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')

uitleg: eerst halen we codes op van de pandas gegevens bibliotheek om het in mijn python omgeving te zetten. Vervolgens hebben ik de drie bestanden geladen (laten lezen door Python).

<https://www.statology.org/import-pandas-as-pd/>

1. # Append pulls\_one to pulls\_two

pulls = pulls\_one.append(pulls\_two)

# Convert the date for the pulls object

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

uitleg: ik heb dataset Pulls\_one toegevoegd aan dataset Pulls\_two. Vervolgens heb ik de datums veranderd van dataset “pulls-object”

<https://www.digitalocean.com/community/tutorials/python-string-to-datetime-strptime>

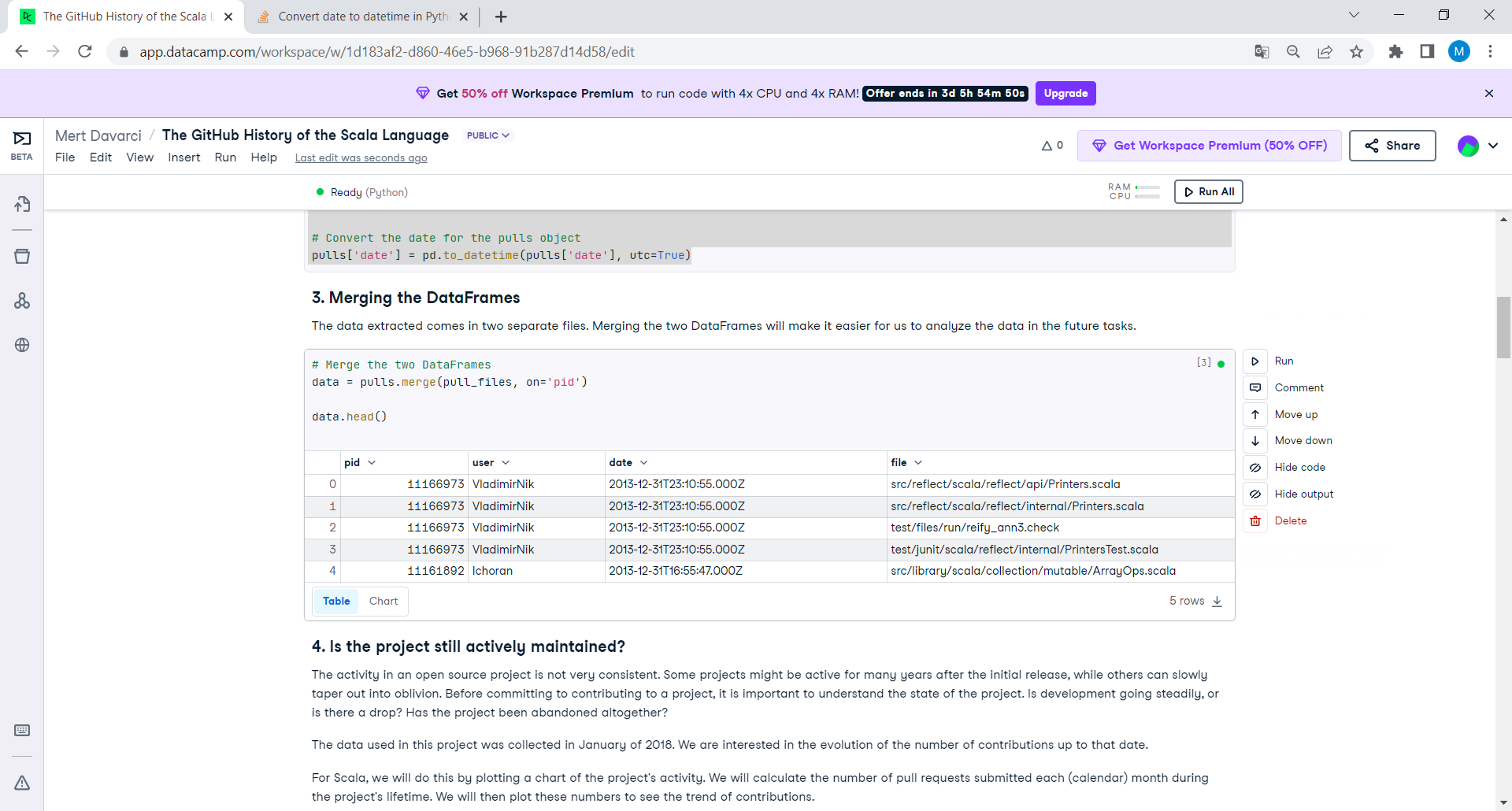
<https://stackoverflow.com/questions/1937622/convert-date-to-datetime-in-python>

