Final Project

Anish Gupta

5/10/2022

Object-Oriented Programming 04:547:202:03

My question is which classical music composer has the most articles written about them in the last 50 years. I will fetch the number of articles for each composer from a list of composers written over the last 50 years by decade. This information will be stored in a csv file. I will read this csv file and generate a line plot comparing the number of articles written for each composer over time. I will use this information to determine which composer has the most articles in the last 50 years. Once I have the name of the top composer, I will go back to the New York Times API to get the articles published over the last 10 years about the composer and store them in a csv file.

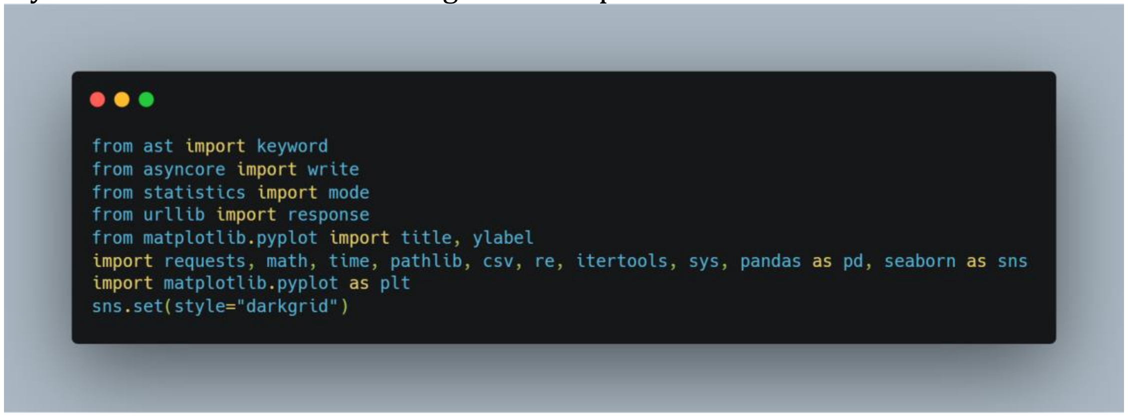
Technologies used for this project:

1. Python programming language for coding.
2. IDE used: VS Code.
3. Carbon for creating code images.
4. Imgbb for referencing code images.
5. Stackedit for writing Markdown file.
6. Pandoc and Microsoft Word for converting Markdown file to PDF.
7. New York Times API for data extraction.

Results:

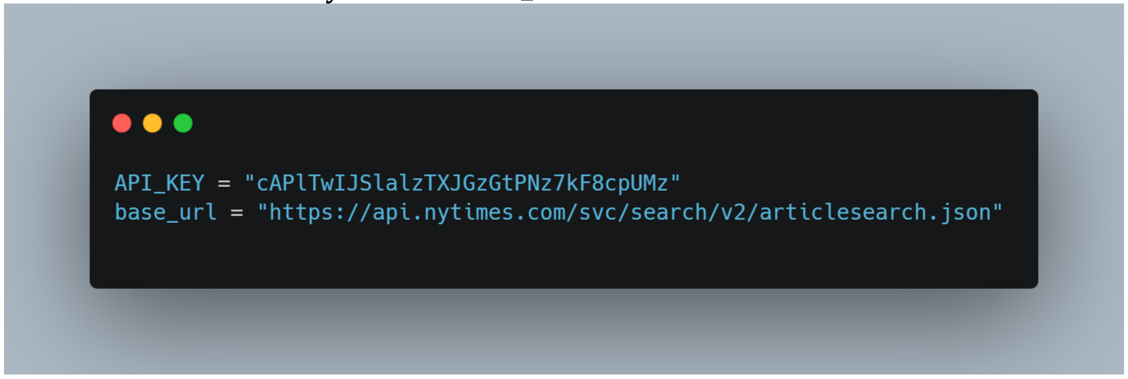
1. Comparing between the composers Beethoven, Mozart, Bach, and Chopin, it is found that Mozart is the most written about composer over the last 50 years.
2. In each decade, Mozart had the most articles written about him.
3. The overall trend of the number of articles written about all four composers appear to be declining.

