Data Collection

A Quick Tutorial on Setting Up YouTube API Credentials

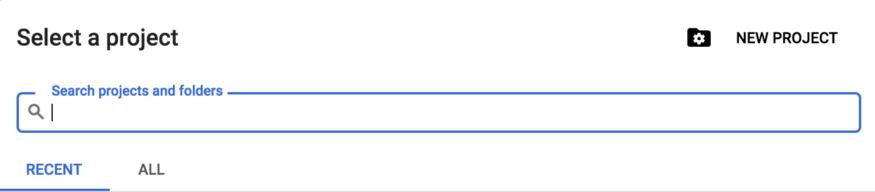
1. Head over to [Google Developer’s Console](https://console.developers.google.com/) and create a new project.





Click on the down arrow button





Choose the NEW PROJECT option (or select an existing one)



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

And give it a name!

2. Once you’ve set up a new project, select + ENABLE APIS AND SERVICES

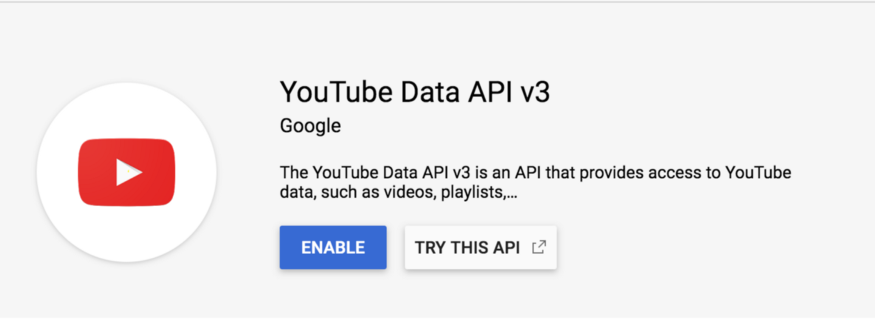


תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

4. Search for YouTube Data API v3 and click on Enable.





5. Then return to Credentials. You can do so by clicking on the hamburger menu, ☰



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

6. Select + CREATE CREDENTIALS, and then API Key.



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

According to the developer docs, [we do not need user authorization to retrieve information about a public YouTube channel](https://developers.google.com/youtube/v3/quickstart/python), so an API key is all we’ll need to collect comments off a video.

7. Finally, install the Google API Client for Python.

pip install --upgrade google-api-python-client

If you’re curious, you can read more about setting up Google APIs with Python here:

**[Python Quickstart | YouTube Data API | Google Developers](https://developers.google.com/youtube/v3/quickstart/python" \t "_blank)**

[This quickstart guide explains how to set up a simple, Python command-line application that makes requests to the…](https://developers.google.com/youtube/v3/quickstart/python" \t "_blank)

[developers.google.com](https://developers.google.com/youtube/v3/quickstart/python" \t "_blank)

Using YouTube Data API v3 to Query YouTube Comments

Once we have our credentials set up, we can now start collecting comments! We’ll first build the service for calling the YouTube API:

Now let’s take a look at the resource of interest. In order to obtain all YouTube comments on a specific video, we’ll need to send a request for [CommentThreads](https://developers.google.com/youtube/v3/docs/commentThreads/list). An example request in Python for a commentThread will look like the following:

# you only need to build the service once  
service = build\_service('path/to/apikey.json') response = service.commentThreads().list(  
 part='snippet',  
 maxResults=100,  
 textFormat='plainText',  
 order='time',  
 videoId='ioNng23DkIM'  
).execute()

Of the parameters listed above, there are two parameters that are required, **part**,and exactly one of **allThreadsRelatedToChannelId**, **channelId**, **id**, and **videoId**. For the **part** parameter, we need to pass a comma-separated list consisting of any combination of **id**, **snippet**, and **replies**. The **snippet** [keyword will return basic details about the comment thread and the thread’s top-level comment, while](https://developers.google.com/youtube/v3/docs/commentThreads#replies.comments[]) replies [contains a list of replies to the top level comment.](https://developers.google.com/youtube/v3/docs/commentThreads#replies.comments[])

The second required parameter is a filter, and we can choose between **allThreadsRelatedToChannelId**, **channelId**, **id**, and **videoId**. Since I was interested in just the YouTube comments on Blackpink’s *How You Like That*, I chose to filter by **videoId**.

