```
'''
Lab 3

Trang Van
CIS 41B

Back End - Access the web with a URL to scrape data and create 3 files:
pickle, JSON, and a SQL database

'''


import requests
from bs4 import BeautifulSoup
import re
import pickle
import json
import sqlite3



'''
Class to access web and create dictionary from extracted data

'''
class extractData:
    def __init__(self, url):
        self._links = []
        self._movieTitles = []
        self._releaseDates = []
        self._genres = []
        self._ratings = []

        page = requests.get(url)

        self._soup = BeautifulSoup(page.content, "lxml")
        self._soup.prettify().encode("utf8")

    '''
    Uses main page to get names of movies
    '''
    def getMovieTitles(self):
        temp_titles = []
        for  elem in self._soup.find_all('h3'):  #multiple tag names by
giving find_all a list

            if 'Review' in elem.get_text():
                temp_titles.append(elem.text[:-7].strip('\n'))
            else:
                temp_titles.append(elem.text.strip('\n'))

        self._movieTitles = temp_titles[15:-5]

    '''
```

```
    From main page, gets links of each movie listed
    '''
    def getMovieLinks(self):
        for link in self._soup.find_all('a'):
            try:
                if re.search('moviereviews', link['href']):
                    self._links.append(link['href'])
            except KeyError:
                pass

    '''
    Follows links to get information on each movie's ratings, release
date, and genre

    '''
    def getSubPageData(self):
        for link in self._links:
            #Create new page to parse with each link

            pageSub = requests.get(link)
            soupSub = BeautifulSoup(pageSub.content, "lxml")

            # Find ratings of movie
            for div in soupSub.find_all('div'):
                try:
                    if 'data-star-rating' in div.attrs:
                        self._ratings.append(div['data-star-rating'])
                except KeyError:
                    pass

            # Get release date of movie
            for li in soupSub.find_all('li'):
                try:
                    if 'movie-details__release-date' in li['class']:
                        self._releaseDates.append(li.get_text().strip('\n'
))
                except KeyError:
                    pass

            #Get genre(s) of movie
            for info in soupSub.find_all('ul'):

                try:
                    if 'movie-details__detail' in info['class']:
                        # get genres from ul tag
                        genre_buf = re.findall(r'\S+', info.get_text())

                        if len(genre_buf) == 11:
                            self._genres.append(genre_buf[7:8])
                        elif len(genre_buf) == 12:
                            self._genres.append(['NONE'])
                        elif len(genre_buf) == 13:
                            self._genres.append(genre_buf[9:10])
                        elif len(genre_buf) == 14:
```

```
                            self._genres.append(genre_buf[9:11])
                        elif len(genre_buf) == 15:
                            self._genres.append(genre_buf[9:12])
                        elif len(genre_buf) == 16:
                            self._genres.append(genre_buf[9:11])
                except KeyError:
                    pass


    '''
    Zips lists of data into dictionary and returns dictionary with movie
name as key

    '''
    def createDataDictionary(self):
        return dict(zip(self._movieTitles,
zip(self._ratings,self._releaseDates, self._genres)))

# END OF CLASS

'''
Write Pickle File with dictionary returned from extractData class
'''

def makePickleFile(data_diction):
    pickle.dump(data_diction, open('pickledData.pickle', "wb"))

'''
Write JSON File with dictionary returned from extractData class
'''

def makeJSONFile(data_diction):
    with open('data.json', 'w') as fh:
        json.dump(data_diction, fh, indent =3)


'''
Create database with SQLite and json file

'''
def createDatabase():
    with open('data.json', 'r') as fh:
        data = json.load(fh)

    conn = sqlite3.connect('movieInfo.db')

    cur = conn.cursor()
    cur.execute("DROP TABLE IF EXISTS Genres")
    cur.execute('''CREATE TABLE Genres(
                        id INTEGER NOT NULL PRIMARY KEY,
                        genre TEXT UNIQUE ON CONFLICT IGNORE)''')

    cur.execute("DROP TABLE IF EXISTS MovieDB")
    cur.execute('''CREATE TABLE MovieDB(
```

```
                        id INTEGER NOT NULL PRIMARY KEY UNIQUE,
                        movie TEXT,
                        rating REAL,
                        genre_id INTEGER,
                        date TEXT)''')

    id_prim = 1
    for key,val in data.items() :
        cur.execute('''INSERT INTO Genres (genre) VALUES (?)''',
(val[2][0], ))
        cur.execute('SELECT id FROM Genres WHERE genre = ? ',
(val[2][0], ))
        genre_id = cur.fetchone()[0]

        cur.execute('''INSERT INTO MovieDB
            (id, movie, rating, genre_id, date)
            VALUES ( ?, ?, ?, ?, ? )''', (id_prim, key, val[0], genre_id,
val[1]) )
        id_prim += 1
    conn.commit()

    '''
    # Test Database
    cur.execute("SELECT * from MovieDB")
    print(cur.fetchall())
    cur.execute("SELECT * from Genres")
    print(cur.fetchall())

    '''
'''
Used to create database by access web with extratData object
'''

def main():

    '''
    ed = extractData('https://www.fandango.com/movie-reviews')
    ed.getMovieTitles()
    ed.getMovieLinks()
    ed.getSubPageData()

    date_dict = ed.createDataDictionary()


    makePickleFile(date_dict)
    makeJSONFile(date_dict)
    '''
    createDatabase()


main()
```