```
#! /usr/bin/python3

'''
Author: Christen Ford
Since: 02/22/2019
Name: State of the Union Scraper

Purpose:
This script connects to the infoplease state of the union website and scrapes the speaker, date, and text of the state of the union speech for every speech on the website. Each speech is stored in a separate file; likewise, the scraper generates a master file that in addition to containing the aforementioned information on each speech, also contains the filepath to the speech on disk as well as the web address of the speech.

Notes:
This software in accordance with standards set forth by the Python Software Foundation has been written for Python 3, it will not currently function using Python2. However, if one wishes to do so, they are free to backport this software Python2.

That said, all commands (whether running the software or installing the necessary pip packages) must be done using Python3 and pip3.

certifi, parsel, and urllib3 are third party python libraries; the easiest way to acquire them is through PyPi:
    pip[3] install --user certifi parsel urllib3 (Linux/Mac)
    pip -m install --user certifi parsel urllib3 (Windows)
all other libraries used by this software are standard python libraries

License: This software is distributed AS-IS with no warranty express or implied. The author of this software assumes no responsibility for any issues that arise from its use either technical or legal. You are free to modify this software as you see fit, however it is not available for use in commercial applications. Any version of this software must retain this license verbatim.
'''

import certifi      # certificate validation
import csv          # python csv library
import logging      # error logging
import os           # directory management
import parsel       # XPath parser
import urllib3      # HTTP client

# declare variables needed later
_sotu_master_addr = 'https://www.infoplease.com/homework-help/history/collected-state-union-addresses-us-presidents'
_sotu_root_addr = 'https://www.infoplease.com/homework-help/us-documents/state-union-address-'
_speeches_txt = 'all_speeches.txt'
_speech_dir = './Speeches'
_speeches_csv = 'all_speeches.csv'

# The following snippet was taken from https://doc.scrapy.org/en/xpath-tutorial/topics/xpath-tutorial.html
#
# Below is a small "hack" to change the representation of extracted nodes when using parsel.
# This is to represent return values as serialized HTML element or
# string, and not parsel's wrapper objects.
#
parsel.Selector.__str__ = parsel.Selector.extract
parsel.Selector.__repr__ = parsel.Selector.__str__
parsel.SelectorList.__repr__ = lambda x: '[{}]'.format(
    '\n '.join("({}) {!r}".format(i, repr(s))
                for i, s in enumerate(x, start=1))
).replace(r'\n', '\n')
```

```
50
51  # create some variables for use later
52
53  def scrape():
54      '''
55      Part1: Create a pool and get the master web page containing links to all the speeches
56      '''
57
58      # create a request pool and get the master web page
59      pool = urllib3.PoolManager(
60          cert_reqs='CERT_REQUIRED',
61          ca_certs=certifi.where())
62      req = pool.request('GET', _sotu_master_addr)
63
64      # create a parsel selector from the requests data
65      parent = parsel.Selector(text=str(req.data))
66
67      # select the span article tags that contains our data
68      articles = parent.xpath('//span[contains(@class, \'article\')]').getall()
69
70      # stores the speeches so we can write them out later
71      speeches = []
72
73      # step through each article
74      for article in articles:
75          '''
76          Part 2: Build the file path for the speech
77          '''
78          # select the text from the a element
79          text = parsel.Selector(article).xpath('//a/text()')[0].get()
80
81          # lowercase the text
82          text = text.lower()
83
84          # split the text
85          parts = text.split(" ")
86
87          # conditionally treat the split words
88          #   this is terrible, needs refactoring, only works
89          #   because I'm fairly confident the dataset won't change
90          if len(parts) == 6:  # all parts present, most common
91              parts[1] = parts[1].replace(".", "")
92              parts[3] = parts[3].replace("(", "")
93              parts[4] = parts[4].replace(',', "")
94              parts[5] = parts[5].replace(")", "")
95          elif len(parts) == 5:  # no middle name, occurs a few times
96              parts[2] = parts[2].replace("(", "")
97              parts[3] = parts[3].replace(",", "")
98              parts[4] = parts[4].replace(")", "")
99          elif len(parts) == 4:  # no middle name and date, occurs once
100             parts[2] = parts[2].replace("(", "")
