JES Standards Scrape with BeautifulSoup (v2)

This script parses .sec files (which are actually XML), and produces an Excel report with cited standards by JES section.

Jupyter Notebook written by Ben Fisher on 25 November 2024
benjamin.s.fisher@usace.army.mil

Imports

The following imports are assumed to have been previously installed (for Notebook installs, use *! pip install ~*)

In [1]:
```
import os, warnings, datetime, time
from pathlib import Path
import pandas as pd
import numpy as np
import bs4 as bs
import lxml
import regex as re
```

Warning Suppression (Jupyter Notebooks only)

The following code will suppress the user warning generated by Beautiful Soup when parsing XML files with lxml: *XMLParsedAsHTMLWarning: It looks like you're parsing an XML document using an HTML parser. If this really is an HTML document (maybe it's XHTML?)...*

This is necessary **only** when running the script in a Notebook context. You can skip this line if script ran elsewhere.

In [2]:
```
warnings.filterwarnings('ignore')
```

Path Definitions

Assumptions:

- Batch processing will be assumed. Tkinter's filedialog.askdirectory() won't be used, rather, it'll be assumed that files for processing are placed in named subfolders, adjacent to this .ipynb (Notebook) file.
- Files will be .sec (internally are XML).
- Div 00 will be assumed not included.

Directories

Working directories are made relative to the 'current working directory,' which is where the Notebook (.ipynb) file is located.

In [3]:
```
parent_folder = os.getcwd()

jes_masters = parent_folder + '\\JES\\'
ufgs_masters = parent_folder + '\\UFGS\\'

jes_cleaned = parent_folder + '\\JES Cleaned\\'
ufgs_cleaned = parent_folder + '\\UFGS Cleaned\\'
```

Clean 0x81 Character from Files

The non-printing byte 0x81 (129) creates an ill-formed error. Simply opening the file as a UTF-8 string is problematic, so read the string as bytes, then replace the character and write as bytes.

Don't use BS4 for this, rather read the binary.

I believe this may be the result of copy-pasting standards titles from the the web: non-printing characters are accidentally introduced?

Define Helper Functions

In [4]:
```
def get_error_files(folder):
    find_text = b'\x81'
    replace_text = b''
    error_files = []

    for file in os.listdir(folder):
        file_path = folder + file
        with open(file_path, 'rb') as f:
            content = f.read()
        start_count = len(content)
        content = content.replace(find_text, replace_text)
        end_count = len(content)

        if not end_count == start_count:
            error_files.append(file)

    return error_files
```

In [5]:
```
def clean_file(file):
    find_text = b'\x81'
    replace_text = b''

    with open(file, 'rb') as f:
        content = f.read()

    content = content.replace(find_text, replace_text)

    with open(file, 'wb') as f:
        f.write(content)
```

In [6]:
```
def batch_clean_files(folder, file_list):
    for file in file_list:
        file_path = folder + file
        clean_file(file_path)
```

Get UFGS Errors

In [7]:
```
ufgs_error_files = get_error_files(ufgs_masters)

if len(ufgs_error_files) > 0:
    ufgs_unicode_error_report = parent_folder + '\\UFGS Unicode Byte Report ' + '{:%Y%

    df = pd.DataFrame(ufgs_error_files)
    df.index = np.arange(1, len(df) + 1)
    df.to_excel(ufgs_unicode_error_report, header=['Sections'])

ufgs_error_files
```

Out[7]:
```
['02 83 00.SEC']
```

Get JES Errors

In [8]:
```
jes_error_files = get_error_files(jes_masters)

if len(jes_error_files) > 0:
    jes_unicode_error_report = parent_folder + '\\JES Unicode Byte Report ' + '{:%Y%m%

    df = pd.DataFrame(jes_error_files)
    df.index = np.arange(1, len(df) + 1)
    df.to_excel(jes_unicode_error_report, header=['Sections'])

jes_error_files
```

