelmingly more popular than repositories written in any other languages.

2.3 Changelog Nightly

Changelog Nightly is a daily newsletter that is sent out to subscribers at 10 PM U.S. Central Time every day [15]. The newsletter lists the public GitHub repositories that have gotten the highest numbers of new stars during the last day. Since all past newsletters are available as web-pages online, a big archive of popular repositories can be accessed.

The newsletter lists repositories in three different categories. It lists the repositories that overall have gotten the most new stars during the day in two categories, "First timers" and "Repeat Performers", depending on whether the repository has made an appearance on the list before. A third category, "Top New Repositories", lists the most starred repositories of all repositories that were made public on the day of the newsletter. This third category gives an insight into new repositories that might be trending later. The amount of repositories listed in the different categories, and the newsletter, differ from day to day.

The newsletter includes data about the repository's total amount of stars, new stars received that day and primary programming language. All repositories on the "Repeat Performers" also include information on the amount of times it has been listed before. The name and a short description of the repository is also included. Out of this data, the area of focus will be the programming language in order to see which programming languages are the most popular, and how these trends have changed over the years.

Chapter 3

Method

The research question is addressed by collecting five years of historical data on GitHub's most starred repositories in order to get a somewhat broad perspective. A too small dataset would result in less convincing results since long-term trends are sought after. Even though the trending page has existed for longer than five years, only archives starting at January 1st, 2015 have been found.

3.1 Scraping Data from Changelog Nightly

The trending data is gathered from Changelog Nightly [15]. All previous newsletters can be accessed on the site by modifying the URL. The archives go back to January 1st 2015, which results in more than five years of daily data. Changelog Nightly hosts their code on GitHub under the MIT licence, meaning that the code can be used and modified freely as long as the copyright and license notices are preserved.

A Python script using Beautiful Soup was used to access and save all past newsletters and scrape the relevant information from the source code. This included the name and creator of the repository, information about its amount of stars. The information was then saved into CSV-files in order to be processed later. The code that was used can be seen in Appendix A.

3.2 Getting Additional Data with GitHub's API

In order to get more detailed information about the repository, GitHub's API will be used. The difference between the data acquired through the API and the data scraped from Changelog Nightly is that the API data will be current data about the repository. The API data will also be more detailed in regards to what can be extracted. In order to make comparisons with the Changelog data, information such as the current number of stars and primary programming language will be extracted. In addition to this the number of contributors, commits, all used languages and commit activity will also be collected. The purpose for collecting additional data is to see if any conclusions can be drawn in regards to the current state of the repository versus its state when showing up on the Changelog Nightly list. The code used for acquiring the API data can be found in the Appendix section: API Fetching.

3.3 Analyzing Data

Python's data-analysis tools were used in order to analyze the data from the CSV-files that were created when scraping the newsletters. These tools include Pandas, Matplotlib and the Jupyter Notebook environment. The final version of the used Jupyter Notebook can be found in appendix A.

In most of the analysis all repositories from the category "New Repositories" were filtered out. This is since the category only lists new repositories that were made public on the same day as the newsletter is released. The repositories in this category receive far fewer stars than the repositories in the other categories and therefore cannot be considered popular.

Many repositories quickly turned out to not be tagged with a primary programming language, meaning that they contain things other than code. In order to explore these repositories further, a sample of 50 unique repos was randomly selected and manually examined. Each repository in the sample was visited and categorized according to its current contents.

Chapter 4

Results

From the scraped newsletters we get daily data from January 1st, 2015 to April 26th, 2020 (the day that the data was scraped). This gives us 1 940 days of data. We have a total 43 158 unique entries from all newsletters, but only 24 306 entries remain after the repositories in the "New Repository" category have been filtered out. The reason for this filtering is stated in the Analyzing Data section of the Methods chapter. All results are based on these filtered repositories. The number of entries differs each year, however, and declines with time. Since a repository can appear on the list multiple weeks, the number of unique repositories is lower than the number of entries each year. Table 4.1 shows the number of entries and unique repositories obtained from each year.

Year	Total Entries	Unique Repositories
2015	5465	1996
2016	5174	1976
2017	4501	1548
2018	4069	1237
2019	3863	1143
2020 [1]	1234	462
Total	24306	7647

Table 4.1: Total amount of repositories in the dataset

[1]From January 1st, 2020 up until (including) April 26th, 2020

4.1 Primary Programming Languages

From all repositories in the dataset 97 unique programming languages can be found. The amount of unique programming languages each year ranges between 43 and 60 languages. Even though the number of unique repositories and total entries decreases each year, the number of unique programming languages do not follow this trend (excluding the current year, from which we only have about four months of data).

Year	Unique Programming Languages
2015	54
2016	60
2017	50
2018	43
2019	53
2020 [2]	36
Total	97

Table 4.2: The amount of unique primary programming languages by year

The most popular primary programming languages are clearly JavaScript, Python, Java and Go. A large amount of the repositories (17.88%) do not contain any code files at all. The repository might instead only contain text-files, images or PDF-files or other file-types. More on these repositories can be found in the next section.

JavaScript is by far the most common programming language and represents almost a third of all trending repositories from 2015 to 2018. The popularity however dips in 2019 and the language is only the primary programming language of around 17% of the repositories in 2019 and the first part of 2020. Even though the language dips in popularity it remains the most popular language throughout the whole time measured.

Another programming language that has dipped in popularity is Java. In 2015 it was the second most represented language (after JavaScript) but got overtaken by Python in 2016 and fell down to fifth place by 2018. The 10 most popular programming languages, and the amount of repositories that do not contain any code, can be found in figure 4.1.