A video’s ID can be obtained from its YouTube link. They will generally look like this:

<https://www.youtube.com/watch?v=ioNng23DkIM>

The video ID in this case will be ioNng23DkIM. And in general, the video ID follows ‘?v=’.

But sometimes a link may look like the following, such as when you obtain a link through the share option on a video:

<https://youtu.be/ioNng23DkIM>

In that case, the ID will be directly after ‘youtu.be’.

We can handle both cases with the [following function](https://stackoverflow.com/questions/45579306/get-youtube-video-url-or-youtube-video-id-from-a-string-using-regex) (although this is unnecessary if you’ll be manually sourcing YouTube video links. If that’s the case, you can just copy the ID part of the link.)

Deciding on the Items of Interest

For this project, I was only interested in top level comments, the number of replies and likes, and whether the commenter also rated (liked) the video, so I passed just the string ‘snippet’ to parameter part.

After running the code above, you’ll get a JSON response that [looks like this](https://developers.google.com/youtube/v3/docs/commentThreads/list):

{  
 "kind": "youtube#commentThreadListResponse",  
 "etag": etag,  
 "nextPageToken": string,  
 "pageInfo": {  
 "totalResults": integer,  
 "resultsPerPage": integer  
 },  
 "items": [  
 [commentThread Resource](https://developers.google.com/youtube/v3/docs/commentThreads#resource)  
 ]  
}

The items of interest are **nextPageToken** and **items**. Let’s talk about **items** first. The key **items** contains a list of **commentThreads**, and each **commentThread** consists of the [following](https://developers.google.com/youtube/v3/docs/commentThreads#resource):

{  
 "[**kind**](https://developers.google.com/youtube/v3/docs/commentThreads#kind)": "youtube#commentThread",  
 "**[etag](https://developers.google.com/youtube/v3/docs/commentThreads" \l "etag)**": **etag**,  
 "[**id**](https://developers.google.com/youtube/v3/docs/commentThreads#id)": **string**,  
 "[**snippet**](https://developers.google.com/youtube/v3/docs/commentThreads#snippet)": {  
 "**[channelId](https://developers.google.com/youtube/v3/docs/commentThreads" \l "snippet.channelId)**": **string**,  
 "**[videoId](https://developers.google.com/youtube/v3/docs/commentThreads" \l "snippet.videoId)**": **string**,  
 "**[topLevelComment](https://developers.google.com/youtube/v3/docs/commentThreads" \l "snippet.topLevelComment)**": [**comments Resource**](https://developers.google.com/youtube/v3/docs/comments#resource),  
 "**[canReply](https://developers.google.com/youtube/v3/docs/commentThreads" \l "snippet.canReply)**": **boolean**,  
 "**[totalReplyCount](https://developers.google.com/youtube/v3/docs/commentThreads" \l "snippet.totalReplyCount)**": **unsigned integer**,  
 "**[isPublic](https://developers.google.com/youtube/v3/docs/commentThreads" \l "snippet.isPublic)**": **boolean**  
 },  
 "[**replies**](https://developers.google.com/youtube/v3/docs/commentThreads#replies)": {  
 "[**comments**](https://developers.google.com/youtube/v3/docs/commentThreads#replies.comments[])": [  
 [**comments Resource**](https://developers.google.com/youtube/v3/docs/comments#resource)  
 ]  
 }  
}

Since I chose to pass only the string **snippet** to the part parameter, I will only get the **snippet** portion of the JSON resource above. The **snippet** is a dictionary containing keys and corresponding values for **channelId**, **videoId**, **topLevelComment**, **canReply**, **totalReplyCount**, and **isPublic**.