Out[8]:
```
['02 83 00.sec',
 '05 12 00.SEC',
 '05 30 00.SEC',
 '05 51 33.SEC',
 '05 52 00.SEC',
 '07 60 00.SEC',
 '08 11 16.SEC',
 '08 31 00.sec',
 '08 91 00.sec',
 '10 14 00.10.SEC',
 '10 14 00.20.SEC',
 '21 30 00.SEC',
 '22 00 00.sec',
 '23 05 15.SEC',
 '23 07 00.SEC',
 '23 23 00.SEC',
 '23 30 00.SEC',
 '23 64 26.SEC',
 '23 65 00.SEC',
 '23 81 00.SEC',
 '26 11 16.00 33.SEC',
 '33 11 00.SEC']
```

Clean 0x81 Character in Files

In [9]:
```
batch_clean_files(ufgs_cleaned, ufgs_error_files)
batch_clean_files(jes_cleaned, jes_error_files)
```

Check if Error Files Cleaned

In [10]:
```
ufgs_cleaned_test = get_error_files(ufgs_cleaned)
jes_cleaned_test = get_error_files(jes_cleaned)
print(f'Check UFGS cleaned: {len(ufgs_cleaned_test)} errors')
print(f'Check JES cleaned: {len(jes_cleaned_test)} errors')
```

```
Check UFGS cleaned: 0 errors
Check JES cleaned: 0 errors
```

Read Files for Unprocessed Changes

Read through files to see if there are any \<ADD> or \<DEL> tags. We have to clean out the \<DEL> tags that poison everything, especially because examples of their inconsistent application have been found in several sections.

Helper Functions

In [11]:
```
def kill_tags(soup, tag_name):
    # This processes the SOUP not the files
    tags = soup.find_all(tag_name)
    needs_repair = True if len(tags) > 0 else False
    for tag in tags:
        tag.decompose()
    return needs_repair
```

In [12]:
```
def get_unprocessed_files(folder):
    unprocessed_files = []

    for file in os.listdir(folder):
        file_path = folder + file
        if Path(file_path).suffix.lower() == '.sec':
            try:
                with open(file_path, 'r') as doc:
                    soup = bs.BeautifulSoup(doc.read(), 'lxml')
                    if len(soup.find_all('del')) > 0:
                        unprocessed_files.append(file)
            except:
                pass

    return unprocessed_files
```

Process Cleaned Files (0x81 removed) to Find Unprocessed Files ('\<ADD> and \<DEL> tags remaining')

These next two cells are somewhat time consuming because you have to create the Soup for each file

In [13]:
```
unprocessed_ufgs = get_unprocessed_files(ufgs_cleaned)

print(f'Total files with <DEL> tags: {len(unprocessed_ufgs)}')

if len(unprocessed_ufgs) > 0:
    ufgs_unprocessed_report = parent_folder + '\\UFGS Unprocessed Report ' + '{:%Y%m%d

    df = pd.DataFrame(unprocessed_ufgs)
    df.index = np.arange(1, len(df) + 1)
    df.to_excel(ufgs_unprocessed_report, header=['Sections'])
```

```
unprocessed_ufgs
```

```
Total files with <DEL> tags: 0
[]
```

Out[13]:

In [14]:

```
unprocessed_jes = get_unprocessed_files(jes_cleaned)

print(f'Total files with <DEL> tags: {len(unprocessed_jes)}')

if len(unprocessed_jes) > 0:
    jes_unprocessed_report = parent_folder + '\\JES Unprocessed Report ' + '{:%Y%m%d %

    df = pd.DataFrame(unprocessed_jes)
    df.index = np.arange(1, len(df) + 1)
    df.to_excel(jes_unprocessed_report, header=['Sections'])

unprocessed_jes
```

```
Total files with <DEL> tags: 106
```