101             parts[3] = parts[3].replace(")", "")
102
103         # build the url string and key for the dictionary
104         sotu_path = ""
105         for part in parts:
106             # special case 1: chester's middle name is ignored in the url
107             if parts[0] == "chester" and part == "a":
108                 continue
109             sotu_path += part + "-"
110         # strip of the trailing hyphen
111         sotu_path = sotu_path.rstrip("-")
112
113         '''
114         Part 3: Get the speech html document and build the speech
115         '''
116         # for the weblink
```

```
117             weblink = _sotu_root_addr + sotu_path
118             # try to get the web page using the normal route
119             req = pool.request('GET', weblink)
120             # if the normal route failed, use the alternative route
121             #   used for certain speeches where infoplease doees not use their
122             #   standard address format
123             if req.status == 404:
124                 weblink = _sotu_root_addr.replace("state-union-address-", "") + sotu_path
125                 req = pool.request('GET', weblink)
126             # create a selector to run XPath on
127             child = parsel.Selector(text=str(req.data))
128 
129             # get all the speech parts from the article div's p elements
130             speech_parts = ""
131             if len(parts) == 6 and parts[0] == "george" and parts[5] == "2006":
132                 speech_parts = child.xpath("//div[contains(@class,
                    \'section\')]/p/text()").getall()
133             else:
134                 speech_parts = child.xpath("//div[contains(@class,
                    \'article\')]/p/text()").getall()
135 
136             # get the speech file name
137             filelink = _speech_dir + "/" + sotu_path + ".txt"
138 
139             # build the speech
140             speech = ""
141             for part in speech_parts:
142                 # strip leading and trailing spaces since the formatting is wonked
143                 speech += (part.strip() + " ")
144             # strip the final trailing space
145             speech = speech.strip()
146 
147             # store the speech in the speeches dictionary
148             # speeches are stored as tuples in the format:
149             #   (name, date, filelink, weblink, speech)
150             name = ""
151             if len(parts) == 6:
152                 name = "{0} {1} {2}".format(parts[0].capitalize(), parts[1].capitalize(),
                    parts[2].capitalize())
153             else:
154                 name = "{0} {1}".format(parts[0].capitalize(), parts[1].capitalize())
155             date = ""
156             if len(parts) == 6:
157                 date = "{0} {1} {2}".format(parts[3].capitalize(), parts[4], parts[5])
158             elif len(parts) == 5:
159                 date = "{0} {1} {2}".format(parts[2].capitalize(), parts[3], parts[4])
160             else:
161                 date = "{0} {1}".format(parts[2].capitalize(), parts[3])
162 
163             # store the speech in the speeches container
164             speeches.append((name, date, filelink, weblink, speech))
165 
166     return speeches
167 
168 def write_to_file(speeches):
169     # make the speech directory if it does not already exist
170     if not os.path.exists(_speech_dir):
171         os.mkdir(_speech_dir)
172 
173     # open the all_speeches file, overwrite the old file
174     speeches_file = open(_speeches_txt, 'w')
175 
176     for speech in speeches:
177         try:
178             # write the speech to its own file
179             with open(speech[2], 'w') as out:
180                 out.write(speech[4])
```

```
181
182             # write the speech to the all_speeches file
183             speeches_file.write(speech[4] + "\n")
184         except IOError:
185             logging.exception('There was an error writing the speech to file!')
186
187     # close the all_speeches file
188     speeches_file.close()
189
190 def write_to_csv(speeches):
191     try:
192         # write all of the speeches to the csv file
193         with open(_speeches_csv, 'w') as csv_file:
194             # write to the csv file
195             csvwriter = csv.writer(csv_file, delimiter=',', quoting=csv.QUOTE_MINIMAL)
196             # write the header row
197             csvwriter.writerow(['Name', 'Date', 'Filelink', 'Weblink', 'Speech'])
198             for speech in speeches:
199                 # write the speeches
200                 csvwriter.writerow([*speech])
201     except IOError:
202         logging.exception('There was an issue writing the speeches to CSV!')
203
204 # scrape the speeches, and write them to file
205 scraped = scrape()
206 write_to_file(scraped)
207 write_to_csv(scraped)
208
```