Open many files

January 28, 2019

1 How to open a list of files and concatenate into a single pandas data frame

1.1 This notebook shows how we can open a list of files and have a single table of data in pandas.

1.1.1 We use the function `pandas.concat()`, which puts one data frame *after* the other, that means that the *columns* must be the same!

For this example, we have three csv files, with three days of data, one day each. They have columns for a timestamp, temperature and relative humidity. The files are in a subfolder named ***three_files***.

We will open all files, one after the other inside a `for` loop, and concatenate them into a single dataframe. In the end, we can save that file to a new dataframe.

Also, we will only select one column, the temperature, to show how this selection can be done.

1.2 Import libraries

```
In [1]: import pandas as pd
```

1.3 Create a list of the files names to iterate over them

1.3.1 Method 1: Write them to a list

```
In [2]: files_to_read = [ '2018-04-18.csv', '2018-04-19.csv', '2018-04-20.csv' ]
```

That works if there are not many files, and also if there are many different files in the same folder, but we don't want to read all of them.

1.3.2 Method 2: Read all the files in the folder

```
In [3]: import os
```

```
In [4]: files_to_read = os.listdir( 'three_files/' ) # returns list with everything in that di
        print( files_to_read )
```

```
['2018-04-18.csv', '2018-04-20.csv', 'some_empty_file.txt', '2018-04-19.csv']
```

That works if we want many files, all of them. If you want to filter some in or out, you need to read as shown, and then delete elements from the list.

For example, you can select only files with a `.csv` extension adding the following line after reading the directory:

```
In [5]: files_to_read = os.listdir( 'three_files/' ) # returns list with everything in that di

        files_to_read = [ name for name in files_to_read if '.csv' in name ] # Select only tho

        print( files_to_read )
```

```
['2018-04-18.csv', '2018-04-20.csv', '2018-04-19.csv']
```

1.4 Make a loop and concatenate all files

1.4.1 Method 1: *normal* `for` loop

We first create an empty data frame.

```
In [6]: big_dataframe1 = pd.DataFrame( )
```

We use a `for` loop to read each file and concatenate into the big data frame:

```
In [7]: for current_file in files_to_read:

            current_df = pd.read_csv( 'three_files/'+current_file ) # Care for the subfolder o

            current_df = current_df[ [ 'Date and Time', 'Temp. (řC)' ] ] # Select only the col

            big_dataframe1 = pd.concat( [ big_dataframe1, current_df ] )
```

```
In [8]: big_dataframe1.shape
```

```
Out[8]: (855, 2)
```

Or we specify the columns while reading the file:

```
In [9]: big_dataframe2 = pd.DataFrame( )
```

```
In [10]: for current_file in files_to_read:

             current_df = pd.read_csv( 'three_files/'+current_file, usecols=[ 'Date and Time',

             big_dataframe2 = pd.concat( [ big_dataframe2, current_df ] )
```

```
In [11]: big_dataframe2.shape
```

```
Out[11]: (855, 2)
```

1.4.2 Method 2: list comprehension

```
In [12]: big_dataframe3 = pd.concat( [ pd.read_csv( 'three_files/'+current_file, usecols=[ 'Da

In [13]: big_dataframe3.shape

Out[13]: (855, 2)
```

You can break that big line into several for better reading, python keeps track of open parenthesis and brackets:

```
In [14]: big_dataframe3 = pd.concat( [ pd.read_csv( 'three_files/'+current_file,
                                                    usecols=[ 'Date and Time', 'Temp. (řC)' ] )
                                       for current_file in files_to_read] )

In [15]: big_dataframe3.shape

Out[15]: (855, 2)
```

1.5 Write to a file

Depending which method you used, you can save to a file activating the corresponding line:

```
In [16]: # big_dataframe1.to_csv( 'big_table.csv' )
         # big_dataframe2.to_csv( 'big_table.csv' )
         # big_dataframe3.to_csv( 'big_table.csv' )
```

Common options that you might want to use are:

```
In [17]: # big_dataframe1.to_csv( 'big_table.csv', sep=';', index=False )
         # big_dataframe2.to_csv( 'big_table.csv', sep=';', index=False )
         # big_dataframe3.to_csv( 'big_table.csv', sep=';', index=False )
```