[2] From January 1st, 2020 up until (including) April 26th, 2020

Programming language	Count	Proportion of all repositories
JavaScript	6672	27.45%
(No code in repository)	4345	17.88%
Python	2507	10.31%
Java	1667	6.86%
Go	1618	6.66%
C++	1025	4.22%
C	795	3.27%
HTML	718	2.95%
Swift	669	2.75%
TypeScript	548	2.25%
CSS	532	2.19%

Table 4.3: Most frequently occurring languages

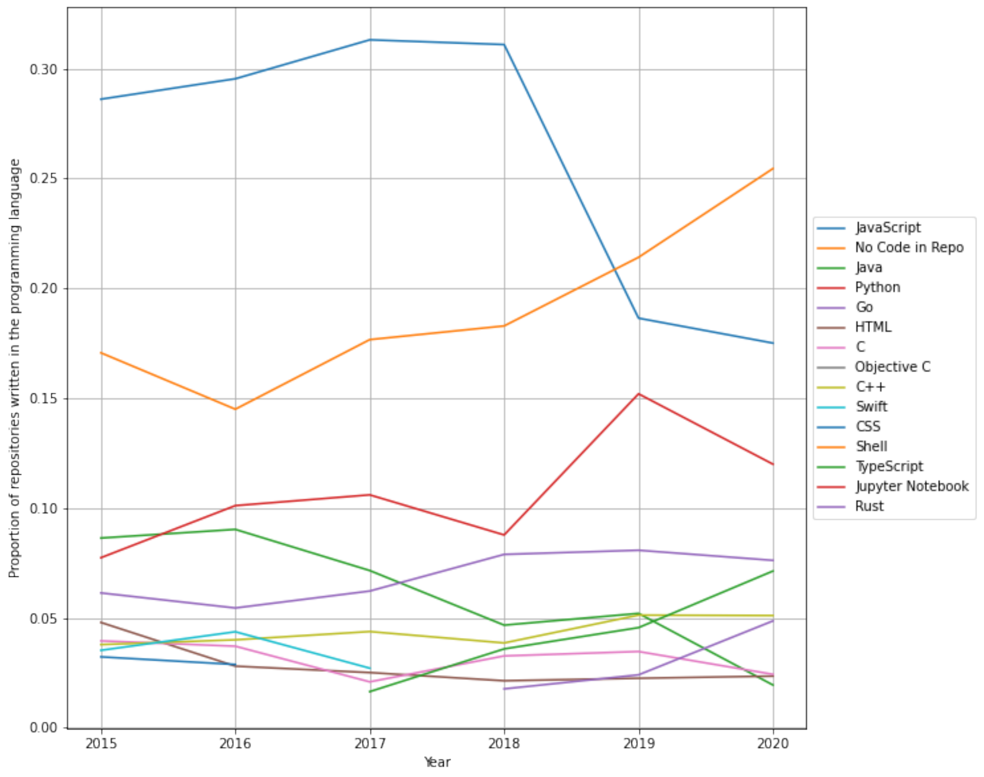


Figure 4.1: A graph of the 11 most popular programming languages (including repositories that do not include any code) the from our dataset. Since the dataset has a differing number of repositories from each year, the data points have been normalized by dividing the number of repositories tagged with each programming language by the total number of repositories from that year.

4.2 Repositories Not Containing Code

As stated above, a surprising amount of repositories in the dataset do not contain any code. This does not mean that the repositories are empty, instead they might contain text-files, PDFs or images. A total of 4 345 data entries, or about 18% of all entries, were repositories in this category. After manual examination of a sample of 50 randomly selected repositories from these entries, they could be divided into the categories seen in table 4.4.

Category	Number of Repos
CS resources [links]	17
Educational CS resource	16
Tech-career related	5
Other GitHub repositories [links]	3
Dataset	2
Documentation	1
Other tech-related	4
Non tech-related	2
Total	50

Table 4.4: Categorization of a random sample of 50 repositories that did not contain code.

The majority of the repositories in the sample contained educational resources for different topics in computer science. The category *CS resources [links]* in table 4.4 is repos with collections of free books, tutorials and links to external resources. These resources have not necessarily been written by the repository contributors themselves, but are available online and have been compiled into lists by the repo collaborators. Some examples of the contents of the repositories in this category are links to online Java-tutorials, a collection of research-papers in deep learning and links to useful YouTube-videos concerning web-development.

In contrast to the category above, the category *Educational CS resource* includes original resources for computer science learning that are hosted directly on GitHub. These seem to be written by the contributor(s) themselves. Some examples are detailed tutorials for a specific programming language, books on e.g. machine learning or write-ups of tips on computer science related subjects.

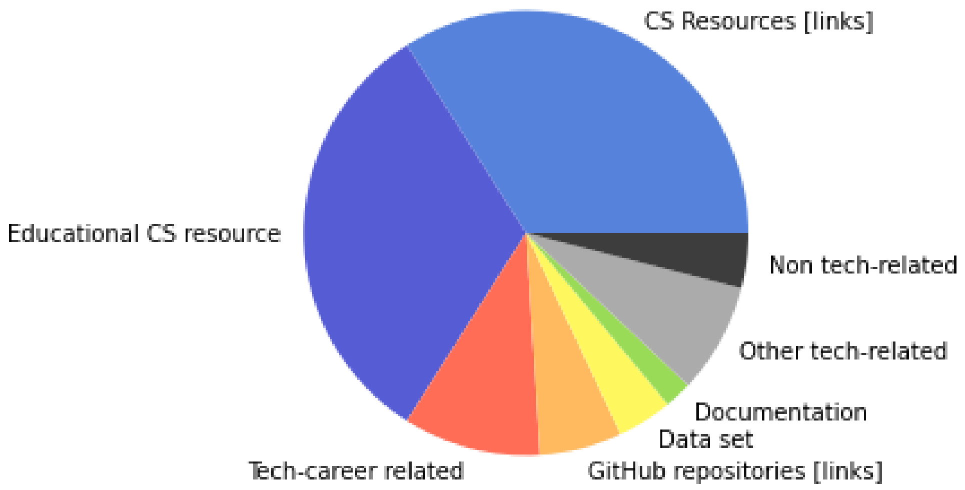


Figure 4.2: A graph showing the proportion of each category in the random sample of 50 repositories that do not contain any code.