My code starts with the following: I have imported the libraries here.



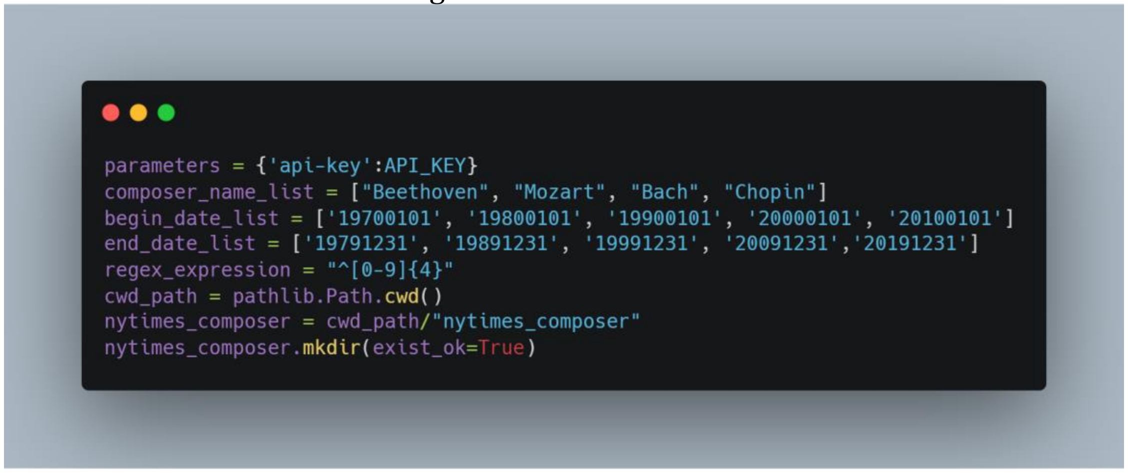
```
from ast import keyword
from asyncore import write
from statistics import mode
from urllib import response
from matplotlib.pyplot import title, ylabel
import requests, math, time, pathlib, csv, re, itertools, sys, pandas as pd, seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="darkgrid")
```

Here I define the API key and the base_url.



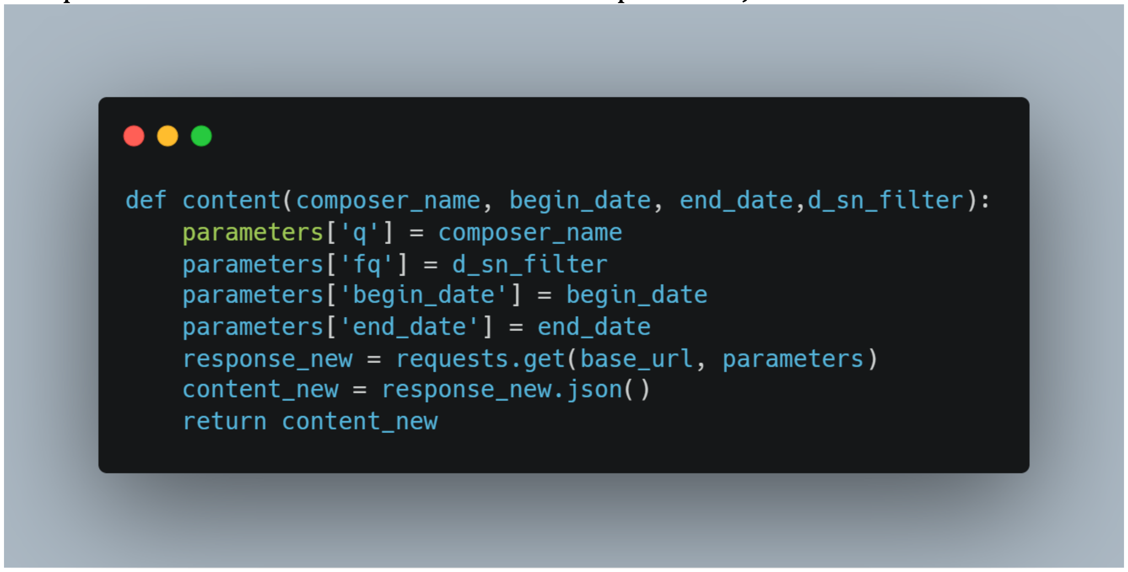
```
API_KEY = "cAPlTwIJSlalzTXJGzGtPNz7kF8cpUMz"
base_url = "https://api.nytimes.com/svc/search/v2/articlesearch.json"
```

Here I define variables and assign values.



