Backslash

Python Project

Avirup Roy Chowdhury

Pranathi Varma M.

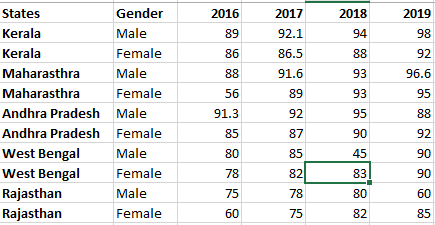
Niharika Gupta

Chaitanya Mohite

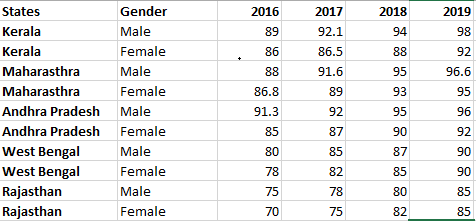
Diptajit Chaurangi

# Problem Statement

To design a python application which will be able to open up 2 Excel files. These files will both be data tables laid out in a similar format as the below mentioned examples.

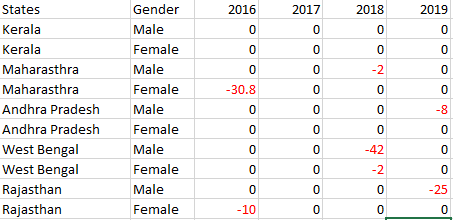


**Fig 1 :** Table 1 for comparison



**Fig 2 :** Table 2 for comparison

It should to be able to tell for each ID what Data Point what has changed with respect to each other. We will be highlighting all the updated rows and columns using a special formatting (in this case the color changes to red).



**Fig 3 :** Comparison Table

# Technologies Used

The project base and structure will be revolving around **Python 3** and will be implemented using two libraries i.e

1. **Pandas -**  an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming
2. **OpenPYXL -**  is a Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files. It was born from lack of existing library to read/write natively from Python

# Scope of the Project

The project will revolve around comparing two excel files and highlighting out the differences to compare values in multiple scenarios, and making a real world application out of it.

# Source Code

## Implementation using Pandas

1. **Import**

*#importing excel to DataFrame*

df1=pd.read\_excel('excel\_file\_old.xlsx') *#Df1 will represent the old file*

df2=pd.read\_excel('excel\_file\_new.xlsx') *#Df2 will represent the new file*

1. **A boolean matrix is generated using the comparison in-built function between the two generated DataFrames**

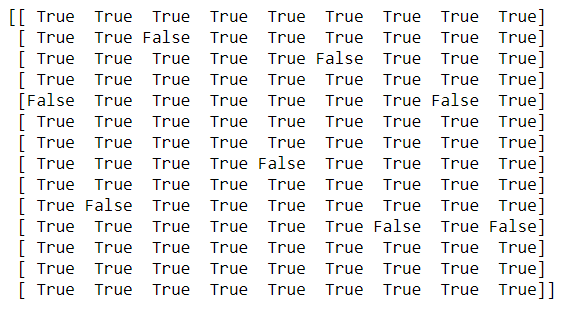


Fig 4 : Boolean Matrix

1. **For different values the value is given as False and for similar values it is given as True**
   1. **If value is True, we subtract the values and it becomes 0**

rows,cols=np.where(comparison\_values==**True**)

**for** item **in** zip(rows,cols):

**if** (type(df1.iloc[item[0],item[1]])!= str):

df3.iloc[item[0],item[1]]=df1.iloc[item[0],

* 1. **If value is False, we subtract the values and it returns a non Zero value**

rows,cols=np.where(comparison\_values==**False**)

**for** item **in** zip(rows,cols):

df3.iloc[item[0],item[1]]=df1.iloc[item[0],item[1]]

1. **Finally the updates values are formatted for better clarity using color\_negative\_red()**

**def** color\_negative\_red(val):

*"""*

*Takes a scalar and returns a string with*

*the css property `'color: red'` for negative*

*strings, black otherwise.*

*"""*

color = 'red' **if** (val != 0 **and** type(val) !=str) **else** 'black'

**return** 'color: **%s**' % color

1. **It is exported into a new file**

s.to\_excel('./Excel\_diff.xlsx',index=**False**,header=**True**)

## Implementation using openpyxl

1. **Setting FilePath for Three Excel Files (Comparison1, Comparison2, NewFile)**

*#set file path for the three excel files*

filepath1="excel\_file\_old.xlsx"

filepath2="excel\_file\_new.xlsx"

filepath3="excel\_file\_diff.xlsx"