Out[14]:
```
['01 11 00.00 10.sec',
 '01 11 30.00 25.sec',
 '01 32 01.00 10.sec',
 '01 33 00.sec',
 '01 33 16.sec',
 '01 33 29.sec',
 '01 35 26.sec',
 '01 42 00.sec',
 '01 45 00.00 10.sec',
 '01 45 00.15 10.sec',
 '01 45 35.SEC',
 '01 50 00.sec',
 '01 57 19.01.sec',
 '01 57 19.sec',
 '01 74 19.sec',
 '01 78 00.SEC',
 '02 61 13.sec',
 '02 65 00.sec',
 '02 81 00.sec',
 '02 82 00.sec',
 '02 83 00.sec',
 '02 84 16.sec',
 '02 84 33.sec',
 '03 30 00.SEC',
 '05 05 23.13 10.SEC',
 '05 05 23.16.SEC',
 '05 12 00.SEC',
 '05 30 00.SEC',
 '05 50 13.SEC',
 '05 51 00.sec',
 '05 51 33.SEC',
 '05 52 00.SEC',
 '06 10 00.sec',
 '06 41 16.00 10.SEC',
 '07 05 23.sec',
 '07 14 00.SEC',
 '07 21 16.sec',
 '07 27 10.00 10.sec',
 '07 52 00.SEC',
 '07 60 00.SEC',
 '07 92 00.SEC',
 '08 11 13.sec',
 '08 11 16.SEC',
 '08 14 00.SEC',
 '08 31 00.sec',
 '08 32 13.sec',
 '08 41 13.sec',
 '08 51 13.SEC',
 '08 51 23.sec',
 '08 71 00.SEC',
 '08 81 00.SEC',
 '08 91 00.sec',
 '09 22 00.SEC',
 '09 29 00.SEC',
 '09 30 10.SEC',
 '09 51 00.SEC',
 '09 65 00.SEC',
 '09 68 00.SEC',
 '09 90 00.SEC',
 '10 14 00.10.SEC',
```

```
'10 14 00.20.SEC',
'10 21 13.SEC',
'10 26 00.sec',
'10 28 13.SEC',
'12 21 00.SEC',
'14 21 23.sec',
'21 13 13.sec',
'21 30 00.SEC',
'22 00 00.sec',
'23 05 15.SEC',
'23 05 48.19.sec',
'23 07 00.SEC',
'23 23 00.SEC',
'23 30 00.SEC',
'23 64 10.SEC',
'23 64 26.SEC',
'23 65 00.SEC',
'23 81 00.SEC',
'26 05 48.00 10.sec',
'26 11 16.00 33.SEC',
'26 12 19.10.SEC',
'26 20 00.sec',
'26 23 00.SEC',
'26 24 13.SEC',
'26 29 23.sec',
'26 41 00.sec',
'26 51 00.SEC',
'26 56 00.SEC',
'27 10 00.SEC',
'31 00 00.SEC',
'31 21 13.sec',
'31 62 13.20.SEC',
'32 11 20.SEC',
'32 11 23.SEC',
'32 12 13.SEC',
'32 12 16.16.SEC',
'32 15 00.SEC',
'32 16 19.SEC',
'32 31 13.SEC',
'33 11 00.SEC',
'33 30 00.SEC',
'33 40 00.SEC',
'33 71 02.SEC',
'33 82 00.SEC',
'34 11 00.sec',
'35 59 13.16.sec']
```

Parse and Process Files

The following route is intended to batch parse all files in a folder, add the relevant REF data to a global list, then export to an Excel report.