1. # Merge the two DataFrames

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

data.head()

uitleg: ik heb de twee dataframes samengevoegd. En daarna heb ik het laten draaien. Ik heb een tabel te verschijn zien komen:



Hierop zijn de gebruikers, datums en bestanden te zien.

<https://app.datacamp.com/learn/courses/joining-data-with-pandas?utm_source=google&utm_medium=paid_search&utm_campaignid=19111730781&utm_adgroupid=144885540180&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative=638890154855&utm_targetid=aud-299261629574:dsa-1877181584102&utm_loc_interest_ms=&utm_loc_physical_ms=1010755&utm_content=dsa~page~courses&utm_campaign=220808_1-sea~dsa~courses_2-b2c_3-n-eu_4-prc_5-na_6-na_7-le_8-pdsh-go_9-na_10-na_11-na-marayc23&gclid=Cj0KCQjwtsCgBhDEARIsAE7RYh2PcEcaIS90hfgtlY71oXks9XLOjS6MjS6r-LNgqn9o__vtd2NmOJsaAu5REALw_wcB> (instuctie video)

1. %matplotlib inline

# Create a column that will store the month

data['month'] = pd.DatetimeIndex(data['date']).month

# Create a column that will store the year

data['year'] = pd.DatetimeIndex(data['date']).year

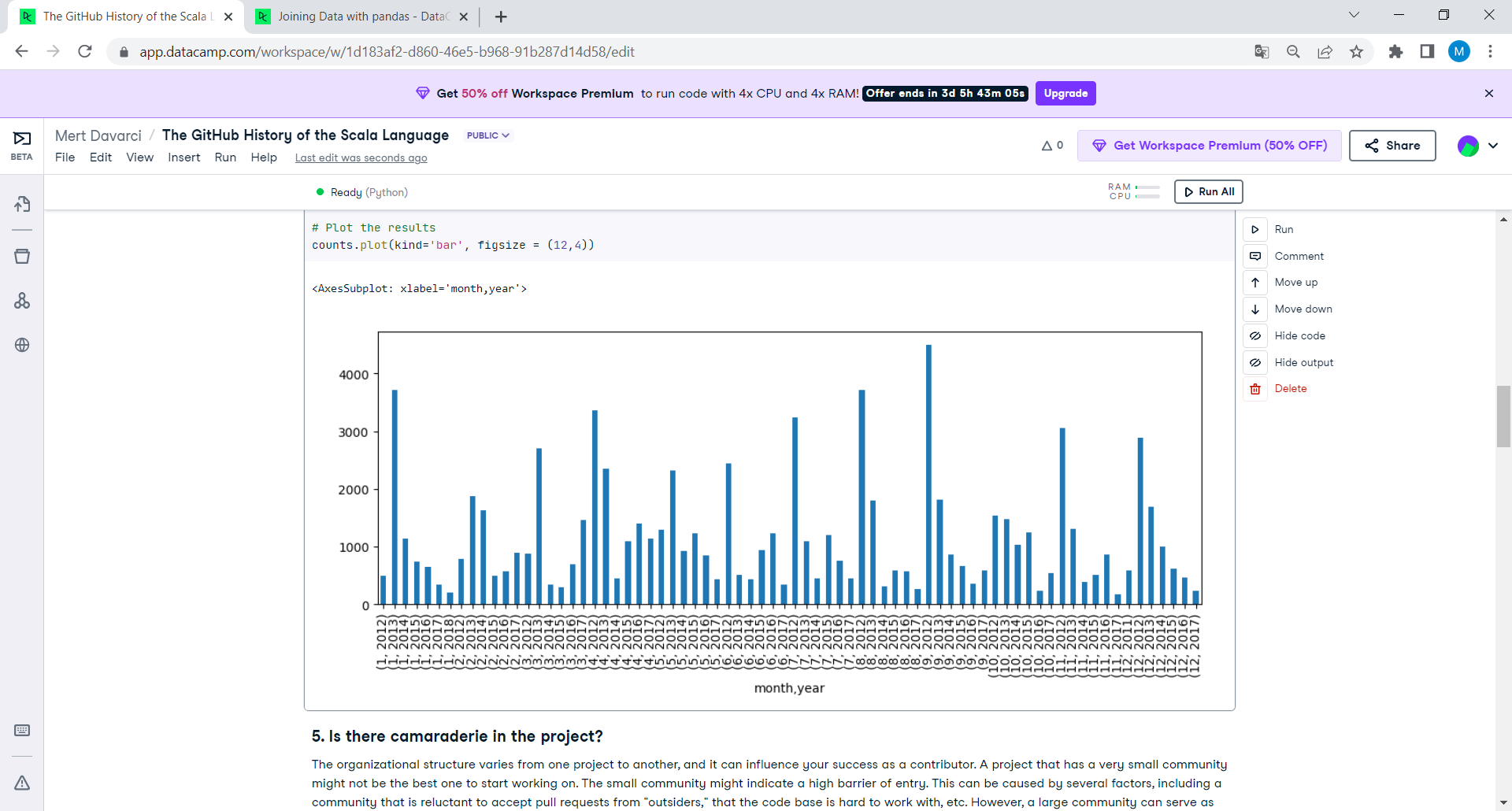
# Group by the month and year and count the pull requests

