

# Using Python to Extract Data from YouTube

# What We Will Need

- Google API's Python client package and unicode package
- Authentication information for Google APIs

# Install Package

- We will install and use a python package provided for Google API-Related product/service development.

*\$ pip install --upgrade google-api-python-client*

reference: <https://developers.google.com/api-client-library/python/>

- Another package we install is unicode, a package that handles unicode-ascii translation.

*\$ pip install unicode*

reference: <https://pypi.python.org/pypi/Unicode>

# Google APIs Set-up

Google APIs ([https://developers.google.com/api-client-library/python/start/get\\_started#setup](https://developers.google.com/api-client-library/python/start/get_started#setup))

If you have never created a Google APIs Console project, read the Managing Projects page and create a project in the Google Developers Console.

<https://console.developers.google.com/apis/library>

For example, I created a project named *SocialMedaiAnalytics* and enabled YouTube Data API v3.

See the instructions in the next slides.

---

New Project

Project name ⓘ

SocialMediaAnalytics

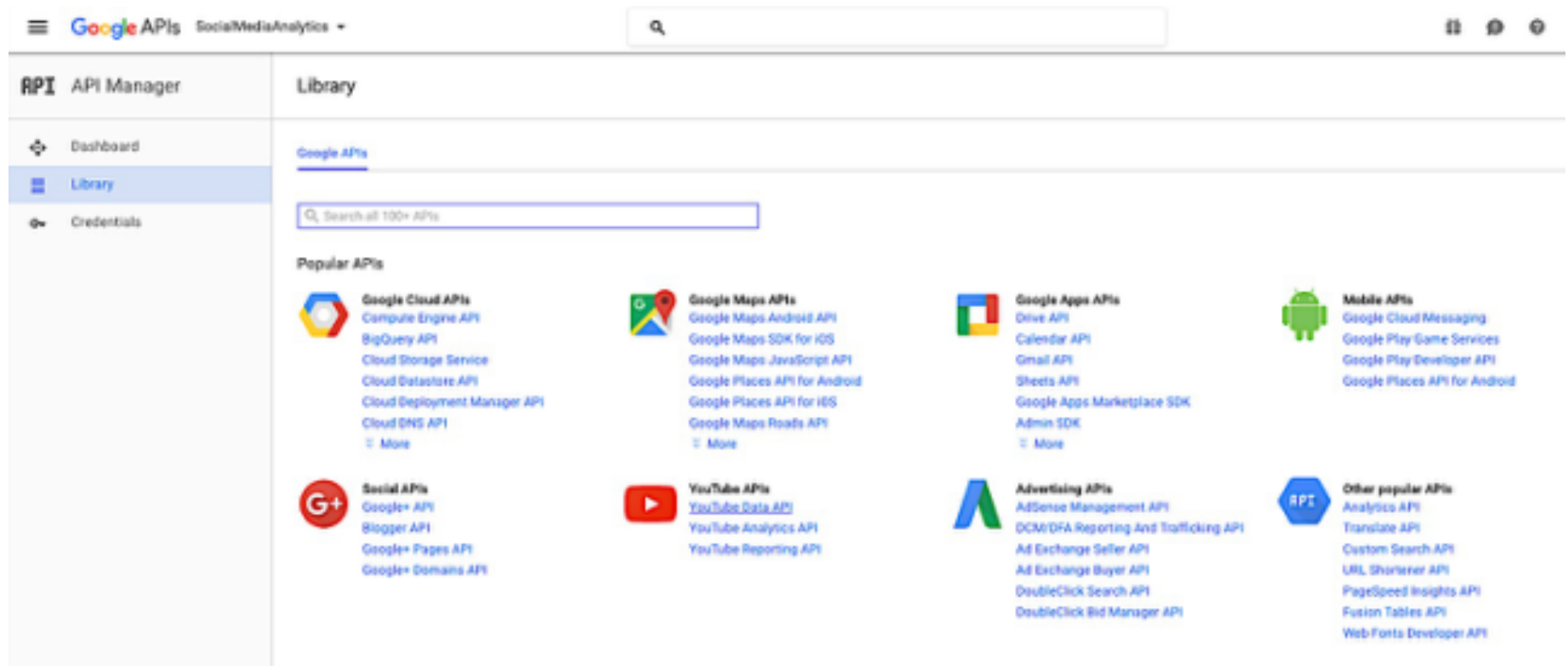
Your project ID will be learned-balm-135808 ⓘ Edit

Show advanced options...