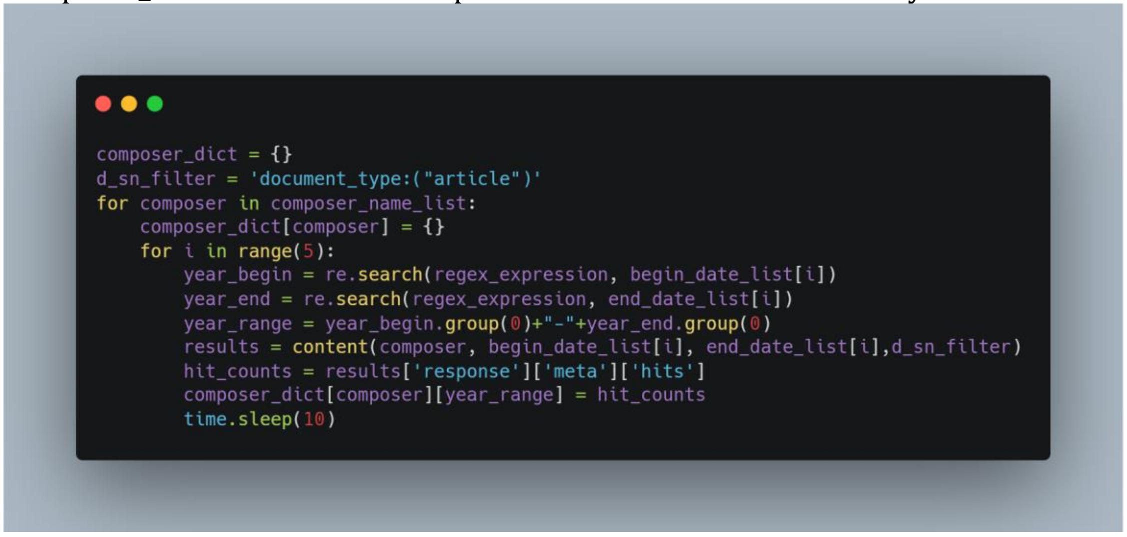
```
parameters = {'api-key':API_KEY}
composer_name_list = ["Beethoven", "Mozart", "Bach", "Chopin"]
begin_date_list = ['19700101', '19800101', '19900101', '20000101', '20100101']
end_date_list = ['19791231', '19891231', '19991231', '20091231','20191231']
regex_expression = "^[0-9]{4}"
cwd_path = pathlib.Path.cwd()
nytimes_composer = cwd_path/"nytimes_composer"
nytimes_composer.mkdir(exist_ok=True)
```

This function “content” pulls data from the New York Times API. The function takes in four parameters namely, composer_name, begin_date, end_date, and d_sn_filter. These four values are assigned to the New York Times API parameters. The API is called with base_url and parameters. The function returns the response as JSON.