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

# Plot the results

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

uitleg: ik heb een kolom gemaakt waarin de “maand” wordt opgeslagen. Ik heb ook een kolom gemaakt waarin “jaar” wordt opgeslagen. Ik heb de pull- request gegroepeerd ik maand en jaar. Vervolgens heb ik de resultaten van de plot uit laten draaien, waardoor ik een staafdiagram kreeg te zien van de maanden en jaren:



<https://stackoverflow.com/questions/43027980/purpose-of-matplotlib-inline>

1. # Required for matplotlib

%matplotlib inline

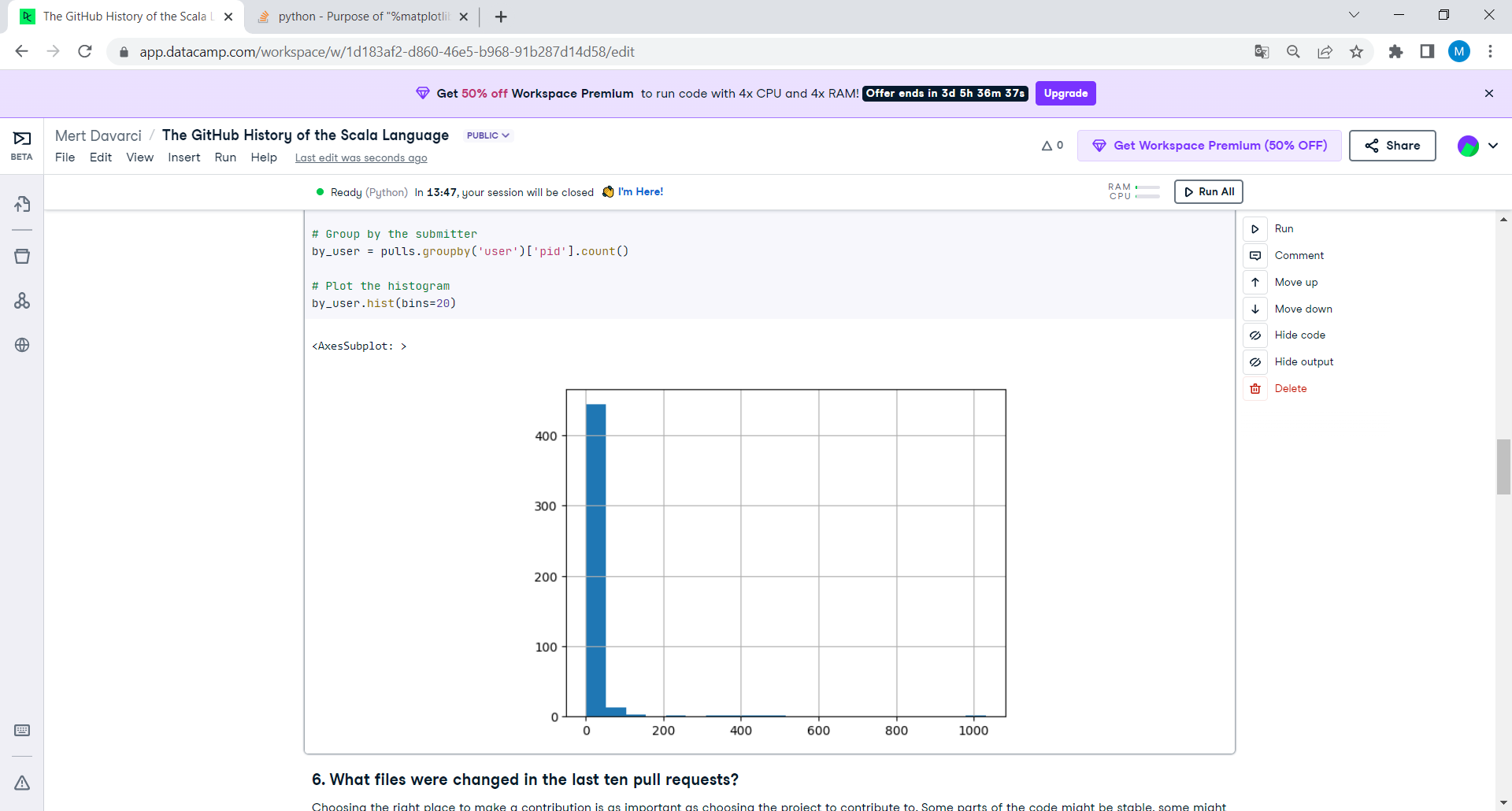
# Group by the submitter

by\_user = pulls.groupby('user')['pid'].count()

# Plot the histogram

by\_user.hist(bins=20)

uitleg: eerst heb ik matplotlib laten draaien op mijn Python omgeving. Vervolgens heb ik het gegroepeerd per gebruiker. Als laatst heb de histrogram laten draaien waardoor ik een grafiek zag:



<https://stackoverflow.com/questions/43027980/purpose-of-matplotlib-inline>

1. # Identify the last 10 pull requests

last\_10 = pulls.sort\_values('date', ascending=False)[:10]

