Using the ERIC API for Research Topics

This Jupyter Notebook demonstrates how to use the ERIC API to research topics. This example uses Python and the Pandas library.

The ERIC API supports any programming language that supports making HTTPS requests. Other popular choices include JavaScript, C#, and Java.

Import required Python libraries

We start by importing the Python libraries used in this notebook.

In [ ]:
```
import numpy as np
import pandas as pd
import requests
import json
import time
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

Next we create a function to make ERIC API requests

The following function (`getEricRecords`) calls the ERIC API when invoked. It takes five parameters:

- search - the search criteria. See the ERIC API docs (https://d170fjyqpmf5gt.cloudfront.net/index.html#/default/get_eric_) for details.
- format - the response format. Valid values are `xml` (default), `json`, and `csv`.
- start - this parameter supports pagination by specifying the starting record index for the returned record set. The default is 0.
- rows - by default, the ERIC API returns 20 records at a time. This parameter can be set to a value between 20 and 200.
- fields - specifies the fields to include in the returned records. See the ERIC API docs (https://d170fjyqpmf5gt.cloudfront.net/index.html#/default/get_eric_) for details.

In [ ]:
```
def getEricRecords(search, fields = None, start=0, rows=200):
    url = 'https://api.ies.ed.gov/eric/?'
    url = url + 'search=' + search + '&rows=' + str(rows) + '&format=json&start=' + str(start)
    if(fields):
        url = url + '&fields=' + ', '.join(fields)
    responseJson = requests.get(url).json()
    return pd.DataFrame(responseJson)
```

Get Number of Records for Search

This utility function returns the number of records returned by the ERIC API for a given search.

In [ ]:
```
def getRecordCount(search):
    dataFrame = getEricRecords(search)
    totalRecords = dataFrame.loc['numFound'][0]
    print('Search', search, 'returned', '{:,}'.format(totalRecords), 'records')
    return totalRecords
```

Create function to get all records for an ERIC API search

The following function `getAllEricRecords` calls the `getEricRecords` function as many times as needed to fetch all records in an ERIC API Search. If a set of return fields are not specified, the Frequently used fields (https://d170fjyqpmf5gt.cloudfront.net/index.html#/default/get_eric_) will be returned.

The `cleanElementsUsingList` function cleans the resulting dataframe from showing empty lists in some fields where the ERIC API returns an empty list.

In [ ]:

```python
def cleanElementsUsingList(x):
    if(not isinstance(x, list)):
        return x
    if(not x or (len(x) == 1 and x[0] == '')):
        return None
    return ', '.join(x)


def getAllEricRecords(search, fields = None, cleanElements = True):
    startTime = time.time()
    nextFirstRecord = 0
    numRecordsReturnedEachApiCall = 200
    totalRecords = getRecordCount(search)
    if(totalRecords == 0):
        print ('Search', search, 'has no results')
        return []

    while(nextFirstRecord < totalRecords):
        dataFrame = getEricRecords(search, fields, nextFirstRecord)
        if(nextFirstRecord == 0):
            records = pd.DataFrame(dataFrame.loc['docs'][0])
        else:
            records = pd.concat([records, pd.DataFrame(dataFrame.loc['docs'][0])], sort=False, ignore_index=T
        nextFirstRecord += numRecordsReturnedEachApiCall
    print('took', '{:,.1f}'.format(time.time() - startTime), 'seconds')
    return records.applymap(cleanElementsUsingList) if cleanElements else records
```

Example of making ERIC API requests

Some example search values:

- autism
- title:"growth Mindset" AND subject:"Music Education"
- subject:"Blended Learning" AND source:"US Department of Education"
- source:"What Works Clearinghouse" AND subject:Mathematics
- author:"Gersten, Russell"

Example1: Start with the search we did when we looked at the ERIC API docs

In [ ]:

```python
search = 'subject:autism AND subject:”teaching methods” AND publicationdateyear:2019'
records = getAllEricRecords(search)
records
```

This shows what columns are returned and the frequency of attributes in each record

In [ ]:

```python
records.info()
```

We can rearrange the column order and show a subset of the columns

In [ ]:

```python
records[['id', 'title', 'author','publicationdateyear', 'description']].head(20)
```

Add all ERIC fields to be returned

In [ ]:

```python
allEricFields = ['id', 'title', 'author', 'source', 'publicationdateyear', 'description',
                      'subject', 'peerreviewed', 'abstractor', 'audience', 'authorxlink',
                      'e_datemodified', 'e_fulltextauth', 'e_yearadded', 'educationlevel',
                      'identifiersgeo', 'identifierslaw', 'identifierstest', 'iescited',
                      'iesfunded', 'iesgrantcontractnum', 'iesgrantcontractnumxlink',
                      'ieslinkpublication', 'ieslinkwwcreviewguide', 'ieswwcreviewed',
                      'institution', 'isbn', 'issn', 'language', 'publicationtype',
                      'publisher', 'sourceid', 'sponsor', 'url']
print('There are', len(allEricFields), 'possible fields in the ERIC API response')
```

In [ ]:

```python
search = 'subject:autism AND subject:”teaching methods” AND publicationdateyear:2019'

records = getAllEricRecords(search, allEricFields)
```

Inspect the first 7 columns

In [ ]:
```
records.head()
```

In [ ]:
```
records.info()
```

Write these records to an Excel file

In [ ]:
```
records.to_excel('ERIC records for autism and teaching methods in 2019.xlsx')
```

Example 2: When was "hazing" researched and when was an author published?

Find all records about "hazing"

In [ ]:
```
records = getAllEricRecords('hazing', allEricFields)
```

In [ ]:
```
records.info()
```

Group and count the number of ERIC records by year and graph the results

In [ ]:
```
groupedData = records[['id', 'publicationdateyear']].groupby('publicationdateyear').count()
ax = groupedData.plot(title='Publications by Year', legend=False)
ax.set(xlabel = 'Year', ylabel = 'Count')
plt.show()
```

When did Russell Gersten publish articles?

Russell has authored many studies for IES, including acting as the principal investigator or the panel chair for many What Works Clearinghouse Practice Guides and randomized control trials funded by the RELs.

In [ ]:
```
gerstenRecords = getAllEricRecords('author:"Gersten, Russell"')
data = gerstenRecords[['id', 'publicationdateyear']].groupby('publicationdateyear').count()
ax = data.plot(title='Publications by Year', legend=False)
ax.set(xlabel = 'Year', ylabel='Count')
ax.set_yticks(range(1,10))
plt.show()
```

Example 3: Tracking changes in ERIC over time

In this example, we save an ERIC API query to a file. We'll then remove some records to simulate that this query was done a couple of years ago. Finally, we will rerun the query and automatically identify the new or changed records.

Some examples of how tracking changes may be used include:

- researchers tracking the newest research for input to their projects
- professors and other experts wanting to stay abreast of the research in their field
- third party providers and librarians wanting to update their repositories or catalogs with new records

Start by running a search

In [ ]:
```
requiredFields = ['id', 'title', 'publicationdateyear', 'e_datemodified']
records = getAllEricRecords('source:"What Works Clearinghouse" AND subject:Mathematics',
                            requiredFields, False)
records[requiredFields].head()
```

Then save results to a CSV file

In [ ]:
```
records.sort_values(['publicationdateyear']).to_csv('ERIC What Works Clearinghouse Records (Up through 2017).
```

Open the file and delete or change some of the recent entries to simulate having run this query in the past

Then load this file as the `previous records` data frame

In [ ]:
```
previousRecords = pd.read_csv('ERIC What Works Clearinghouse Records (Up through 2017).csv')
previousRecords.shape
```

Rerun the query (in the present)

In [ ]:
```
records = getAllEricRecords('source:"What Works Clearinghouse" AND subject:Mathematics',
                            requiredFields, False)
```

Compare the two sets of records to find new and updated records

In [ ]:
```
merged = records.merge(previousRecords, how='outer', indicator=True)
merged.loc[merged['_merge'] == 'left_only', ['id', 'publicationdateyear', 'title']]
```

See the previous state of changed records

In [ ]:
```
merged.loc[merged['_merge'] == 'right_only', ['e_datemodified', 'id', 'publicationdateyear', 'title']]
```

Other Data Exploration

Sort records

This is an example of sorting records. In this case, sorting by author and then title.

In [ ]:
```
records = getAllEricRecords('subject:autism AND subject:”teaching methods” AND peerreviewed:T', allEricFields
records = records.sort_values(['author', 'title'])
records.head(20)
```

Find top sources for a given query

In [ ]:
```
records['source'].value_counts().head(15)
```

Quickly get record counts for a number of different queries

In [ ]:
```
getRecordCount('subject:autism')
getRecordCount('subject:autism AND subject:”teaching methods”')
getRecordCount('subject:autism AND subject:”teaching methods” AND peerreviewed:T')
getRecordCount('subject:autism AND subject:”teaching methods” AND peerreviewed:T AND publicationdateyear:2019
getRecordCount('education')
getRecordCount('museum AND education')
getRecordCount('subject:hazing')
getRecordCount('author:"Gersten, Russell"')
count = getRecordCount('source:"What Works Clearinghouse" AND subject:Mathematics')
```

In [ ]:
```

```