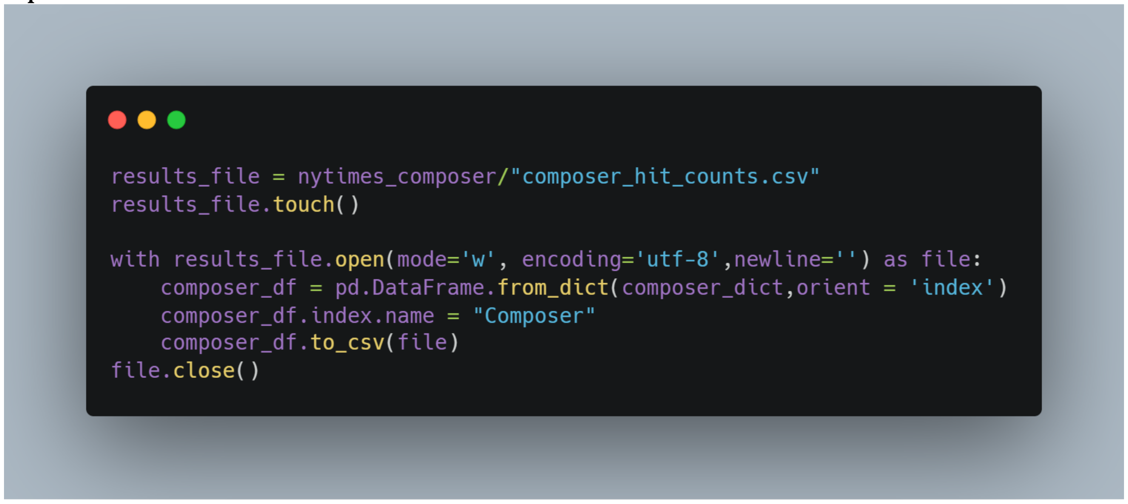
```python
def content(composer_name, begin_date, end_date,d_sn_filter):
    parameters['q'] = composer_name
    parameters['fq'] = d_sn_filter
    parameters['begin_date'] = begin_date
    parameters['end_date'] = end_date
    response_new = requests.get(base_url, parameters)
    content_new = response_new.json()
    return content_new
```

For each composer, get a date range from the begin_date_list and the end_date_list. Call the content function and use the results to get the number of hits. Create a dictionary called composer_dict and store the composers and their number of hits by each decade.



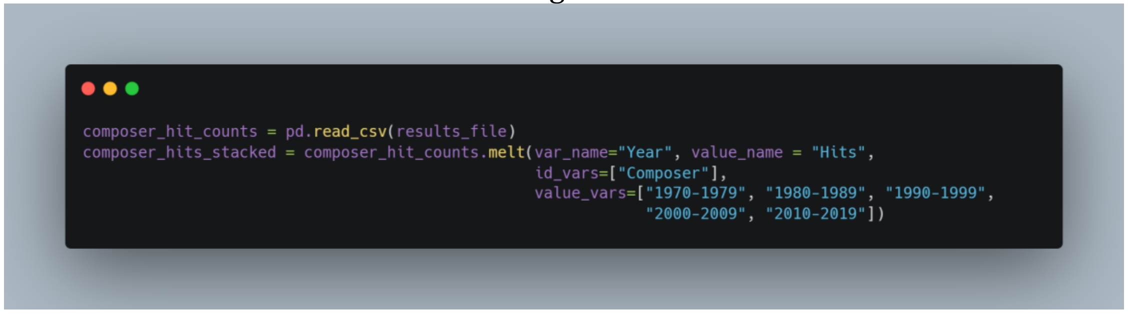
```python
composer_dict = {}
d_sn_filter = 'document_type:("article")'
for composer in composer_name_list:
    composer_dict[composer] = {}
    for i in range(5):
        year_begin = re.search(regex_expression, begin_date_list[i])
        year_end = re.search(regex_expression, end_date_list[i])
        year_range = year_begin.group(0)+"-"+year_end.group(0)
        results = content(composer, begin_date_list[i], end_date_list[i],d_sn_filter)
        hit_counts = results['response']['meta']['hits']
        composer_dict[composer][year_range] = hit_counts
        time.sleep(10)
```

Create a csv file to save the information in the composer_dict. Read the composer_dict into a pandas dataframe. Write the dataframe to a csv file. Close the file.



