```
#!/usr/bin/python
"""Sample code showing how to access the Google Flu Trends API.
"""

import csv
import datetime
import sys
import time

from apiclient.discovery import build

# ------ Insert your API key in the string below. -------
API_KEY = ''

SERVER = 'https://www.googleapis.com'
API_VERSION = 'v1beta'
DISCOVERY_URL_SUFFIX = '/discovery/v1/apis/trends/' + API_VERSION + '/
rest'
DISCOVERY_URL = SERVER + DISCOVERY_URL_SUFFIX

MAX_QUERIES = 30


def DateToISOString(datestring):
  """Convert date from (eg) 'Jul 04 2004' to '2004-07-11'.

  Args:
    datestring: A date in the format 'Jul 11 2004', 'Jul 2004', or
'2004'

  Returns:
    The same date in the format '2004-11-04'

  Raises:
     ValueError: when date doesn't match one of the three expected
formats.
  """

  try:
    new_date = datetime.datetime.strptime(datestring, '%b %d %Y')
  except ValueError:
    try:
      new_date = datetime.datetime.strptime(datestring, '%b %Y')
    except ValueError:
      try:
        new_date = datetime.datetime.strptime(datestring, '%Y')
      except:
        raise ValueError("Date doesn't match any of '%b %d %Y', '%b
%Y', '%Y'.")
```

```
  return new_date.strftime('%Y-%m-%d')


def GetQueryVolumes(queries, start_date, end_date,
                    geo='US', geo_level='country', frequency='week'):
  """Extract query volumes from Flu Trends API.

  Args:
    queries: A list of all queries to use.
    start_date: Start date for timelines, in form YYYY-MM-DD.
    end_date: End date for timelines, in form YYYY-MM-DD.
    geo: The code for the geography of interest which can be either
country
         (eg "US"), region (eg "US-NY") or DMA (eg "501").
    geo_level: The granularity for the geo limitation. Can be
"country",
               "region", or "dma"
    frequency: The time resolution at which to pull queries. One of
"day",
               "week", "month", "year".

  Returns:
    A list of lists (one row per date) that can be output by
csv.writer.

  Raises:
    ValueError: when geo_level is not one of "country", "region" or
"dma".
  """

  if not API_KEY:
    raise ValueError('API_KEY not set.')

  service = build('trends', API_VERSION,
                  developerKey=API_KEY,
                  discoveryServiceUrl=DISCOVERY_URL)

  dat = {}

  # Note that the API only allows querying 30 queries in one request.
In
  # the event that we want to use more queries than that, we need to
break
  # our request up into batches of 30.
  batch_intervals = range(0, len(queries), MAX_QUERIES)

  for batch_start in batch_intervals:
    batch_end = min(batch_start + MAX_QUERIES, len(queries))
    query_batch = queries[batch_start:batch_end]
```

```
    # Make API query
    if geo_level == 'country':
      # Country format is ISO-3166-2 (2-letters), e.g. 'US'
      req = service.getTimelinesForHealth(terms=query_batch,
                                          time_startDate=start_date,
                                          time_endDate=end_date,

timelineResolution=frequency,
                                          geoRestriction_country=geo)
    elif geo_level == 'dma':
      # See https://support.google.com/richmedia/answer/2745487
      req = service.getTimelinesForHealth(terms=query_batch,
                                          time_startDate=start_date,
                                          time_endDate=end_date,

timelineResolution=frequency,
                                          geoRestriction_dma=geo)
    elif geo_level == 'region':
      # Region format is ISO-3166-2 (4-letters), e.g. 'US-NY' (see
more examples
      # here: en.wikipedia.org/wiki/ISO_3166-2:US)
      req = service.getTimelinesForHealth(terms=query_batch,
                                          time_startDate=start_date,
                                          time_endDate=end_date,

timelineResolution=frequency,
                                          geoRestriction_region=geo)
    else:
      raise ValueError("geo_type must be one of 'country', 'region' or
'dma'")

    res = req.execute()

    # Sleep for 1 second so as to avoid hittting rate limiting.
    time.sleep(1)

    # Convert the data from the API into a dictionary of the form
    # {(query, date): count, ...}
    res_dict = {(line[u'term'], DateToISOString(point[u'date'])):
                point[u'value']
                for line in res[u'lines']
                for point in line[u'points']}

    # Update the global results dictionary with this batch's results.
    dat.update(res_dict)

  # Make the list of lists that will be the output of the function
  res = [['date'] + queries]
  for date in sorted(list(set([x[1] for x in dat]))):
    vals = [dat.get((term, date), 0) for term in queries]
```

```
    res.append([date] + vals)

  return res


def main():

  # Examples of calling the GetQueryVolumes function for different geo
  # levels and time resolutions.
  us_weekly = GetQueryVolumes(['flu', 'cough'],
                              start_date='2011-01-01',
                              end_date='2015-01-01',
                              geo='US',
                              geo_level='country',
                              frequency='week')

  ma_region_daily = GetQueryVolumes(['flu', 'cough'],
                                    start_date='2011-01-01',
                                    end_date='2015-01-01',
                                    geo='US-MA',
                                    geo_level='region',
                                    frequency='day')

  boston_dma_monthly = GetQueryVolumes(['flu', 'cough'],
                                       start_date='2011-01-01',
                                       end_date='2015-01-01',
                                       geo='506',
                                       geo_level='dma',
                                       frequency='month')

  # Example of writing one of these files out as a CSV file to STDOUT.
  outwriter = csv.writer(sys.stdout)
  for row in us_weekly:
    outwriter.writerow(row)

if __name__ == '__main__':
  main()
```