1. **Activate All the Sheets in the File**

*#selecting the three excel sheets*

sheet1=wb1.active

sheet2=wb2.active

sheet3=wb3.active

1. **Will Evaluate Maximum number of rows and columns to generate operation size**

*#evaluating the maximum rows and columns*

max\_row1=sheet1.max\_row

max\_column1=sheet1.max\_column

1. **Will print the header(title) of each excel File**

**for** i **in** range(1,2):

**for** j **in** range(1,max\_column1+1):

cell\_obj1=sheet1.cell(row=i,column=j)

sheet3.cell(row=i, column=j).value = cell\_obj1.value

wb3.save(filepath3)

1. **Will compare cell by cell for all the rows and columns and update accordingly**

**for** i **in** range(2,max\_row1+1):

**for** j **in** range(1,max\_column1+1):

cell\_obj1=sheet1.cell(row=i,column=j)

cell\_obj2=sheet2.cell(row=i,column=j)

**if** type(cell\_obj1.value) == str **and** type(cell\_obj2.value) == str :

sheet3.cell(row=i, column=j).value = cell\_obj1.value

**else**:

temp=cell\_obj1.value-cell\_obj2.value

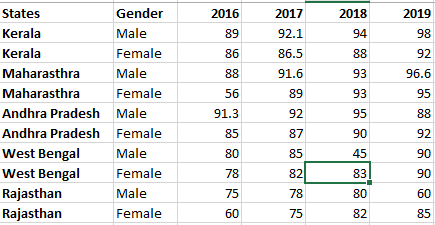
sheet3.cell(row=i, column=j).value = temp

**f) Now we will format the updated values with openpyxl.styles**

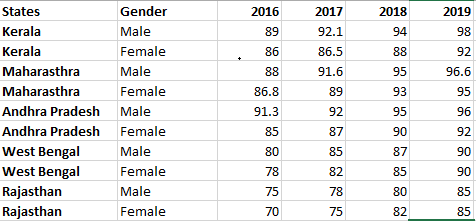
**if** temp!=0:

sheet3.cell(row=i, column=j).font = Font(color=colors.RED)

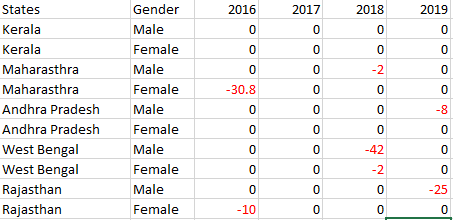
# Screenshots



**Fig 5 :** excel\_file\_new.xlsx



**Fig 6 :** excel\_file\_old.xlsx



**Fig 7 :** excel\_file\_diff.xlsx

# Result

The files have been compared using execution of two technologies and a new file has been generated in all instances.

# Libraries and Functions Used :

1. **Pandas** –
   1. ***read\_excel*** - Read an Excel file into a pandas DataFrame.
   2. ***iloc()* -** Purely integer-location based indexing for selection by position.
   3. ***style()* –** to apply conditional formatting
      1. **applymap() -** Apply a function to a Dataframe elementwise
   4. ***to\_excel()* -** Write object to an Excel sheet.
2. **Numpy –** 
   1. ***where()* -** Return elements chosen from *x* or *y* depending on *condition*.
3. **Python –** 
   1. ***copy() -*** This module provides generic shallow and deep copy operations
   2. ***zip() -*** Make an iterator that aggregates elements from each of the iterables.
   3. ***type() -*** With one argument, return the type of an *object*.
   4. ***range() -*** Rather than being a function, [range](https://docs.python.org/3.3/library/stdtypes.html#range) is actually an immutable sequence type
4. **User-defined Functions –**
   1. ***color\_negative\_red() –*** adds red colour to the font.
5. **openpyxl**
   1. ***Workbook -*** A workbook is always created with at least one worksheet.
   2. ***Load\_Workbook –*** Loading a workbook into an object for i/o operations
   3. ***Styles() -*** Styles are used to change the look of your data while displayed on screen.
      1. **Fill -** fill to set a pattern or color gradient
      2. **Font -** font to set font size, color, underlining, etc.
      3. **Color –** to import colour codes
      4. **Colors –** to import colour codes
   4. ***Save() -*** The simplest and safest way to save a workbook
   5. ***Active() –*** To activate a workbook
   6. ***Max\_row –*** count maximum rows in a worksheet
   7. ***Max\_column –*** count maximum columns in a worksheet
   8. ***Cell(row,column) –*** returns cell in the actual excel sheet