These two categories span almost half of the sample, with 17 repositories of the sample being collections of external computer science resources and 16 repositories being original resources hosted on GitHub. The rest of the repositories in the sample also contain computer science or tech related information, which the exception of two repositories. These are one repo with tips on English learning for Chinese speakers and one repo with links to resources on the stock market.

Some categories did not contain many repositories, but might still be considered concrete categories. Five repositories has content related to careers in tech-industries, such as examples of interview questions. Three repositories in the sample were collections of links leading to other repositories on GitHub, collecting projects that fall under specific categories. The sample also included two dataset and one repository that hosted the documentation for a programming language.

Four repositories in the sample were unique in their contents and do not fit into a broader category, but were still related to technology or computer science. These were a list of names of awesome programmers, information on one user's Mac setup, a handbook for employees of a software company and one repo with personal blog-like text about technology tips the creator learnt each day.

4.3 API Data

Data regarding each of the 24 306 repositories were collected from GitHub's API. 1 508 of them were inaccessible either because of being made private instead of public, removal, or been emptied out. No analysis was made due to time constraints. Data is however open for access per request. The following figure shows exactly what a data sample contains and how it is formatted.

```
{
  "name" : "pingcap/chaos-mesh",
  "commits": 500,
  "contributors": 5,
  "stars": 15000,
  "primary_language": "Go",
  "languages": {
    "Go": 3505876,
    "Java": 239289
  },
  "commit_activity": {
    "2019": {
      "20": [0, 0, 0, 0, 0, 1, 0],
      "26": [0, 0, 0, 2, 0, 0, 0]
    },
    "2020": {
      "16": [0, 0, 0, 1, 0, 0, 3]
    }
  }
}
```

Figure 4.3: Example of repository data for one repository.

Chapter 5

Discussion

To reiterate, our research question is to find the most prevalent programming languages in GitHub's trending page, ranging from 2015 to the first quarter of 2020. The results we have been able to generate, display both the most frequently occurring languages across this time span, as well as how the trends change over the years.

5.1 Frequently Occurring Languages

Looking at the most frequently occurring languages in table 4.3, we see that 27.45% of the scraped repos contain JavaScript. This is then followed by repositories containing no code at all at 17.88%. Finally Python ends up in third place with a share of 10.31%. These three combined make up for more than half of all repositories, which is quite an interesting result as a only few languages are used the most. What is perhaps most intriguing, is the "no code in repository" finding. As mentioned in the background chapter, GitHub is primarily used as a platform for hosting and sharing source-code. Here we however see that about a fifth of the trending repositories do not contain any code at all. This shows that GitHub is not solely used for the development of code, but also as a place for storage of information.

5.2 Repositories Without Code

The main purpose of GitHub is software collaboration and version control, which makes the high percentage of repositories in our data that do not contain code surprising. The repositories, however, seem to mostly be related to computer science and technology, which fits with the user base of software developers. Over half of the repositories in our random sample contain educational computer science resources, which most likely are starred by users in order to have it saved for future reference.

There are many reasons as to why people might use GitHub for projects in other areas than software development. First off, GitHub has great tools for collaboration, with users being able to make small additive contributions without giving others the ability to make detrimental and irreversible changes to the project. GitHub is also already an industry standard with a large user base. This means that many people are familiar with its interface and function, as well as giving the project the chance to be discovered by people on the website. Lastly, many software developers include their GitHub profile as a part of their application when looking for a job. Working on educational resources shows a clear understanding of the area. Furthermore, hosting their work on GitHub ensures that the recruiter sees a more multifaceted picture of the applicant in one place, without having to visit multiple websites.

There is a clear trend towards compilations of lists. This could be explained by the ease of collaboration on GitHub, as many people can add their favorite resources to the list. GitHub user's affinity towards lists is made clear by one repository in our sample, lists lists by the user "jnv", where almost one hundred users have contributed to a long list of other GitHub repositories that also contain lists. These repositories might be starred by users for two reasons; both in order to save them as reference, as well as being able to make additions to the lists when the user finds new appropriate resources.

Unfortunately, we do not have information on whether this type of repository without code is over or under represented in our dataset, compared to all repositories in GitHub. One study looked at a sample of 434 random open GitHub repositories from the whole site and found that 8.3% of these were used for storage [16]. Their classification of repositories is however too different from ours to draw conclusions from the results.

5.3 Comparison With Other Indexes of Popular Programming Languages

5.3.1 PYPL Index

If we recall, PYPL's results come from looking at the number of Google searches of the term "[language] + tutorial", which gives a hint of which languages are in demand to learn. We can immediately see a difference between their results and ours. According to PYPL, Java started off the strongest in 2015, while JavaScript is the most popular programming language in our results. However, in around mid 2018 it was overtaken by Python which by the start of 2020 had a share of around 30%. Three languages that appear to be pretty stable in their popularity are JavaScript, C/C++, and PHP, at around ten percent each. What can be noted however is that both PHP and C/C++ are declining in popularity according to PYPL.

5.3.2 TIOBE Index

To get a broader perspective we can take a look at the TIOBE index as well (figure 2.3). Focusing on the relevant year-span of ca. 2015 to 2020, we see a bit of a different picture than the one PYPL paints. What is important to note is the way TIOBE generates their results. Instead of looking at the number of searches for a term, they look at how many hits various search engines get from the term "[language] + programming". This means that while PYPL looks at how many people are searching for language tutorials, TIOBE looks at how many websites contain the search query. What their results tell us is that Java had a local peak at about 22% around year 2016 but then dove down to ca. 13% in 2018. While its popularity slowly went up after that, it never fully recovered. What is also interesting to note, is that C follows Java's trend quite closely in terms of when they decline and grow. C++ and Python seem to stay quite comfortable at around the five percent mark, up until 2018 where they both increase in popularity. In 2019 we do however see a sharp decline in C++'s popularity while Python increases at about the same amount as C++ declines. Reaching 2020, the top rankings according to TIOBE are C, Java, Python and C++.

