import httplib2  
import os  
import sys  
  
from apiclient.discovery import build  
from apiclient.errors import HttpError  
from oauth2client.client import flow\_from\_clientsecrets  
from oauth2client.file import Storage  
from oauth2client.tools import argparser, run\_flow  
  
  
# The CLIENT\_SECRETS\_FILE variable specifies the name of a file that contains  
# the OAuth 2.0 information for this application, including its client\_id and  
# client\_secret. You can acquire an OAuth 2.0 client ID and client secret from  
# the Google API Console at  
# https://console.cloud.google.com/.  
# Please ensure that you have enabled the YouTube Data API for your project.  
# For more information about using OAuth2 to access the YouTube Data API, see:  
#   https://developers.google.com/youtube/v3/guides/authentication  
# For more information about the client\_secrets.json file format, see:  
#   https://developers.google.com/api-client-library/python/guide/aaa\_client\_secrets  
CLIENT\_SECRETS\_FILE = "client\_secrets.json"  
  
# This OAuth 2.0 access scope allows for full read/write access to the  
# authenticated user's account.  
YOUTUBE\_READ\_WRITE\_SCOPE = "https://www.googleapis.com/auth/youtube"  
YOUTUBE\_API\_SERVICE\_NAME = "youtube"  
YOUTUBE\_API\_VERSION = "v3"  
  
# This variable defines a message to display if the CLIENT\_SECRETS\_FILE is  
# missing.  
MISSING\_CLIENT\_SECRETS\_MESSAGE = """  
WARNING: Please configure OAuth 2.0  
  
To make this sample run you will need to populate the client\_secrets.json file  
found at:  
  
   %s  
  
with information from the API Console  
https://console.cloud.google.com/  
  
For more information about the client\_secrets.json file format, please visit:  
https://developers.google.com/api-client-library/python/guide/aaa\_client\_secrets  
""" % os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_),  
                                   CLIENT\_SECRETS\_FILE))  
  
def get\_authenticated\_service(args):  
  flow = flow\_from\_clientsecrets(CLIENT\_SECRETS\_FILE,  
    scope=YOUTUBE\_READ\_WRITE\_SCOPE,  
    message=MISSING\_CLIENT\_SECRETS\_MESSAGE)  
  
  storage = Storage("%s-oauth2.json" % sys.argv[0])  
  credentials = storage.get()  
  
  if credentials is None or credentials.invalid:  
    credentials = run\_flow(flow, storage, args)  
  
  return build(YOUTUBE\_API\_SERVICE\_NAME, YOUTUBE\_API\_VERSION,  
    http=credentials.authorize(httplib2.Http()))  
  
# This method calls the API's youtube.activities.insert method to post the  
# channel bulletin.  
def post\_bulletin(youtube, args):  
  body = dict(  
    snippet=dict(  
      description=args.message  
    )  
  )  
  
  if args.video\_id:  
    body["contentDetails"] = dict(  
      bulletin=dict(  
        resourceId=dict(  
          kind="youtube#video",  
          videoId=args.video\_id  
        )  
      )  
    )  
  
  if args.playlist\_id:  
    body["contentDetails"] = dict(  
      bulletin=dict(  
        resourceId=dict(  
          kind="youtube#playlist",  
          playlistId=args.playlist\_id  
        )  
      )  
    )  
  
  youtube.activities().insert(  
    part=",".join(body.keys()),  
    body=body  
  ).execute()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  argparser.add\_argument("--message", required=True,  
    help="Text of message to post.")  
  argparser.add\_argument("--video-id",  
    help="Optional ID of video to post.")  
  argparser.add\_argument("--playlist-id",  
    help="Optional ID of playlist to post.")  
  args = argparser.parse\_args()  
  
  # You can post a message with or without an accompanying video or playlist.  
  # However, you can't post a video and a playlist at the same time.  
  if args.video\_id and args.playlist\_id:  
    exit("You cannot post a video and a playlist at the same time.")  
  
  youtube = get\_authenticated\_service(args)  
  try:  
    post\_bulletin(youtube, args)  
  except HttpError, e:  
    print "An HTTP error %d occurred:\n%s" % (e.resp.status, e.content)  
  else:  
    print "The bulletin was posted to your channel."