# YT_data_extract

August 16, 2025

```python
[ ]: !pip install google-api-python-client
```

```python
[1]: from googleapiclient.discovery import build

     api_key = "AIzaSyA88mPtGXLzCKPDFxhIV4AIeizCQFypl9w"
     username = "8bitBinks69"   # Replace with the actual handle (without @)

     youtube = build("youtube", "v3", developerKey=api_key)
     response = youtube.search().list(q=f"@{username}", part="snippet",
       ↪type="channel").execute()
     channel_id = response["items"][0]["snippet"]["channelId"]
     print("Channel ID:", channel_id)
```

    Channel ID: UCSQ7hHnWkDMyDBEp-h1Pqbg

```python
[2]: # Use the channel_id you just got
     response = youtube.channels().list(
         part="contentDetails",
         id=channel_id
     ).execute()

     uploads_playlist_id =
       ↪response["items"][0]["contentDetails"]["relatedPlaylists"]["uploads"]
     print("Uploads Playlist ID:", uploads_playlist_id)
```

    Uploads Playlist ID: UUSQ7hHnWkDMyDBEp-h1Pqbg

```python
[3]: video_ids = []
     next_page_token = None

     while True:
         playlist_response = youtube.playlistItems().list(
             part="contentDetails",
             playlistId=uploads_playlist_id,
             maxResults=50,
             pageToken=next_page_token
         ).execute()
```

```
        for item in playlist_response['items']:
            video_ids.append(item['contentDetails']['videoId'])

        next_page_token = playlist_response.get('nextPageToken')
        if not next_page_token:
            break

ttl_vids = len(video_ids)
print(f" Total videos fetched: {ttl_vids}")
print("Sample Video IDs:", video_ids[:5])
```

```
 Total videos fetched: 2388
Sample Video IDs: ['itPg4y2jVw4', '6zOvXvXGY70', '-nSSEuSIZPc', 'wSDAJsB-JVw',
'M701L-Arx_8']
```

```
[ ]: !pip install isodate
```

```
[7]: import pandas as pd
import time
from datetime import datetime
import isodate
import pytz  # For timezone conversion

all_video_data = []

# Loop through video_ids in chunks of 50 (API limit)
for i in range(0, len(video_ids), 50):
    batch = video_ids[i:i + 50]
    response = youtube.videos().list(
        part="snippet,contentDetails,statistics,liveStreamingDetails,status",
        id=",".join(batch)
    ).execute()

    for item in response['items']:
        snippet = item['snippet']
        stats = item.get('statistics', {})
        content = item['contentDetails']
        status = item.get('status', {})

        # Convert publishedAt (UTC) to IST
        published_at_utc = datetime.strptime(snippet.get('publishedAt'),␣
  ↪"%Y-%m-%dT%H:%M:%SZ")
        published_at_utc = published_at_utc.replace(tzinfo=pytz.utc)
        ist = pytz.timezone("Asia/Kolkata")
        published_ist = published_at_utc.astimezone(ist)

        # Determine video type
```

```python
        if 'liveStreamingDetails' in item:
            video_type = "livestream"
        elif snippet.get('liveBroadcastContent') == 'upcoming':
            video_type = "upcoming livestream"
        else:
            video_type = "video"

        # Convert ISO 8601 duration to hh:mm:ss
        try:
            duration_seconds = isodate.parse_duration(content.get('duration')).
↪total_seconds()
            hours = int(duration_seconds // 3600)
            minutes = int((duration_seconds % 3600) // 60)
            seconds = int(duration_seconds % 60)
            formatted_duration = f"{hours:02}:{minutes:02}:{seconds:02}"
        except Exception:
            formatted_duration = ""

        # Check if video is a short
        is_short = duration_seconds <= 60 if 'duration_seconds' in locals()␣
↪else False

        # Get privacy status
        privacy_status = status.get('privacyStatus', '')

        video_data = {
            'videoId': "'" + item['id'],  # Prevent Excel from treating it as␣
↪formula
            'title': snippet.get('title'),
            'publishedAt': published_ist.strftime("%Y-%m-%d %H:%M:%S"),
            'publishedDate': published_ist.date().isoformat(),
            'publishedTime': published_ist.time().isoformat(),
            'duration': formatted_duration,
            'viewCount': stats.get('viewCount'),
            'likeCount': stats.get('likeCount'),
            'commentCount': stats.get('commentCount'),
            'videoType': video_type,
            'isShort': is_short,
            'privacyStatus': privacy_status
        }

        all_video_data.append(video_data)

    time.sleep(1)  # prevent rate limiting

# Convert to DataFrame
df = pd.DataFrame(all_video_data)
```

```
print(df.head())
```

```
        videoId                                              title  \
0  'itPg4y2jVw4  UNREAL UPDATE THEEK HAI BUT UNREAL TEAM KA KYA…
1  '6zOvXvXGY70                     ROAD TO IMMO 3 ?? Aajao Dewds !
2  '-nSSEuSIZPc  5 MAN BEGINS | SMARTEST VALORANT 5 MAN | AAJAO :(
3  'wSDAJsB-JVw  8 Mins of Binks Getting Frustrated By His Team…
4  'M701L-Arx_8  C9 vs EG | MIBR VS LEV | VCT NA WATCHPARTY | #…


            publishedAt publishedDate publishedTime  duration  viewCount  \
0  2025-07-31 01:27:42    2025-07-31      01:27:42  04:14:48      84176
1  2025-07-30 02:46:24    2025-07-30      02:46:24  04:56:55      92751
2  2025-07-29 01:50:21    2025-07-29      01:50:21  04:02:05      86557
3  2025-07-28 11:00:34    2025-07-28      11:00:34  00:08:08      53631
4  2025-07-28 07:10:21    2025-07-28      07:10:21  04:27:03      29681


   likeCount  commentCount   videoType  isShort privacyStatus
0       5922             7  livestream    False        public
1       5951            14  livestream    False        public
2       6110            12  livestream    False        public
3       5395            29       video    False        public
4       1992             5  livestream    False        public
```

```
[9]: df.to_excel(r"E:\Data Storage\Youtuber's Data\8bit_Binks69's_Data.xlsx",
     ↪sheet_name = "Channel_data", index = False)
```