# Join the two data sets

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

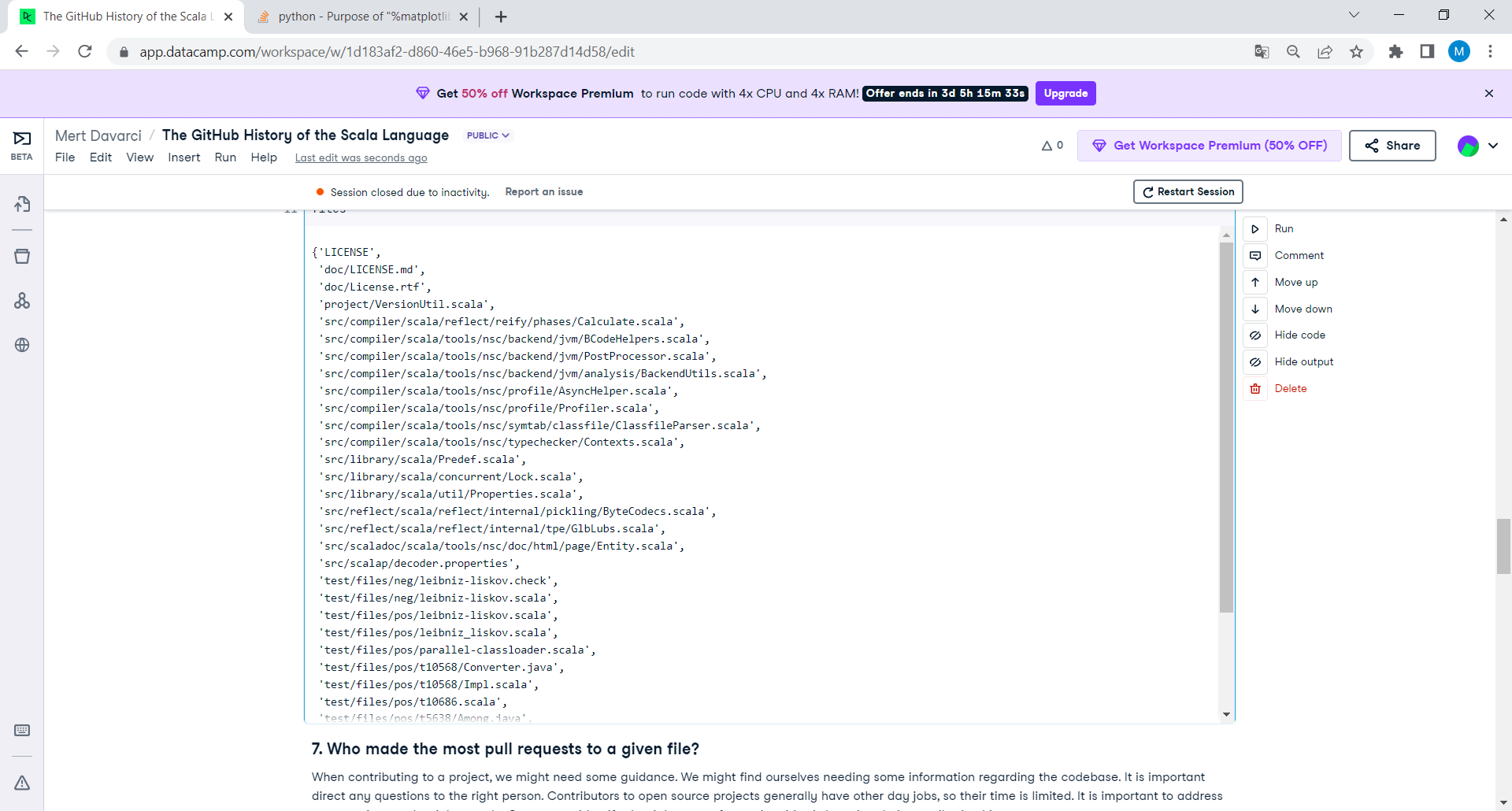
# Identify the unique files

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

# Print the results

files

uitleg: als eerst heb ik de laatste 10 pull requests geïdentificeerd. Daarna heb ik twee datasets toegevoegd aan de laatste 10 pull repuests. Vervolgens heb ik unique files geïdentificeerd. Als laatst heb ik de resultaat geprint.



<https://stackoverflow.com/questions/45427164/how-to-create-a-git-pull-request-in-gitpython>

1. # This is the file we are interested in:

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

# Identify the commits 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')['pid'].count()

# Print the top 3 developers

print(author\_counts.sort\_values(ascending=False)[:3])

uitleg: eerst druk ik af in welke bestand we geïnteresseerd zijn. Daarna identificeer ik commits die de betand hebben gewijzigd. Vervolgens laat ik de aantal wijzigingen dat door de ontwikkelaar is gemaakt tellen. Als laatst print ik de top 3 developers.

<https://python-gitlab.readthedocs.io/en/stable/gl_objects/commits.html>

1. 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 = file\_pr.merge(pulls, 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

uitleg: eerst heb ik het pull requests bestand geselecteerd. Daarna heb ik de resultaten samengevoegd, die ik bij Pull Dataframe kreeg. Daarna heb ik de laatste 10 gebruikers van pull request gevonden. Uiteindelijk heb ik de laatste 10 gebruikers geprint.

<https://stackoverflow.com/questions/646644/how-to-get-last-items-of-a-list-in-python>