Assumptions

- The **0x81** character was previously removed from all files
- Files may contain \<DEL> tags, which will be processed at the Soup level

Helper Functions

In [15]:
```
def split_titles(title):
    pattern = '^\(.*?\)\s*'
    x = re.split(pattern, title)

    if len(x) > 1:
        # Recover the edition and clean the text
        x[0] = re.findall(pattern, title)[0].replace('\n','').strip()
        x[0] = x[0].strip('()')

        # Swap indicies so that edition is the last element
        x[0], x[1] = x[1], x[0]

    return x
```

Main Function

In [16]:
```
def get_standards_list(folder):

    standards = []

    for file in os.listdir(folder):
        file_path = folder + file
        if Path(file_path).suffix.lower() == '.sec':
            try:
                with open(file_path, 'r') as doc:
                    soup = bs.BeautifulSoup(doc.read(), 'lxml')
                    kill_tags(soup, 'del')

                    section = {
                        'number': soup.find('scn').text[8:],
                        'name': soup.find('stl').text,
                        'date': soup.find('dte').text
                        }

                    ref_elements = soup.find_all('ref')
                    if len(ref_elements) > 0:
                        orgs = []
                        abbrev_string = '\(.*?\)$'
                        for ref in ref_elements:
                            raw = ref.find('org').text.replace('\n','').strip()
                            parsed = re.split(abbrev_string, raw)
                            if len(parsed) > 1:
                                parsed[0] = parsed[0].strip()
                                parsed[1] = re.findall(abbrev_string, raw)[0].strip('(
                            orgs.append(parsed)

                        for i in range(len(ref_elements)):
                            org_name = orgs[i][0]
                            org_abbrev = orgs[i][1]

                            ids = ref_elements[i].find_all('rid')
                            titles = ref_elements[i].find_all('rtl')

                            for j in range(len(ids)):
                                this_id = ids[j].text.replace('\n','').strip()
                                this_title = titles[j].text.replace('\n','').strip()
                                title = split_titles(this_title)
                                if len(title) > 1:
```

```
                                    standards.append([section['number'],
                                                      section['name'],
                                                      section['date'],
                                                      org_name,
                                                      org_abbrev,
                                                      this_id,
                                                      title[0],
                                                      title[1]])
                                else:
                                    standards.append([section['number'],
                                                      section['name'],
                                                      section['date'],
                                                      org_name,
                                                      org_abbrev,
                                                      this_id,
                                                      title[0],
                                                      ''])
            except:
                pass

    return standards
```

Produce Standards Report for JES

In [17]:
```
start_time = time.perf_counter()

jes_standards = get_standards_list(jes_cleaned)

if len(jes_standards) > 0:
    standards_report = parent_folder + '\\JES Standards Listing ' + '{:%Y%m%d %H%M%S}'

    df = pd.DataFrame(jes_standards)
    df.index = np.arange(1, len(df) + 1)
    df.to_excel(standards_report, header=['Section',
                                          'Name',
                                          'Date',
                                          'Organization',
                                          'Abbrev',
                                          'Standard',
                                          'Title',
                                          'Edition'])

    print('JES Standards List write complete.')

end_time = time.perf_counter()
total_time = end_time - start_time

print(f'{len(jes_cleaned)} files with {len(jes_standards)} standards processed in {tot
```

```
JES Standards List write complete.
73 files with 2036 standards processed in 6.854137799999997 sec
```

Produce Standards Report for UFSG

The following is expected to take a few minutes to run.

In [18]:
```
start_time = time.perf_counter()

ufgs_standards = get_standards_list(ufgs_cleaned)
```

```
if len(ufgs_standards) > 0:
    standards_report = parent_folder + '\\UFGS Standards Listing ' + '{:%Y%m%d %H%M%S}

    df = pd.DataFrame(ufgs_standards)
    df.index = np.arange(1, len(df) + 1)
    df.to_excel(standards_report, header=['Section',
                                          'Name',
                                          'Date',
                                          'Organization',
                                          'Abbrev',
                                          'Standard',
                                          'Title',
                                          'Edition'])

    print('UFGS Standards List write complete.')

end_time = time.perf_counter()
total_time = end_time - start_time

print(f'{len(ufgs_cleaned)} files with {len(ufgs_standards)} standards processed in {t
```

```
UFGS Standards List write complete.
74 files with 18515 standards processed in 53.3476292 sec
```