Among these resources, I chose to save the values of **topLevelComment** and **totalReplyCount**. However, we still have not accessed the actual text content of the **topLevelComment**. We can extract the text, the number of likes the top level comment has received, and whether the commenter has also rated the video by indexing into the **topLevelComment** object. It is a comment resource, which looks like this:

{  
 "[**kind**](https://developers.google.com/youtube/v3/docs/comments#kind)": "youtube#comment",  
 "**[etag](https://developers.google.com/youtube/v3/docs/comments" \l "etag)**": **etag**,  
 "[**id**](https://developers.google.com/youtube/v3/docs/comments#id)": **string**,  
 "[**snippet**](https://developers.google.com/youtube/v3/docs/comments#snippet)": {  
 "**[authorDisplayName](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.authorDisplayName)**": **string**,  
 "**[authorProfileImageUrl](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.authorProfileImageUrl)**": **string**,  
 "**[authorChannelUrl](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.authorChannelUrl)**": **string**,  
 "**[authorChannelId](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.authorChannelId)**": {  
 "[**value**](https://developers.google.com/youtube/v3/docs/comments#snippet.authorChannelId.value)": **string**  
 },  
 "**[channelId](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.channelId)**": **string**,  
 "**[videoId](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.videoId)**": **string**,  
 "**[textDisplay](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.textDisplay)**": **string**,  
 "**[textOriginal](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.textOriginal)**": **string**,  
 "**[parentId](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.parentId)**": **string**,  
 "**[canRate](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.canRate)**": **boolean**,  
 "**[viewerRating](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.viewerRating)**": **string**,  
 "**[likeCount](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.likeCount)**": **unsigned integer**,  
 "**[moderationStatus](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.moderationStatus)**": **string**,  
 "**[publishedAt](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.publishedAt)**": **datetime**,  
 "**[updatedAt](https://developers.google.com/youtube/v3/docs/comments" \l "snippet.updatedAt)**": **datetime**  
 }  
}

We can index into the response as follows:

comment = response['items']['snippet']['topLevelComment']['snippet']['textDisplay']

Putting it all together, we can use the code snippet below to get the data points of interest.

If you’re interested in additional data points, such as the time at which a comment was updated, you can write something like:

published\_at = item['snippet']['topLevelComment']['snippet']['updatedAt']

The other value of interest for the **commentThreads** resource was the **nextPageToken**. Each time we submit a request, we get **maxResults**number of comments in the **items** list. The maximum number of results we can obtain is limited between 1 and 100. Thus, if a video has more than 100 comments, we’ll need to make an API call several times. The **nextPageToken** helps us start directly on the next page of comments instead of starting from the beginning again. We just need to modify our API call a bit:

response = service.commentThreads().list(  
 part='snippet',  
 maxResults=100,  
 textFormat='plainText',  
 order='time',  
 videoId='ioNng23DkIM',  
 pageToken=response['nextPageToken']  
).execute()

Note that we don’t need a **nextPageToken** for our very first service call. Instead, we use the **nextPageToken** obtained from the current JSON response for our next call to the service object.

Putting It All Together

The function below will help us get comments off a YouTube video:

Feel free to change the function as you see fit! After importing the necessary libraries (#1), I changed the parameters of the function to include an extra variable, csv\_filename (#2). Lists to hold features of interest, code to index for those data points, and code to save the data points to lists are outlined in #3, #5, and #6. I then saved the desired features of each item in the JSON response line-by-line to the csv file (#7). After we check every item in the JSON response, we check if there’s a nextPageToken (#8). If not, we’ll return our data points of interest in dictionary form (#9).

Next Steps

There is a lot more we can do to make this program more modular. For example, instead of hard-coding lists for each feature (#2, #5), we can write a function that takes in a list of keywords and returns a dictionary containing the relevant information for each given keyword. We can also write a dictionary that maps long, involved indexing such as the one for published\_at to a shorthand. For example:

shorthand = {  
 'updated\_at' : item['snippet']['topLevelComment']['snippet']['updatedAt']  
}

This will involve some work the first time around to simplify things down the line. Fortunately, these functions (and more) are already available in the wrapper library [youtube-data-api](https://github.com/mabrownnyu/youtube-data-api).

However, if you’d like to just collect comments out-of-the-box, my repo contains instructions on how to run the provided script [get\_comments\_of\_video\_id.py](https://github.com/XWilliamY/custom_yt_comments_dataset).

Note that Google does impose a daily quota on the number of API calls you can make. This quota is set at around [10 thousand units per day](https://developers.google.com/youtube/v3/getting-started), which becomes more or less 250,000 comments I can collect in one day. To address these limitations, I created two API keys to collect more comments.