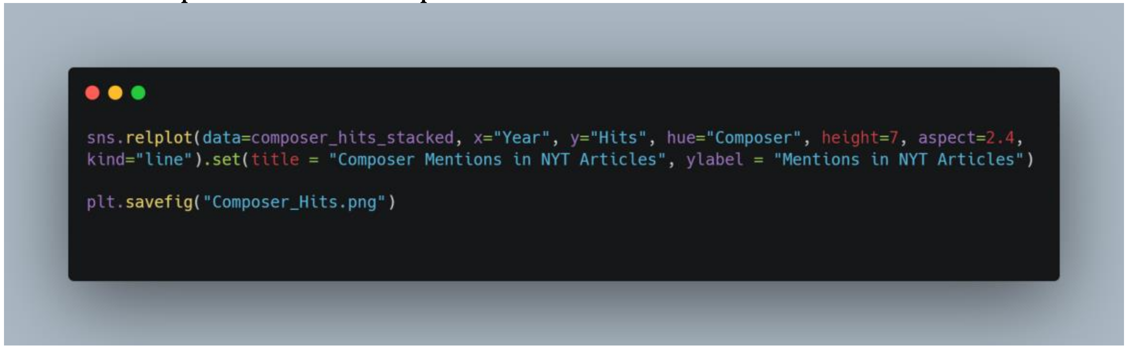
```
results_file = nytimes_composer/"composer_hit_counts.csv"
results_file.touch()

with results_file.open(mode='w', encoding='utf-8',newline='') as file:
    composer_df = pd.DataFrame.from_dict(composer_dict,orient = 'index')
    composer_df.index.name = "Composer"
    composer_df.to_csv(file)
file.close()
```

Read the composer_hit_counts from the csv file and save it to a dataframe. Use the melt function to convert the dataframe to long format.



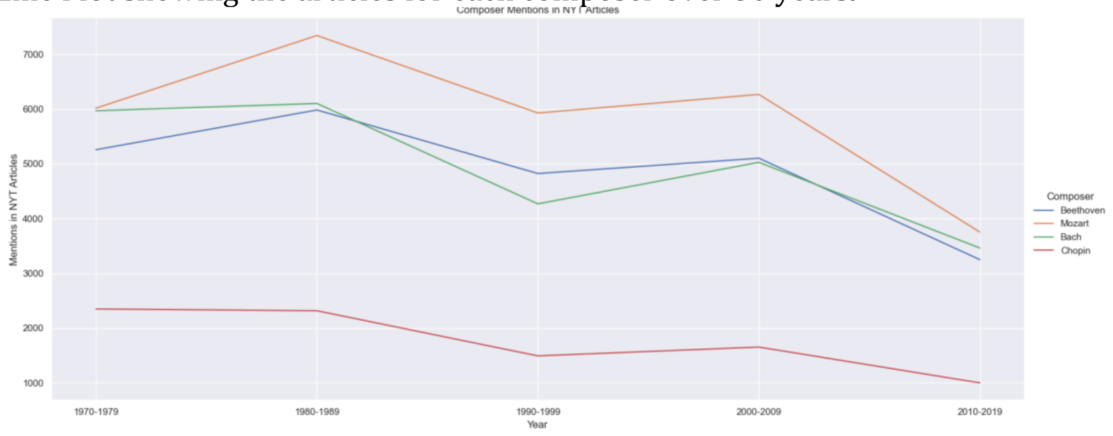
```
composer_hit_counts = pd.read_csv(results_file)
composer_hits_stacked = composer_hit_counts.melt(var_name="Year", value_name = "Hits",
                                                 id_vars=["Composer"],
                                                 value_vars=["1970-1979", "1980-1989", "1990-1999",
                                                             "2000-2009", "2010-2019"])
```

Create a line plot and save the plot.



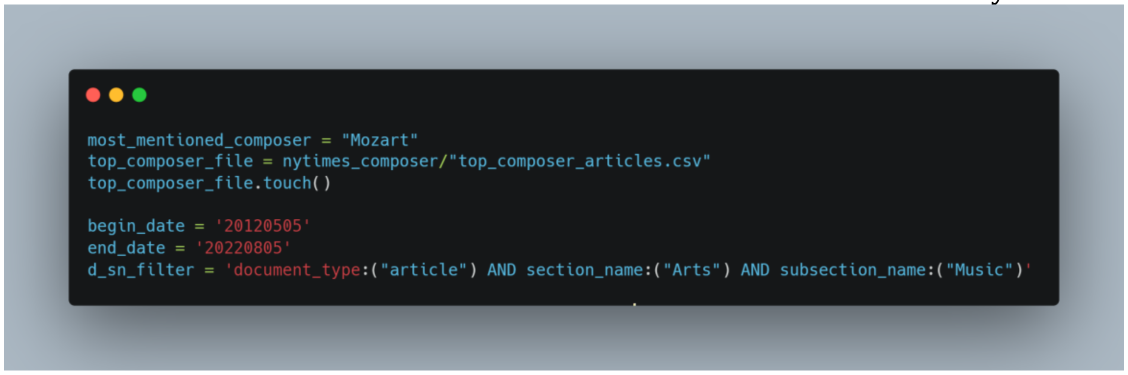
```
sns.relplot(data=composer_hits_stacked, x="Year", y="Hits", hue="Composer", height=7, aspect=2.4,
kind="line").set(title = "Composer Mentions in NYT Articles", ylabel = "Mentions in NYT Articles")

plt.savefig("Composer_Hits.png")
```

