Handling Excel Files with Pandas

In [1]:

```
import pandas as pd
import numpy as np
from pandas import ExcelWriter
from pandas import ExcelFile
import openpyxl
import os
```

1. Read and Write CSV Files

In [2]:

```
df = pd.read_csv('data/class1_test1.tsv', delimiter='\t')
df
```

Out[2]:

	name	english	maths	science
0	Aaron	70	46	47
1	Adrian	72	40	95
2	Alby	49	65	64
3	Abner	86	40	96
4	Benett	50	98	69
5	Brion	81	92	95
6	Collin	45	83	45
7	Cyril	60	46	74
8	Dylan	72	90	74

In [3]:

```
df.to_csv?
```

In [4]:

```
df.to_csv('data/class1_test1_output.csv')
```

By default, row and column labels, i.e. index and columns, will be exported too.

- To ignore `index` in the output, add parameter `index=False` .
- To ignore `header` in the output, add parameter `header=False` .

In [5]:

```
df.to_csv('data/class1_test1_output.csv', index=False, header=False)
```

2. Read and Write Excel Files

Write to an Excel File

Pandas provide a `ExcelWriter` class to write data to Excel File.

- ExcelWriter will **overwrite** existing data

In [6]:

```
writer = pd.ExcelWriter('data/_test2.xlsx', engine='xlsxwriter')
df.to_excel(writer, 'Sheet1')
writer.save()
```

Utility Function to Create New Excel File

In [7]:

```
import os

def create_excel_file(file, remove_existing=False):
    if os.path.exists(file):
        if remove_existing:
            os.remove(file)
    else:
        writer = pd.ExcelWriter(file, engine='xlsxwriter')
        writer.save()
```

In [8]:

```
create_excel_file('data/_test.xlsx')
```

Read from an Excel File

In [9]:

```
pd.read_excel?
```

In [10]:

```
df = pd.read_excel('data/_test.xlsx')
df
```

Out[10]:

	Unnamed: 0	name	english	maths	science
0	0	Aaron	70	46	47
1	1	Adrian	72	40	95
2	2	Alby	49	65	64
3	3	Abner	86	40	96
4	4	Benett	50	98	69
5	5	Brion	81	92	95
6	6	Collin	45	83	45
7	7	Cyril	60	46	74
8	8	Dylan	72	90	74

Utility Function to Read Excel File

In [11]:

```
def read_raw_excel(file, sheet_name=0):
    df = pd.read_excel(file, sheet_name)
    ## clean headers because some column names are multiple lines (by ALT)
    df.columns = [' '.join(c.split()) for c in df.columns]

    return df
```

In [12]:

```
df = read_raw_excel('data/_test.xlsx', 'Sheet1')
df
```

Out[12]:

	Unnamed: 0	name	english	maths	science
0	0	Aaron	70	46	47
1	1	Adrian	72	40	95
2	2	Alby	49	65	64
3	3	Abner	86	40	96
4	4	Benett	50	98	69
5	5	Brion	81	92	95
6	6	Collin	45	83	45
7	7	Cyril	60	46	74
8	8	Dylan	72	90	74

In [13]:

```
df.drop('Unnamed: 0', axis=1, inplace=True)
df
```

Out[13]:

	name	english	maths	science
0	Aaron	70	46	47
1	Adrian	72	40	95
2	Alby	49	65	64
3	Abner	86	40	96
4	Benett	50	98	69
5	Brion	81	92	95
6	Collin	45	83	45
7	Cyril	60	46	74
8	Dylan	72	90	74

Remove Empty Rows

In [14]:

```
df = df.append(pd.Series(), ignore_index=True)
df
```

Out[14]:

	name	english	maths	science
0	Aaron	70.0	46.0	47.0
1	Adrian	72.0	40.0	95.0
2	Alby	49.0	65.0	64.0
3	Abner	86.0	40.0	96.0
4	Benett	50.0	98.0	69.0
5	Brion	81.0	92.0	95.0
6	Collin	45.0	83.0	45.0
7	Cyril	60.0	46.0	74.0
8	Dylan	72.0	90.0	74.0
9	NaN	NaN	NaN	NaN

In [15]:

```
def df_drop_subset_all_na(df, columns):
    '''Drop rows if all subset columns are NaN'''
    size_bef = len(df)
    df2 = df.dropna(subset=columns, how='all')
    print('Rows reduced after dropping NaN:', size_bef, 'to', len(df2))
    return df2
```

In [16]:

```
df_drop_subset_all_na(df, ['name', 'english', 'maths', 'science'])
```

```
Rows reduced after dropping NaN: 10 to 9
```

Out[16]:

	name	english	maths	science
0	Aaron	70.0	46.0	47.0
1	Adrian	72.0	40.0	95.0
2	Alby	49.0	65.0	64.0
3	Abner	86.0	40.0	96.0
4	Benett	50.0	98.0	69.0
5	Brion	81.0	92.0	95.0
6	Collin	45.0	83.0	45.0
7	Cyril	60.0	46.0	74.0
8	Dylan	72.0	90.0	74.0

Insert Sheet to an Excel File

To specify a sheet to insert data, Use `openpyxl` instead of `XlsxWriter` .

- If the sheet name already exists, it will append numeric value, e.g. `1` , behind the sheet name.

In [17]:

```
def insert_sheet_with_data(excel_file, sheet_name, df):
    '''Write a dataframe to a sheet in an excel file'''
    create_excel_file(excel_file, remove_existing=False)

    print('Insert sheet into file:', sheet_name, excel_file)
    with pd.ExcelWriter(excel_file, engine='openpyxl', mode='a') as writer:
        df.to_excel(writer, sheet_name = sheet_name, index=False)

```

In [18]:

```
insert_sheet_with_data('data/_test.xlsx', 'Sheet8', pd.DataFrame())
```

```
Insert sheet into file: Sheet8 data/_test.xlsx
```

In [19]:

```
insert_sheet_with_data('data/_test.xlsx', 'Sheet9', df)
```

```
Insert sheet into file: Sheet9 data/_test.xlsx
```

Get Sheet Names of an Excel

In [20]:

```
def get_sheet_names(excel_file):
    if not os.path.exists(excel_file): return

    workbook=openpyxl.load_workbook(excel_file)
    return workbook.sheetnames
```

In [21]:

```
get_sheet_names('data/_test.xlsx')
```

Out[21]:

```
['Sheet1', 'Sheet9', 'Sheet8', 'Sheet91']
```

Remove a Sheet from Excel

In [22]:

```
def remove_sheets_with_name(excel_file, sheet_name):
    '''Remove sheets from an Excel, where sheet name starts with sheet_name_prefix'''
    if not os.path.exists(excel_file): return

    workbook=openpyxl.load_workbook(excel_file)
#     print("Existing Sheets:", workbook.sheetnames)
    for name in workbook.sheetnames:
        if name == sheet_name:
            print("Remove sheet from file:", sheet_name, excel_file)
            worksheet=workbook[sheet_name]
            workbook.remove(worksheet)
    workbook.save(excel_file)
#     print("Remaining Sheets: ", workbook.sheetnames)
```

In [23]:

```
remove_sheets_with_name('data/_test.xlsx', 'Sheet8')
```

```
Remove sheet from file: Sheet8 data/_test.xlsx
```

In [24]:

```
remove_sheets_with_name('data/_test.xlsx', 'Sheet10')
```