5.3.3 GitHub - The State of the Octoverse

Another interesting source of comparison is GitHub's own study: The State of the Octoverse (figure 2.1). As previously mentioned, this is a study conducted every year by GitHub where they present statistics related to their platform. The relevant part to our study is where they rank the most used programming languages, ranging from 2014 to 2019. In difference to our measurement, they rank languages based on the amount of unique collaborators in both public and private repositories. Comparing this information to our results can be useful since we only have access to public repos. What can be seen from their results is that JavaScript has been the leading language throughout the whole period, which matches with ours. Following JavaScript is Python, Java, and PHP. The three of them have mostly been stable in that order except for a few shifts in rank between them.

5.3.4 Overall Remarks

When comparing these indexes to our study, we see that PHP does not appear anywhere in our top results (table 4.3), even though it is quite popular in other indexes. One of the reasons for this could be that PHP is an older language for web-development, where JavaScript is more often used in its place for new projects. Since it used to be a popular language, there is probably much code written in the language that still is maintained. Our results show projects that are currently popular and most likely quite new, while the other indexes include programming languages that also are used in older projects.

Programming languages used in web-applications, such as JavaScript, HTML, CSS and TypeScript, seem over-represented in our results when comparing to the other general indexes. This, however, matches with the results from the two other studies discussed in Previous Studies on Popular Repositories This might suggest that web-applications either have a higher chance of trending on GitHub, or that a big proportion of all projects on GitHub are web-applications.

5.4 Research Limitations

5.4.1 Primary Programming Language

Since our data relies on the primary programming language of each repository, and not all languages used in the project, the results might look different if all programming languages used were considered. GitHub also has a broad scope of what it classifies as a primary programming language. Therefore, our results include some classifications that traditionally are not considered programming languages, such as HTML, CSS and Jupyter Notebooks.

5.4.2 Unanalyzed API Data

As mentioned in the results, the data gathered from GitHub's API was not analyzed. This was mainly due to lack of time since we chose to investigate further into the "no code repositories" part of our initial results. Another problem which arose was fact that GitHub limits the number of requests an API key can make during a certain time period. Through trial and error we found out that data for around 300 - 400 repos could be downloaded per hour and API key. In order to bypass this limit, the script was modified to alternate between two API keys. This increased the download ratio to about 600 - 800 repos per hour. If we were to do this again, it would be good to investigate if there is a better way to get around the limit.

5.4.3 Future Research

Our dataset can be used to explore other questions than the ones that we looked at in this project. It would be interesting to investigate the time frame of how long repositories trend. We obtained data on the repositories that trended for the biggest total amount of days, but were not able to include or analyze it in our report due to time constraints. The list can be found as output number 16 in the Jupyter Notebook in appendix A.

As we were not able to explore the additional data we obtained through GitHub's API (see section 4.3), this data should be investigated in order to gain further insights. One question that could be answered with this data, is whether projects are still worked on after they trend or if they are already finished by the time they trend. The data also includes information on all programming languages used in the project, not just the primary programming language.

As stated before, it would be interesting to investigate the amount of repositories on GitHub, that do not contain code, and are used for similar purposes as the repositories in our dataset. This would give further insight into our results and whether the repositories without code are over or under represented in trending repositories.

Chapter 6

Conclusions

Our results somewhat correlate with the results of other studies on popular programming languages. Languages such as Java, Python, JavaScript and C++ are represented in the top of both our results and the results of other studies. The only index that contradicted this was the TIOBE index, which can be explained by their method of popularity measurement. Otherwise, it is seems like GitHub's trending page follow roughly the same trends as the compared-to studies suggest.

The languages that made it to the top of our results, but were not represented in other studies, are mostly languages related to web-development, such as HTML, CSS, and TypeScript. The outlying result in terms of web-development is that JavaScript by far is the most popular programming language in our results, which is also represented in other studies. This might suggest that web development projects using JavaScript are more likely to end up on GitHub's trending page.

Even though the intended use of GitHub is to host and collaborate on software projects, a large percentage (almost 18% over the last five years) of the most popular repositories do not contain any code at all. The percentage has been steadily increasing since 2016 and has reached around 25% of the the trending repositories so far in 2020. This shows that many users use GitHub for purposes other than software development. However, the projects in this category were still mostly related to computer science and technology.

Bibliography

[1] Stack Overflow. *2020 Developer Survey*. 2020. URL: `https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-loved`.

[2] *The State of the Octoverse*. 2019. URL: `https://octoverse.github.com/`.

[3] Kelvin Yap. *CCelebrating 10 million Bitbucket Cloud registered users*. 2020. URL: `https://bitbucket.org/blog/celebrating-10-million-bitbucket-cloud-registered-users`.

[4] Jason Warner. *Thank you for 100 million repositories*. 2018. URL: `https://github.blog/2018-11-08-100m-repos`.

[5] T. F. Bissyandé et al. "Popularity, Interoperability, and Impact of Programming Languages in 100,000 Open Source Projects". In: *2013 IEEE 37th Annual Computer Software and Applications Conference*. 2013, pp. 303–312.

[6] Leo A. Meyerovich and Ariel S. Rabkin. "Empirical Analysis of Programming Language Adoption". In: *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*. OOPSLA '13. Indianapolis, Indiana, USA: Association for Computing Machinery, 2013, pp. 1–18. ISBN: 9781450323741. DOI: `10.1145/2509136.2509515`. URL: `https://doi.org/10.1145/2509136.2509515`.

[7] Pierre Carbonnelle. *PopularitY of Programming Language index*. Mar. 2020. URL: `https://pypl.github.io/PYPL.html`.

[8] TIOBE Software BV. *TIOBE Index for March 2020*. Mar. 2020. URL: `https://www.tiobe.com/tiobe-index/`.

[9] Stephen Cass. *The Top Programming Languages 2019*. Sept. 2019. URL: https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019.

[10] GitHub. *About repository languages*. 2020. URL: https://help.github.com/en/github/creating-cloning-and-archiving-repositories/about-repository-languages.

