Find 0x81 Character in JES Standards

Jupyter Notebook written by Ben Fisher on 26 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
import bs4 as bs
import lxml
import pandas as pd
import numpy as np
from pathlib import Path
```

In [2]:
```
parent_folder = os.getcwd()
```

In [3]:
```
test_file = parent_folder + r'\JES\05 12 00.sec'
```

The Culprit

The character is byte 0x81, which is Unicode 129. In Jupyter Notebook (.ipynb) files, it appears as a bullet (see below), but this is just so that it is visible - it is a non-printing character in the wild, and difficult to locate with regular search tools.

This character should be removed from JES versions, because it potentially interferes with reading and parsing .sec files, which are ANSI encoded (windows-1252) XML files.

In [4]:
```
the_character = '
print(ord(the_character))
```

```
129
```

Reading Files and Locating the Character

Try Parsing with BS4

In [5]:
```
with open(test_file, 'r') as f:
    try:
        soup = bs.BeautifulSoup(f.read(), 'lxml')
    except UnicodeDecodeError as e:
        print(e)
```

```
'charmap' codec can't decode byte 0x81 in position 8696: character maps to <undefined
>
```

It appears that attempting to parse the file with bs4 is not going to work. Nor will parsing the file in ANSI or UTF-8 encodings. Next step: read the file byte-wise.

Read File Binary

The following function is intended to return a list of line numbers, which can be used with Notepad++ or a similar text reader.

In [6]:
```
def suspicious_lines(file):
    suspect_lines = []
    with open(file, 'rb') as f:
        lines = f.readlines()
        for i in range(len(lines)):
            before_size = len(lines[i])
            after_size = len(lines[i].replace(b'\x81',b''))
            if not after_size == before_size:
                # Return the line number, but as base 1, not base 0
                suspect_lines.append(i + 1)

    return suspect_lines
```

In [7]:
```
lines = suspicious_lines(test_file)
lines
```

Out[7]:
```
[180]
```

Batch Process Files

The following list of sections is from a different analysis (ref. to *JES Standards Scrape with BeautifulSoup (v2)* Jupyter Notebook)

In [8]:
```
ufgs_sections = ['02 83 00.SEC']
```

In [9]:
```
jes_sections = ['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']
```

In [10]:
```
jes_directory = parent_folder + '\\JES\\'
ufgs_directory = parent_folder + '\\UFGS\\'
ufgs_files = [ufgs_directory + x for x in ufgs_sections]
jes_files = [jes_directory + x for x in jes_sections]
```

In [11]:
```
def get_all_lines(files):
    all_lines = []

    for file in files:
        lines = suspicious_lines(file)
        all_lines.append([Path(file).stem, lines])
    return all_lines
```

In [12]:
```
def make_df(lines):
    df = pd.DataFrame(lines, columns=['Section','Occuring Lines'])
    df.index = np.arange(1, len(df) + 1)
    return df
```

In [13]:
```
jes_lines = get_all_lines(jes_files)
jes_df = make_df(jes_lines)
jes_df
```

Out[13]:

	Section	Occuring Lines
1	02 83 00	[1632]
2	05 12 00	[180]
3	05 30 00	[128]
4	05 51 33	[101, 103]
5	05 52 00	[108, 117, 123]
6	07 60 00	[128, 130, 131]
7	08 11 16	[117, 119]
8	08 31 00	[134]
9	08 91 00	[146]
10	10 14 00.10	[121]
11	10 14 00.20	[186, 188]
12	21 30 00	[447]
13	22 00 00	[387, 460, 463, 466, 469, 516]
14	23 05 15	[103, 135, 145, 147, 149, 151, 155]
15	23 07 00	[185, 187]
16	23 23 00	[395]
17	23 30 00	[169, 174, 183, 185, 3525]
18	23 64 26	[115, 125, 176, 179, 194, 197, 200, 221]
19	23 65 00	[77]
20	23 81 00	[121]
21	26 11 16.00 33	[100]
22	33 11 00	[129]

In [14]:
```
ufgs_lines = get_all_lines(ufgs_files)
ufgs_df = make_df(ufgs_lines)
ufgs_df
```

Out[14]:

	Section	Occuring Lines
1	02 83 00	[1347]

NOTE: that the error in section **02 83 00** is inherited from the UFGS version, however, it appears on a different line number only because there are additional lines in the JES version.

Export to Excel

In [15]:
```
def export_df(title, df, parent_folder):
    error_lines_report = parent_folder + f'\\{title} Error Lines ' + '{:%Y%m%d %H%M%S}
    if len(df) > 0:
```

```
        df.to_excel(error_lines_report, header=['Section',
                                                'Error Lines'])
```

In [16]:
```
export_df('UFGS', ufgs_df, parent_folder)
```

In [17]:
```
export_df('JES', jes_df, parent_folder)
```