Stap 1:

# Importing pandas

import pandas as pd

# Loading in the data

pulls\_one = pd.read\_csv('datasets/pulls\_2011-2013.csv')

pulls\_two = pd.read\_csv('datasets/pulls\_2014-2018.csv')

pull\_files = pd.read\_csv('datasets/pull\_files.csv')

Import: <https://www.w3schools.com/python/python_modules.asp>

Pd.read\_csv: https://www.w3schools.com/python/pandas/pandas\_csv.asp

Stap 2:

# Append pulls\_one to pulls\_two

pulls = pulls\_two.append(pulls\_one, ignore\_index=True)

# Convert the date for the pulls object

pulls['date'] = pd.to\_datetime(pulls['date'], utc=True)

.Append: <https://www.w3schools.com/python/ref_list_append.asp>

.To\_datetime: https://www.w3schools.com/python/pandas/pandas\_cleaning\_wrong\_format.asp

Stap 3:

# Merge the two DataFrames

data = pulls.merge(pull\_files, on='pid')

.Merge: https://www.w3schools.com/python/pandas/ref\_df\_merge.asp

Stap 4:

%matplotlib inline

# Create a column that will store the month

data['month'] = data['date'].dt.month

# Create a column that will store the year

data['year'] = data['date'].dt.year

# Group by month\_year and count the pull requests

counts = data.groupby(['year', 'month'])['pid'].count()

# Plot the results

counts.plot(kind='bar', figsize = (12,4))

%matplotlib inline: https://www.geeksforgeeks.org/how-to-use-matplotlib-plot-inline/?ref=gcse

.Dt.month: <https://www.geeksforgeeks.org/python-pandas-series-dt-month/>

.Dt.year: <https://www.geeksforgeeks.org/python-pandas-series-dt-year/?ref=gcse>

.groupby: <https://www.w3schools.com/python/pandas/ref_df_groupby.asp>

.count: <https://www.w3schools.com/python/ref_list_count.asp>

.plot: https://www.w3schools.com/python/matplotlib\_plotting.asp

Stap 5:

# Required for matplotlib

%matplotlib inline

# Group by the submitter

by\_user = data.groupby('user').agg({'pid': 'count'})

# Plot the histogram

by\_user.hist()

.agg: <https://www.w3schools.com/python/pandas/ref_df_agg.asp>

.hist: https://www.w3schools.com/python/matplotlib\_histograms.asp

Stap 6:

# Identify the last 10 pull requests

last\_10 = pulls.sort\_values(by = 'date').tail(10)

last\_10

# Join the two data sets

joined\_pr = pull\_files.merge(last\_10, on='pid')

# Identify the unique files

files = set(joined\_pr['file'])

# Print the results

files

.tail: https://www.w3schools.com/python/pandas/ref\_df\_tail.asp

.sort\_values: <https://www.w3schools.com/python/pandas/ref_df_sort_values.asp>

Set: https://www.geeksforgeeks.org/sets-in-python/?ref=gcse

Stap 7:

# This is the file we are interested in:

file = 'src/compiler/scala/reflect/reify/phases/Calculate.scala'

# Identify the pull requests that changed the file

file\_pr = data[data['file'] == file]

# Count the number of changes made by each developer

author\_counts = file\_pr.groupby('user').count()

# Print the top 3 developers

author\_counts.nlargest(3, 'file')

Dataframes: <https://www.geeksforgeeks.org/python-pandas-dataframe/?ref=gcse>

.nlargest: https://www.w3schools.com/python/pandas/ref\_df\_nlargest.asp

Stap 8:

file = 'src/compiler/scala/reflect/reify/phases/Calculate.scala'

# Select the pull requests that changed the target file

file\_pr = pull\_files[pull\_files['file'] == file]

# Merge the obtained results with the pulls DataFrame

joined\_pr = pulls.merge(file\_pr, on='pid')

# Find the users of the last 10 most recent pull requests

users\_last\_10 = set(joined\_pr.nlargest(10, 'date')['user'])

# Printing the results

users\_last\_10

Stap 9:

%matplotlib inline

# The developers we are interested in

authors = ['xeno-by', 'soc']

# Get all the developers' pull requests

by\_author = pulls[pulls['user'].isin(authors)]

# Count the number of pull requests submitted each year

counts = by\_author.groupby([by\_author['user'], by\_author['date'].dt.year]).agg({'pid': 'count'}).reset\_index()

# Convert the table to a wide format

counts\_wide = counts.pivot\_table(index='date', columns='user', values='pid', fill\_value=0)

# Plot the results

counts\_wide.plot(kind='bar')

.isin: <https://www.w3schools.com/python/pandas/ref_df_isin.asp>

.reset\_index: <https://www.w3schools.com/python/pandas/ref_df_reset_index.asp>

.pivot\_table: https://www.geeksforgeeks.org/python-pandas-pivot\_table/?ref=gcse

Stap 10:

authors = ['xeno-by', 'soc']

file = 'src/compiler/scala/reflect/reify/phases/Calculate.scala'

# Merge DataFrames and select the pull requests by the author

by\_author = data[data['user'].isin(authors)]

# Select the pull requests that affect the file

by\_file = by\_author[by\_author['file'] == file]

# Group and count the number of PRs done by each user each year

grouped = by\_file.groupby(['user', by\_file['date'].dt.year]).count()['pid'].reset\_index()

# Transform the data into a wide format

by\_file\_wide = grouped.pivot\_table(index='date', columns='user', values='pid', fill\_value=0)

# Plot the results

by\_file\_wide.plot(kind='bar')

Github link: https://github.com/Fausto2412/Scala\_opdracht

Github data vinden en downloaden:

Er zijn verschillende manieren om gegevens van GitHub te vinden en te downloaden, afhankelijk van wat je zoekt. Hier zijn enkele algemene methoden:

1. Zoek naar een GitHub-repository die de gegevens bevat die je zoekt. Repositories bevatten vaak gegevens zoals broncode, documentatie en datasets. Je kunt zoeken naar repositories door de GitHub-zoekfunctie te gebruiken of door te browsen door populaire en trending repositories.
2. Als je op zoek bent naar specifieke soorten gegevens, zoals datasets, kun je gebruik maken van een zoekmachine voor datasets, zoals datasearch. Deze zoekmachines indexeren gegevens die openbaar beschikbaar zijn op GitHub en andere opslagplaatsen.
3. Als je geïnteresseerd bent in specifieke gegevens van een bepaalde repository, kun je deze gegevens downloaden door de repository te clonen of te downloaden. Clonen betekent dat je een lokale kopie maakt van de gehele repository, terwijl downloaden alleen de inhoud van de huidige versie van de repository downloadt. Je kunt de repository clonen of downloaden door naar de repositorypagina te gaan en te klikken op de knop 'Clone or download'.
4. Als je geïnteresseerd bent in gegevens van meerdere repositories, kun je gebruik maken van de GitHub API om de gegevens te extraheren en te downloaden. De API stelt je in staat om te zoeken naar repositories, issues, pull requests en andere gegevens en deze te downloaden in verschillende formaten, zoals JSON en CSV. Je moet wel een account aanmaken en een persoonlijke toegangstoken genereren om toegang te krijgen tot de API.