[11] *linguist*. 2020. URL: https://github.com/github/linguist.

[12] The Changelog. *thechangelog/nightly*. 2020. URL: https://github.com/thechangelog/nightly.

[13] Hudson Borges, Andre Hora, and Marco Tulio Valente. "Understanding the Factors That Impact the Popularity of GitHub Repositories". In: Raleigh, NC. Raleigh, NC: IEEE, 2016, pp. 334–344. ISBN: 978-1-5090-3807-7. DOI: 10.1109/ICSME.2016.31.

[14] Yan Hu et al. "Influence analysis of Github repositories". In: *SpringerPlus* 5 (2016).

[15] Changelog. *Changelog Nightly*. 2020. URL: https://changelog.com/nightly.

[16] Eirini Kalliamvakou et al. "The Promises and Perils of Mining GitHub". In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. MSR 2014. Hyderabad, India: Association for Computing Machinery, 2014, pp. 92–101. ISBN: 9781450328630. DOI: 10.1145/2597073.2597074. URL: https://doi.org/10.1145/2597073.2597074.

Appendix A

Code

This appendix contains all code used for the project. The datasets that were obtained and analyzed are available by request.

A.1 Web-scraper

```python
import requests
from bs4 import BeautifulSoup
import csv

import datetime
from datetime import timedelta
import time

# Change these parameters to choose the range of data to be scraped
start_date = datetime.date(2020, 1, 1)
end_date = datetime.date(2020, 4, 26)

# Get all available information for each repo in list
def getRepoInformation(repos, date, liststatus):
    for repos in repos:

        total_stars = repos.find('span', title = 'Total Stars').text.strip()
        new_stars = repos.find('span', title= 'New Stars').text.strip()

        programming_language = repos.find('span', title= 'Language')
        if (programming_language is None):
            programming_language = '[No code in repository]'
        else:
            programming_language = programming_language.find('a').text.strip()

        owner_link = repos.find('td', width = '32', valign = 'top').find('a')['
    href']

        repo_link = repos.find('h3').find('a')['href']
        repo_name = repos.find('h3').find('a').text.strip()

        if date < datetime.date(2017,5,3):
            repo_description = repos.find('td', valign='top').findNext('td').find(
    'p').text.strip()
        else:
            repo_description = repos.find('tr', class_='about').find('p').text.
    strip()

        with open('changelog_data.csv', 'a+') as csvfile:
            csv_writer = csv.writer(csvfile)
            csv_writer.writerow([date, repo_name, repo_description, repo_link,
    owner_link, total_stars, new_stars, programming_language, liststatus])


# Scape website of one day
def scrapeNewsletter(date):
    year = date.strftime("%Y")
    month = date.strftime("%m")
    day = date.strftime("%d")
    datestring = year +'/'+ month +'/'+ day +'/'
    URL = 'http://nightly.changelog.com/' + datestring

    page = requests.get(URL)

    soup = BeautifulSoup(page.content, 'html.parser')

    # Find all repos in the three different categories
    firsts = soup.find(id='top-all-firsts')
    new = soup.find(id='top-new')
    repeats = soup.find(id='top-all-repeats')

    # Get the information for each repo in each category
    if firsts is not None:
        firsts = firsts.find_all('div', class_ = 'repository')
        getRepoInformation(firsts, date, 'First Time on List')

    if new is not None:
        new = new.find_all('div', class_ = 'repository')
        getRepoInformation(new, date, 'New Repository')

    if repeats is not None:
        repeats = repeats.find_all('div', class_ = 'repository')
        getRepoInformation(repeats, date, 'Repeat Performer')


# Create csv-file where data is saved
with open('changelog_data.csv', 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)
    csv_writer.writerow(['Date', 'Repository', 'Description', 'Repo link', 'Owner
   link', 'Total stars', 'New stars', 'Primary Programming Language', 'List Status
   '])

# Get the data from each day
totaldays = (end_date-start_date).days
for x in range(0, totaldays+1):
    date = start_date + timedelta(days = x)
    scrapeNewsletter(date)

    time.sleep(5) # Wait 5 seconds between loops as to not overwhelm site
    print(str(x) + ' out of ' + str(totaldays)) # Print status
```

