

```

1  from datetime import datetime, timedelta
2  from bs4 import BeautifulSoup
3  from urllib.request import urlopen
4
5
6  def parse_station(station):
7      '''
8      This function parses the web pages downloaded from wunderground.com
9      into a flat CSV file for the station you provide it.
10
11      Make sure to run the wunderground scraper first so you have the web
12      pages downloaded.
13      '''
14
15      # Scrape between July 1, 2014 and July 1, 2015
16      # You can change the dates here if you prefer to parse a different range
17      current_date = datetime(year=2014, month=7, day=1)
18      end_date = datetime(year=2015, month=7, day=1)
19
20      with open('{} .csv'.format(station), 'w') as out_file:
21          out_file.write('date,actual_mean_temp,actual_min_temp,actual_max_temp,'
22                        'average_min_temp,average_max_temp,'
23                        'record_min_temp,record_max_temp,'
24                        'record_min_temp_year,record_max_temp_year,'
25                        'actual_precipitation,average_precipitation,'
26                        'record_precipitation\n')
27
28      while current_date != end_date:
29          try_again = False
30          with open('{} / {} - {} - {} .html'.format(station,
31                                                    current_date.year,
32                                                    current_date.month,
33                                                    current_date.day)) as in_file:
34              soup = BeautifulSoup(in_file.read(), 'html.parser')
35
36              weather_data = soup.find(id='historyTable').find_all('span', class_='wx-
value')
37              weather_data_units = soup.find(id='historyTable').find_all('td')
38
39              try:
40                  actual_mean_temp = weather_data[0].text
41                  actual_max_temp = weather_data[2].text
42                  average_max_temp = weather_data[3].text
43                  record_max_temp = weather_data[4].text
44                  actual_min_temp = weather_data[5].text
45                  average_min_temp = weather_data[6].text
46                  record_min_temp = weather_data[7].text
47                  record_max_temp_year = weather_data_units[
48                      9].text.split('(')[-1].strip(' ')
49                  record_min_temp_year = weather_data_units[
50                      13].text.split('(')[-1].strip(' ')
51
52                  actual_precipitation = weather_data[9].text
53                  if actual_precipitation == 'T':
54                      actual_precipitation = '0.0'
55                  average_precipitation = weather_data[10].text

```

```

56         record_precipitation = weather_data[11].text
57
58         # Verify that the parsed data is valid
59         if (record_max_temp_year == '-1' or record_min_temp_year == '-1' or
60             int(record_max_temp) < max(int(actual_max_temp),
int(average_max_temp)) or
61             int(record_min_temp) > min(int(actual_min_temp),
int(average_min_temp)) or
62             float(actual_precipitation) > float(record_precipitation) or
63             float(average_precipitation) > float(record_precipitation)):
64             raise Exception
65
66         out_file.write('{}-{}-{}'.format(current_date.year,
current_date.month, current_date.day))
67         out_file.write(', '.join([actual_mean_temp, actual_min_temp,
actual_max_temp,
68                                 average_min_temp, average_max_temp,
69                                 record_min_temp, record_max_temp,
70                                 record_min_temp_year, record_max_temp_year,
71                                 actual_precipitation, average_precipitation,
72                                 record_precipitation]))
73         out_file.write('\n')
74         current_date += timedelta(days=1)
75     except:
76         # If the web page is formatted improperly, signal that the page may
need
77         # to be downloaded again.
78         try_again = True
79
80         # If the web page needs to be downloaded again, re-download it from
# wunderground.com
81
82         # If the parser gets stuck on a certain date, you may need to investigate
# the page to find out what is going on. Sometimes data is missing, in
# which case the parser will get stuck. You can manually put in the data
# yourself in that case, or just tell the parser to skip this day.
87         if try_again:
88             print('Error with date {}'.format(current_date))
89
90             lookup_URL =
91             'http://www.wunderground.com/history/airport/{}/{}{}{}/DailyHistory.html'
92             formatted_lookup_URL = lookup_URL.format(station,
93                                                         current_date.year,
94                                                         current_date.month,
95                                                         current_date.day)
96             html = urlopen(formatted_lookup_URL).read().decode('utf-8')
97
98             out_file_name = '{}-{}-{}.html'.format(station,
99                                                         current_date.year,
100                                                         current_date.month,
101                                                         current_date.day)
102
103             with open(out_file_name, 'w') as out_file:
104                 out_file.write(html)
105
106         # Parse the stations used in this article

```

```
107 for station in ['KCLT', 'KCQT', 'KHOU', 'KIND', 'KJAX',  
108                 'KMDW', 'KNYC', 'KPHL', 'KPHX', 'KSEA']:  
109     parse_station(station)  
110
```