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

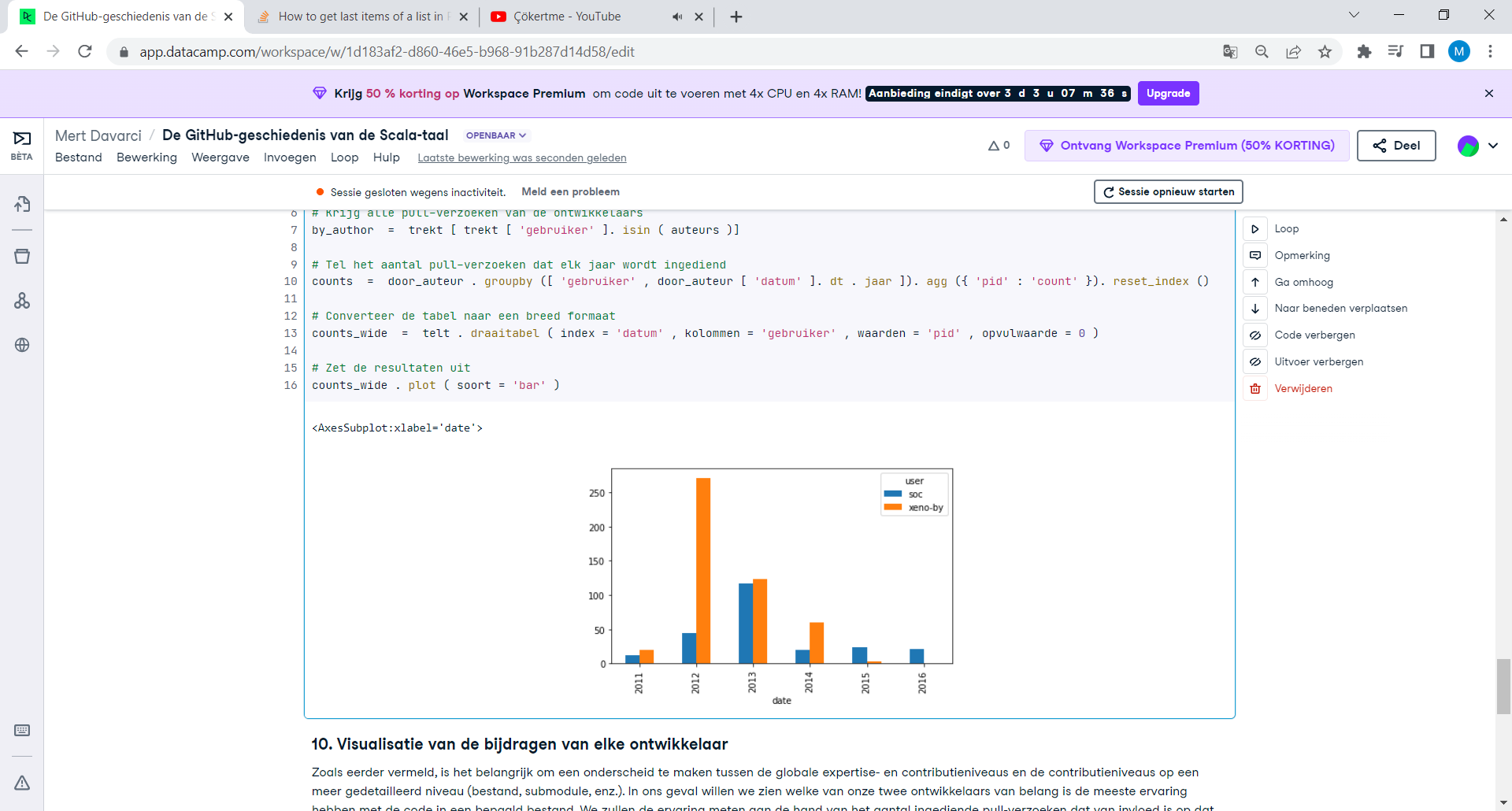
# Convert the table to a wide format

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

# Plot the results

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

uitleg: als eerst heb ik de developers geselecteerd waarin we geïnteresseerd zijn. Vervolgens heb ik alle pull request vann gebruikers in een lijst gezet. Vervolgens heb ik alle pull request dat elk jaar ingediend wordt, laten tellen. Daarna heb ik de tabellen veranderd naar een bredere formaat. Daarna heb ik de resultaten laten uitdraaien en kreeg ik een staagdiagram te zien:



<https://stackoverflow.com/questions/70007486/multiple-authors-for-blogpost>

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

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

# Select the pull requests submitted by the authors, from the `data` DataFrame

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()