Line Plot showing the articles for each composer over 50 years.



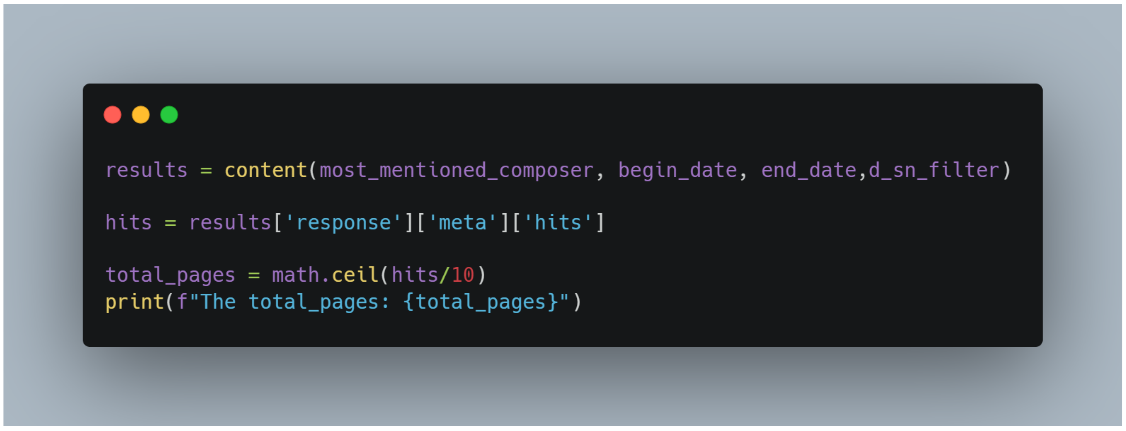
Looking at the plot, it is obvious that Mozart is the most written about composer over the last 50 years.

I will now search for articles on Mozart over the last 10 years. Create a file called top_composer_articles.csv to write the articles on Mozart over the last 10 years. Set the begin and end dates. Set the filters for New York Times API to document_type as "article", section_name as "Arts", and subsection_name as "Music". These extra parameters help narrow down the search and return articles that are related to music only.



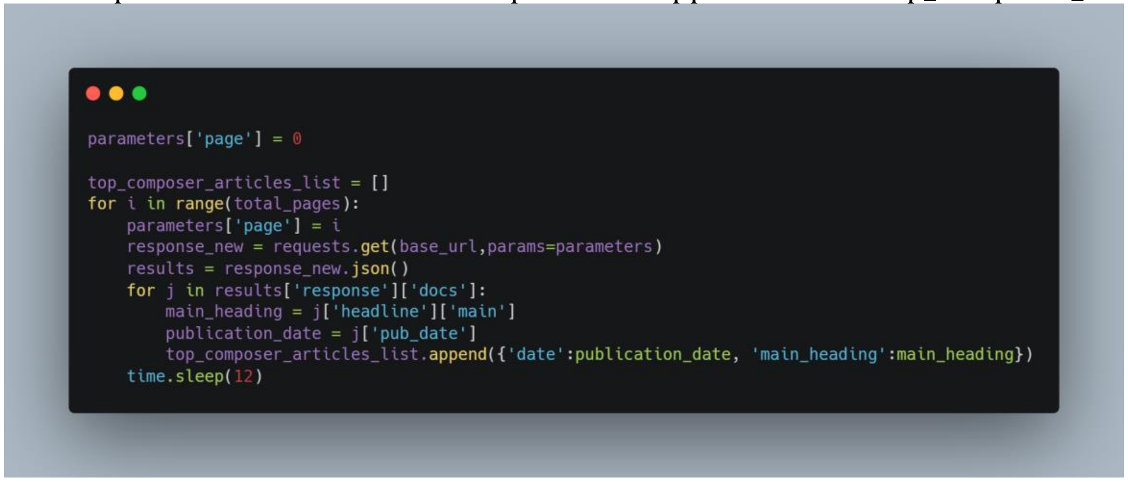
```
most_mentioned_composer = "Mozart"
top_composer_file = nytimes_composer/"top_composer_articles.csv"
top_composer_file.touch()

begin_date = '20120505'
end_date = '20220805'
d_sn_filter = 'document_type:("article") AND section_name:("Arts") AND subsection_name:("Music")'
```