A.2 API Fetching

```python
from github import Github, GithubException
from github_token import ACCESS_TOKEN, ACCESS_TOKEN_2
import plotly as px
import csv
import os
import json
import time

def download_repos(token):

  t0 = time.time()

  # First create a Github instance:
  g = Github(token)

  repo_list = []
  directory = "./data/"
  i = 0
  curr_repo = 0
  repo_limit = 500

  #read repos into memory
  for filename in os.listdir(directory):
    if filename == "all_unique_repos.csv":
      filename = directory + filename
      with open(filename, newline='', encoding="UTF-8") as csvfile:
        file_reader = csv.DictReader(csvfile)
        for row in file_reader:
          if (i < repo_limit):
            i = i + 1
            repo = row['Repository']
            repo_list.append(repo)
          else:
            break

  #remove repos from file
  tempRepos = []
  downloadedRepos = []
  j = 0
  with open(directory + "all_unique_repos.csv", 'r') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader:
      if j >= repo_limit + 1:
        tempRepos.append(row)
      else:
        downloadedRepos.append(row)
      j = j + 1

  with open(directory + "all_unique_repos.csv", 'w', newline='') as csvfile:
```

```
  writer = csv.writer(csvfile)
  writer.writerow(["Repository"])
  writer.writerows(tempRepos)

with open("downloaded_repos.txt", "a") as downloaded_repos_file:
  for x in downloadedRepos[1:]:
    downloaded_repos_file.write(x[0]+ "\n")
  downloaded_repos_file.close()

repo_dict = []
parsed_data = []

print("Starting fetch process\n")
for repo in repo_list:
  #Get repo

  try:
    repo = g.get_repo(repo)
    curr_repo = curr_repo + 1
  except GithubException as e:
    curr_repo = curr_repo + 1
    print("Repo " + str(curr_repo) + " of " + str(repo_limit) + " - " + repo)
    print("Error fetching repo: " + repo)
    print(e)
    with open("missing_repos.txt", "a") as missing_file:
      missing_file.write(repo + "\n")
      missing_file.close()
    continue

  #Get repo name
  name = repo.full_name

  #Get number of stars
  num_stars = repo.stargazers_count

  #Get number of commits
  try:
    num_commits = repo.get_commits().totalCount
  except:
    print("Repo " + str(curr_repo) + " of " + str(repo_limit) + " - " + name)
    print("Empty repo: " + name)
    with open("missing_repos.txt", "a") as missing_file:
      missing_file.write(name + "\n")
      missing_file.close()
    continue

  #Get number of collaborators
  try:
    num_contributors = repo.get_contributors().totalCount
  except GithubException as e:
    print(e)
    num_contributors = "inf"


  #Get number of issues
  num_issues = repo.get_issues().totalCount

  #Get commit activity stats for a repo
  commit_activity = repo.get_stats_commit_activity()

  #Get repo languages
  langs = repo.get_languages()

  #Get most used repo language
  big_lang = repo.language


  #print("{}\n\tCommits: {}\n\tContributors: {}\n\tIssues: {}\n\tStars: {}".
format(name, num_commits, num_contributors, num_issues, num_stars))

  #print("\n\tLanguages: {}".format(langs))

  #Get commit activity stats for a repo
  commit_activity = repo.get_stats_commit_activity()


  #print("\n\tCommit activity:")

  commits_per_week = []
  json_commit_activity = []
  dict_year = {}
  for activity in commit_activity:
    commitWeek = []
    dict_week = {}
    commits_per_week.append(activity.total)

    days_string = "["
    for days in activity.days:
      days_string += "{}, ".format(days)
      commitWeek.append(days)
    days_string = days_string[:-2]
    days_string += "]"

    year = activity.week.isocalendar()[0]
    week_number = activity.week.isocalendar()[1]

    #if activity.total != 0:
      #print("\t{}, W: {}\t Total: {}\t Per day: {}".format(year, week_number,
activity.total, days_string))

    if dict_year.get(year) == None:
      dict_year[year] = {}
    for x in commitWeek:
      if x != 0:
```

```python
          dict_year[year][week_number] = commitWeek
          break

    dict_entry = {
                  "name": name,
                  "commits": num_commits,
                  "contributors": num_contributors,
                  "stars": num_stars,
                  "primary_language": big_lang,
                  "languages" : langs,
                  "commit_activity": dict_year
                }
    repo_dict.append(dict_entry)
    parsed_data.append(dict_entry)
    try:
      with open("data.json", "r") as read_file:
        parsed_data = json.load(read_file)
        parsed_data.append(dict_entry)
        read_file.close()
    except:
      pass

    with open("data.json", "w") as write_file:
      json.dump(parsed_data, write_file, indent=2)
      write_file.close()

    print("Repo " + str(curr_repo) + " of " + str(repo_limit) + " - " + name)

  t1 = time.time()
  timediff = t1 - t0
  timediff = int(timediff)
  minutes = int(timediff / 60)
  seconds = int(timediff % 60)
  print("\nFetching complete.\nElapsed time: " + str(minutes) + " min " + str(seconds) + " sec.\n")
  #with open("data.json", "w") as write_file:
  # json.dump(repo_dict, write_file, indent=2)

if __name__ == "__main__":
  i = 0
  while i < 1:
    download_repos(ACCESS_TOKEN)
    download_repos(ACCESS_TOKEN_2)
    i = i + 1
```

JupyterNotebook

June 7, 2020

1 Setup

```
[3]: import pandas as pd
     import glob
     import csv
     import matplotlib.pyplot as plt
```

```
[4]: path = "./data"
```

```
[5]: # Load csv's into dataframes
     year2015 = pd.read_csv(path + '/2015.csv', index_col=None, header=0)
     year2016 = pd.read_csv(path + '/2016.csv', index_col=None, header=0)
     year2017 = pd.read_csv(path + '/2017.csv', index_col=None, header=0)
     year2018 = pd.read_csv(path + '/2018.csv', index_col=None, header=0)
     year2019 = pd.read_csv(path + '/2019.csv', index_col=None, header=0)
     year2020 = pd.read_csv(path + '/2020-until_april26.csv', index_col=None,␣
      ↪header=0)
```

```
[6]: # Create dataframe with data from all years
     all_year_list = [year2015, year2016, year2017, year2018, year2019, year2020]
     all_years = pd.concat(all_year_list, axis=0, ignore_index=True)
```

```
[7]: # Filter out repos in the category "New Repo"
     new_repos = all_years.loc[all_years['List Status'] == 'New Repository']
     repos = all_years.loc[all_years['List Status'] != 'New Repository']
```

```
[8]: # Filter out repos that are not in the category "New Repo" for each year
     repos2015 = year2015.loc[year2015['List Status'] != 'New Repository']
     repos2016 = year2016.loc[year2016['List Status'] != 'New Repository']
     repos2017 = year2017.loc[year2017['List Status'] != 'New Repository']
     repos2018 = year2018.loc[year2018['List Status'] != 'New Repository']
     repos2019 = year2019.loc[year2019['List Status'] != 'New Repository']
     repos2020 = year2020.loc[year2020['List Status'] != 'New Repository']
```

2 Analysis

```
[9]: # Total amount of entries
     len(repos)
```

```
[9]: 24306
```

```
[10]: # Amount of days in the dataset
      len(repos['Date'].unique())
```

```
[10]: 1940
```

```
[11]: # Number of entries for a specific year
      rep = year2015
      len(rep.loc[rep['List Status'] != 'New Repository'])
```

```
[11]: 5465
```

```
[13]: # Numer of unique repositories each year
      unique_repos = repos2020['Repository'].unique().tolist()
      len(unique_repos)
```

```
[13]: 462
```

```
[14]: # Writes new csv file, uncomment if new is needed
      # all_years.to_csv('./data/all_years.csv')
```