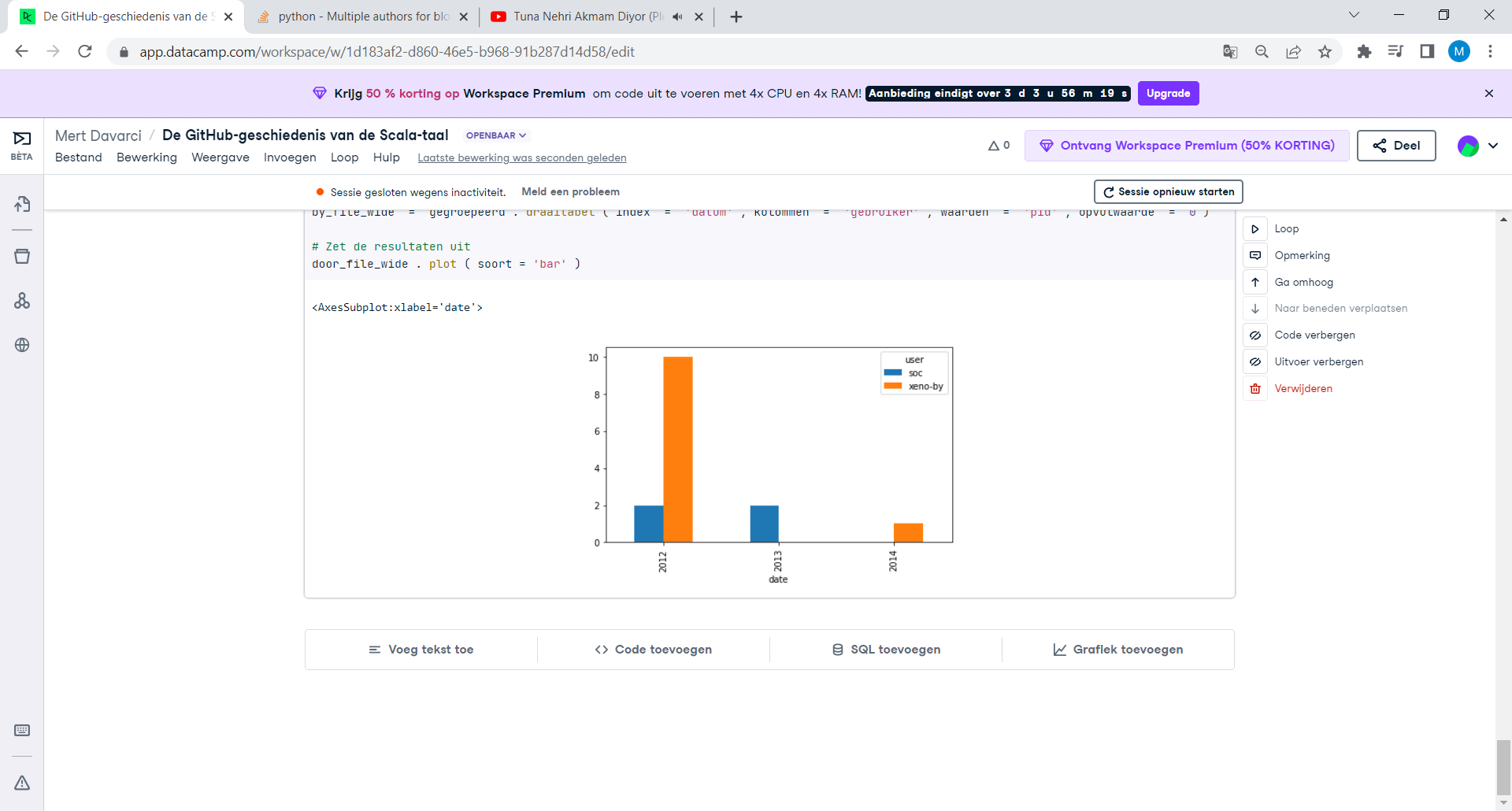
# Transform the data into a wide format

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

# Plot the results

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

uitleg: ik heb als eerst de pull request van de auteurs geselecteerd uit de ‘data’ dataframe. Daarna heb ik de pull requast geselecteerd die invloed hebben op de file. Vervolgens heb ik de aantal PR’s geteld en gegroepeerd dat elk jaar door gebruikers wordt gedaan. Ik heb de data getransformeerd in een bredere formaat. En uiteindelijk heb ik de plot van de resultaten laten uitdraaien:



<https://python.arviz.org/en/stable/contributing/pr_tutorial.html>