Create Cancel

# Google APIs Set-up

Created project - SocialMediaAnalytics



# Google APIs Set-up

Enable 'YouTube Data API v3'

The screenshot shows the Google API Manager interface. At the top, there's a header with the Google APIs logo, a search bar, and a user profile icon. Below the header, the left sidebar contains the 'API Manager' title and a navigation menu with 'Dashboard', 'Library', and 'Credentials'. The main content area is titled 'YouTube Data API v3' with an 'ENABLE' button. It includes sections for 'About this API', 'Using credentials with this API', and 'Accessing user data with OAuth 2.0'. The 'Using credentials with this API' section contains a diagram showing the flow from 'Your application' to 'API key' to 'Google service'. The 'Accessing user data with OAuth 2.0' section contains a diagram showing the flow from 'Your app' to 'User consent' to 'User data'.

**API Manager**

YouTube Data API v3 **ENABLE**

**About this API** [Documentation](#) [Try this API in APIS Explorer](#)

The YouTube Data API v3 is an API that provides access to YouTube data, such as videos, playlists, and channels.

**Using credentials with this API**

**Using an API key**  
To use this API you need an API key. An API key identifies your project to check quotas and access. Go to the [Credentials](#) page to get an API key. You'll need a key for each platform, such as Web, Android, and iOS. [Learn more](#)

**Accessing user data with OAuth 2.0**  
You can access user data with this API. On the [Credentials](#) page, create an OAuth 2.0 client ID. A client ID requests user consent so that your app can access user data. Include that client ID when making your API call to Google. [Learn more](#)

**Diagram 1: Using an API key**

Your application → API key → Google service

**Diagram 2: Accessing user data with OAuth 2.0**

Your app → User consent → User data

# Google APIs Set-up

After YouTube Data API 3 is enabled, you are asked to create credentials for this API use.

Click the 'Create Credentials' button.



# Google APIs Set-up

You can configure credentials based on your purpose.

I chose the options as follows:

The screenshot shows the Google API Manager interface. The left sidebar has 'API Manager' and 'Credentials' tabs, with 'Credentials' selected. The main area is titled 'Add credentials to your project'. Step 1 is 'Find out what kind of credentials you need'. It includes instructions, a dropdown for 'Which API are you using?' (set to 'YouTube Data API v3'), a dropdown for 'Where will you be calling the API from?' (set to 'Other UI (e.g. Windows, GUI tool)'), and radio buttons for 'What data will you be accessing?' (with 'Public data' selected). A 'What credentials do I need?' button is at the bottom.

The screenshot shows the same Google API Manager interface, but at step 2: 'Create an API key'. It includes a warning about keeping the key secret, a text input for 'Name' (containing 'Server key 1'), and a section for 'Accept requests from these server IP addresses' with an 'IP address' input field. A 'Create API key' button is at the bottom.



# Google APIs Set-up

You have now the API key for your application. It looks like this:

YIzaSyCsABEu4ffovhN9WTK9mqMqewhPmO0LuRz

## Sample Code

The script, ***youtube\_search.py***, will retrieve YouTube content with a keyword it receives as an argument and show the results. The code will store video results - *title, videoid, viewCount, likeCount, dislikeCount, commentCount, and favoriteCount* - in ***video\_result.csv*** file in the same directory.

Edit the script to add your API key at the line “DEVELOPER\_KEY = ”.

# youtube\_search\_keyword.py

Put your API key in the variable of DEVELOPER\_KEY.

Run the script using command line (console) or through Spyder 'Run configuration':

In console

```
$ python ./youtube_search.py --q olympics
```

In Spyder

Set argument as '--q olympics'