3 Repositories

```
[15]: # Writes new csv file, uncomment if new is needed
      #pd.DataFrame(all_years['Repository'].unique()).to_csv('./data/all_unique_repos.
       ↪csv', index=False, header=False)
```

```
[16]: # List of repositories that have made the list the most amount of times
      repo_counts_series = all_years['Repository'].value_counts()
      repo_counts = pd.DataFrame(repo_counts_series)
      repo_counts.head(15)
```

```
[16]:                                      Repository
      FreeCodeCamp/FreeCodeCamp                   245
      kamranahmedse/developer-roadmap             101
      vuejs/vue                                    99
      jlevy/the-art-of-command-line                96
      danistefanovic/build-your-own-x              82
      MisterBooo/LeetCodeAnimation                 77
```

```
trekhleb/javascript-algorithms           72
sindresorhus/awesome                     72
CyC2018/Interview-Notebook               70
vhf/free-programming-books               65
flutter/flutter                          62
toddmotto/public-apis                    56
testerSunshine/12306                     54
tensorflow/tensorflow                    52
scutan90/DeepLearning-500-questions      51
```

4 Programming language

```
[17]: # Amount of unique programming languages
      unique_languages = repos['Primary Programming Language'].unique().tolist()
      len(unique_languages)
```

```
[17]: 97
```

5 Making of graph

```
[18]: popular = repos['Primary Programming Language'].value_counts().head(11)
      popular2015 = repos2015['Primary Programming Language'].value_counts().head(11)
      popular2016 = repos2016['Primary Programming Language'].value_counts().head(11)
      popular2017 = repos2017['Primary Programming Language'].value_counts().head(11)
      popular2018 = repos2018['Primary Programming Language'].value_counts().head(11)
      popular2019 = repos2019['Primary Programming Language'].value_counts().head(11)
      popular2020 = repos2020['Primary Programming Language'].value_counts().head(11)
```

```
[19]: popular2015 = popular2015.rename("2015").map(lambda x: x/5363)
      popular2016 = popular2016.rename("2016").map(lambda x: x/5174)
      popular2017 = popular2017.rename("2017").map(lambda x: x/4501)
      popular2018 = popular2018.rename("2018").map(lambda x: x/4069)
      popular2019 = popular2019.rename("2019").map(lambda x: x/3863)
      popular2020 = popular2020.rename("2020").map(lambda x: x/1234)
```

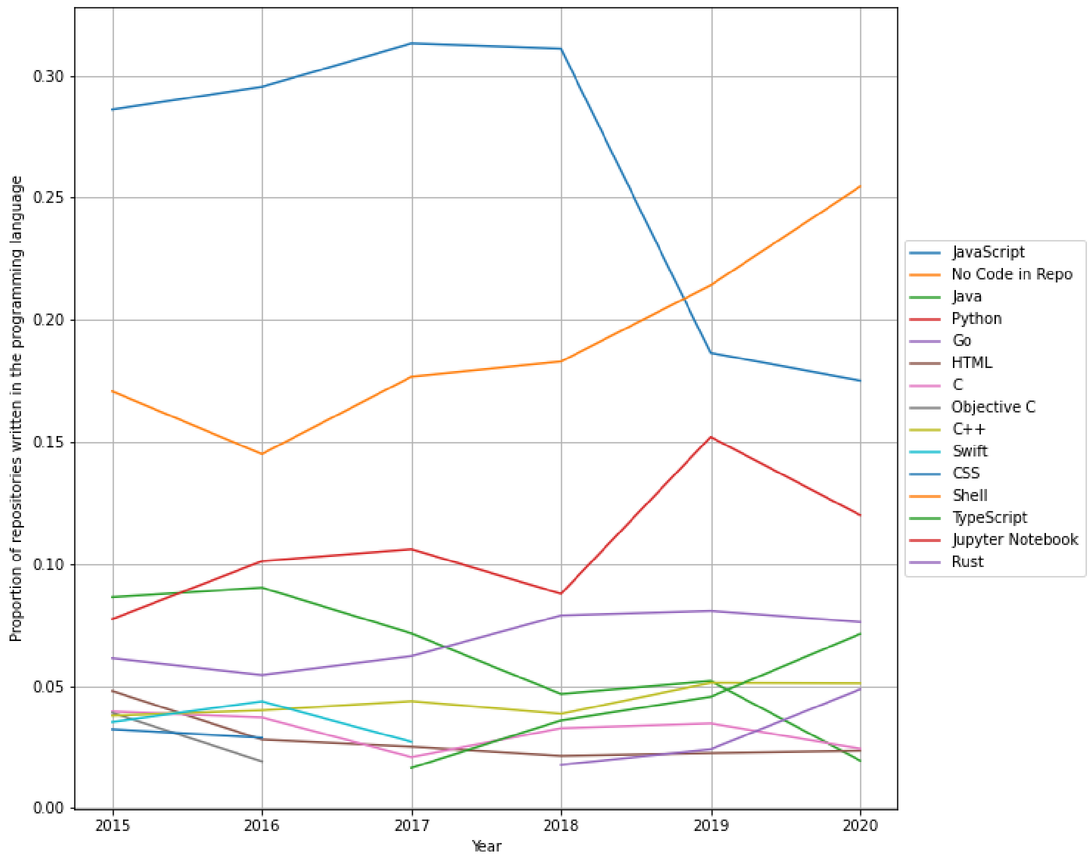
```
[20]: result = pd.concat([popular2015, popular2016, popular2017, popular2018,␣
       ↪popular2019, popular2020], axis = 1)
```

```
[20]:                              2015      2016      2017      2018      2019  \
      JavaScript               0.286034  0.295323  0.313042  0.310887  0.186384
      [No code in repository]  0.170613  0.144956  0.176627  0.182846  0.214082
      Java                     0.086332  0.090259  0.071540  0.046695  0.052032
      Python                   0.077382  0.101082  0.105976  0.087737  0.151954
```

```
Go                         0.061346  0.054503  0.062208  0.078889  0.080766
HTML                       0.047921  0.028025  0.025106  0.021381  0.022521
C                          0.039530  0.037109  0.020884  0.032686  0.034688
Objective-C                0.038971  0.019134       NaN       NaN       NaN
C++                        0.037852  0.040008  0.043768  0.038584  0.051256
Swift                      0.035241  0.043680  0.027105       NaN       NaN
CSS                        0.032258  0.028798       NaN  0.029983       NaN
TypeScript                      NaN       NaN  0.016441  0.035881  0.045560
Shell                           NaN       NaN  0.014663       NaN  0.023039
Rust                            NaN       NaN       NaN  0.017695  0.024075
Jupyter Notebook                NaN       NaN       NaN       NaN       NaN

                               2020
JavaScript                 0.175041
[No code in repository]    0.254457
Java                       0.019449
Python                     0.119935
Go                         0.076175
HTML                       0.023501
C                          0.024311
Objective-C                     NaN
C++                        0.051053
Swift                           NaN
CSS                             NaN
TypeScript                 0.071313
Shell                           NaN
Rust                       0.048622
Jupyter Notebook           0.017828
```