Call the content function to get the articles on Mozart over the last 10 years. Get the number of hits for Mozart over the last 10 years. Calculate the total pages from the number of hits.



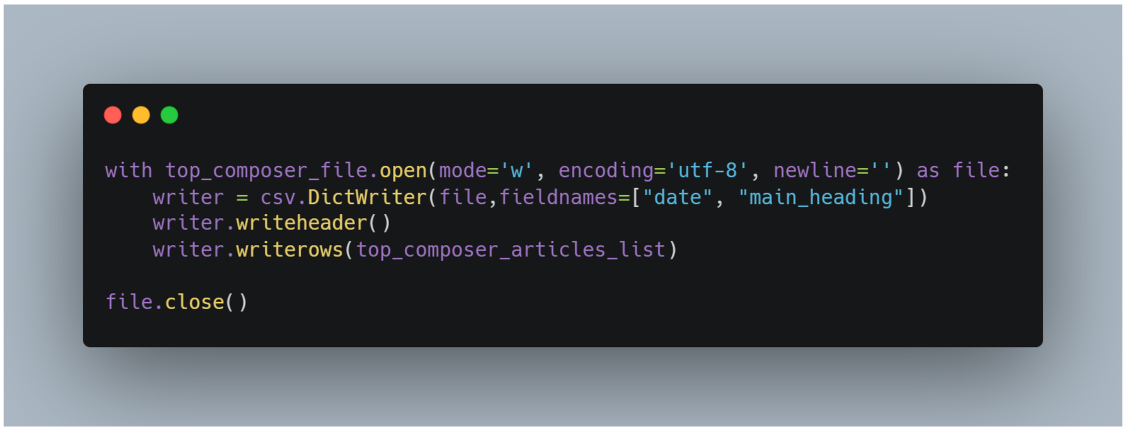
```python
results = content(most_mentioned_composer, begin_date, end_date,d_sn_filter)

hits = results['response']['meta']['hits']

total_pages = math.ceil(hits/10)
print(f"The total_pages: {total_pages}")
```

Create an empty list for storing all the articles for Mozart. Loop through each page and call the New York Times API to get the articles on Mozart page by page. Get the main heading and the publication date from the response and append it to the top_composer_articles_list.



```python
parameters['page'] = 0

top_composer_articles_list = []
for i in range(total_pages):
    parameters['page'] = i
    response_new = requests.get(base_url,params=parameters)
    results = response_new.json()
    for j in results['response']['docs']:
        main_heading = j['headline']['main']
        publication_date = j['pub_date']
        top_composer_articles_list.append({'date':publication_date, 'main_heading':main_heading})
    time.sleep(12)
```

Write the top_composer_articles_list to the top_composer_articles.csv file. Finally, close the file.



```python
with top_composer_file.open(mode='w', encoding='utf-8', newline='') as file:
    writer = csv.DictWriter(file,fieldnames=["date", "main_heading"])
    writer.writeheader()
    writer.writerows(top_composer_articles_list)

file.close()
```

What would I do next?

With more time, this project can be further enhanced to study the articles, stored in the csv file top_composer_articles.csv. I would like to see if there are certain trends like more articles written on Mozart during certain times of the year, year over year. This study can be taken beyond 10 years to look at historical trends.