[21]:
```
year = ['2015', '2016', '2017', '2018', '2019', '2020']

javascript = result.loc['JavaScript']
no_code = result.loc['[No code in repository]']
java = result.loc['Java']
python = result.loc['Python']
go = result.loc['Go']
html = result.loc['HTML']
c = result.loc['C']
objective_c = result.loc['Objective-C']
c_plus = result.loc['C++']
swift = result.loc['Swift']
css = result.loc['CSS']
shell = result.loc['Shell']
typescript = result.loc['TypeScript']
jupyter = result.loc['Jupyter Notebook']
rust = result.loc['Rust']
```

[22]:
```python
fig, ax = plt.subplots()
ax.plot(year, javascript, label = 'JavaScript')
ax.plot(year, no_code, label = 'No Code in Repo')
ax.plot(year, java, label = 'Java')
ax.plot(year, python, label = 'Python')
ax.plot(year, go, label = 'Go')
ax.plot(year, html, label = 'HTML')

ax.plot(year, c, label = 'C')
ax.plot(year, objective_c, label = 'Objective C')
ax.plot(year, c_plus, label = 'C++')
ax.plot(year, swift, label = 'Swift')
ax.plot(year, css, label = 'CSS')
ax.plot(year, shell, label = 'Shell')
ax.plot(year, typescript, label = 'TypeScript')
ax.plot(year, jupyter, label = 'Jupyter Notebook')
ax.plot(year, rust, label = 'Rust')

ax.legend(loc = 'center left', bbox_to_anchor=(1,0.5))

fig.set_size_inches(10, 10)
ax.set_ylabel('Proportion of repositories written in the programming language')
ax.set_xlabel('Year')

ax = plt.gca()
ax.grid(True)


plt.show()
```



6 No code repos

```
[23]: # Get all repositorise without code
      no_code = repos.loc[repos['Primary Programming Language'] == '[No code in␣
       ↪repository]']
```

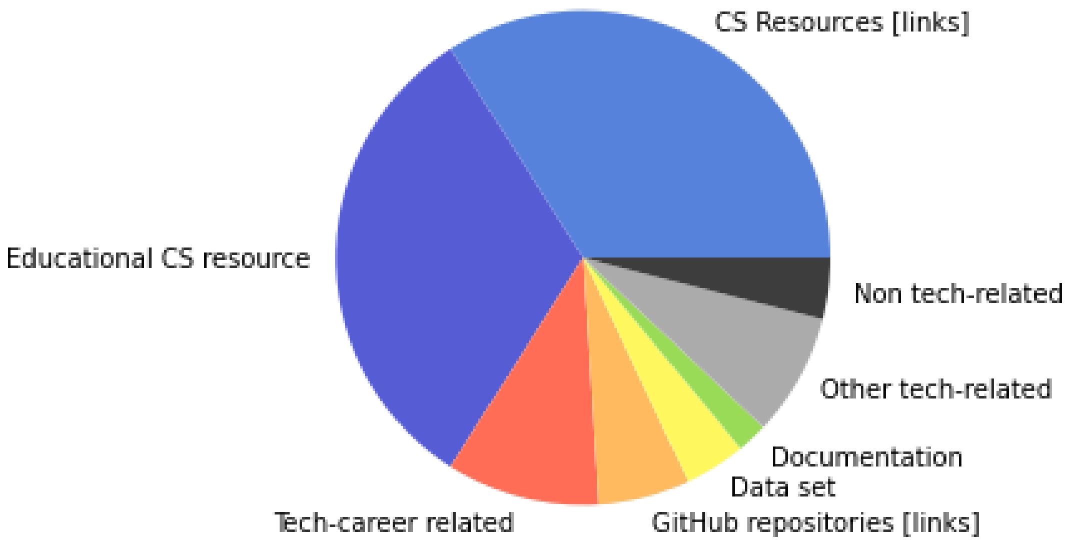
```
[24]: # Amount of repositories without code
      len(no_code)
```

```
[24]: 4345
```

```
[25]: # Generate a random sample of these repositories
      # Note: This sample might contain duplicates of the same repository
      random_sample = no_code.sample(50)
```

Making chart from results

```
[26]: labels = 'CS Resources [links]', 'Educational CS resource', 'Tech-career␣
      ↪related', 'GitHub repositories [links]', 'Data set', 'Documentation', 'Other␣
      ↪tech-related', 'Non tech-related'
      sizes = [17,16,5,3,2,1,4,2]
      colors = ['#5682db', '#555cd4', '#ff6d57', '#ffb95e', '#fff75e', '#99db56',␣
      ↪'#ababab', '#3d3d3d']

      fig1, ax1 = plt.subplots()
      ax1.pie(sizes, labels = labels, colors = colors)
      ax1.axis('equal')



      plt.show()
```



```
[ ]:
```

TRITA-EECS-EX-2020:363