Handling Excel

Excel Automation

In order to automate the excel spread sheet, we use the following module.

Pandas

Automating the excel using the PANDAS

- With Excel being so pervasive, data professionals must be familiar with it. You'll also want a tool that can easily read and write Excel files — Pandas is perfect for this.
- In order to use the pandas, first we need to install the Pandas module.

Steps to install the pandas

1. Open a command prompt
2. Enter the following statement
   "Pip install pandas"

Note:

- To read the data from Excel, Pandas internally uses the module called as "XLRD"
- To write the data to Excel, Pandas uses the module called as "OPENPYEXCEL"

Along with Pandas, we need to install two more modules.

- XLRD
- OpenPyXL

Install the above two modules using "PIP"

```
C:\>pip install xlrd
```

```
C:\>pip install openpyxl
```

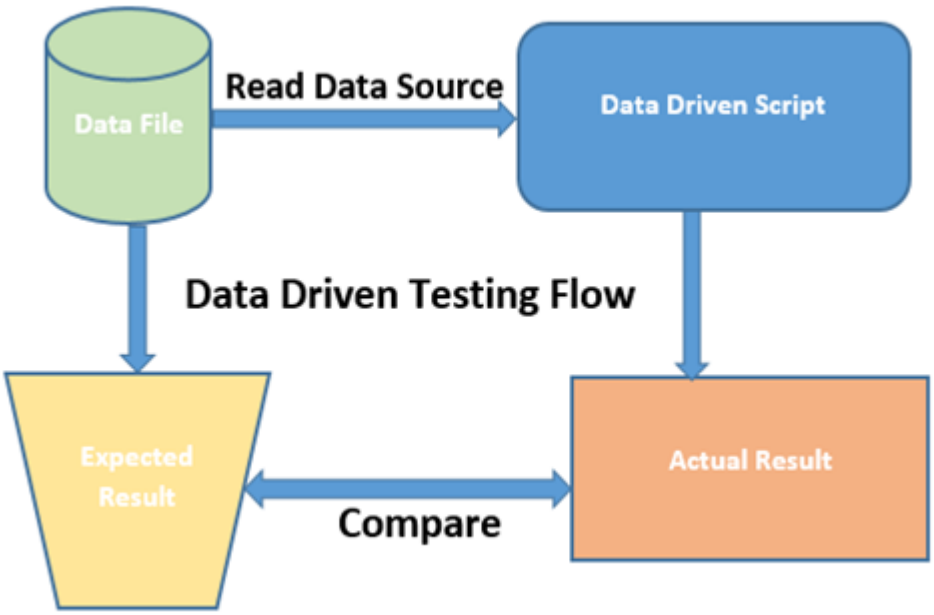
Pandas has excellent methods for reading all kinds of data from Excel files. You can also export your results from pandas back to Excel.

Data-driven test includes the following operations performed in a loop:

- Retrieving input data from storage.
- Entering data in an application form.

- Verifying the results.
- Continuing with the next set of input data.

Data Driven Testing can be understood by the following diagram:



Read data from the Excel file

- We need to first import the data from the Excel file into pandas. To do that, we start by importing the Pandas module.

```
import pandas as PD
```

- We then use the pandas "**read_excel**" method to read in data from the Excel file. The easiest way to call this method is to pass the file name. If no sheet name is specified then it will read the first sheet in the index.

```
import pandas as PD

file_path = "C:/Users/PriyaPramod/Desktop/TestData/Sample.xlsx"
#Load Spread Sheet
xl = PD.read_excel(file_path)
```

- Here, the read_excel method read the data from the Excel file into a pandas DataFrame object. Pandas defaults to storing data in DataFrames. We then stored this DataFrame into a variable called XL.

- **Program to count the number of sheets in excel**

```
import pandas as PD

file_path = "C:\\Users\\PriyaPramod\\Desktop\\DataSource\\TestData.xlsx"

data = PD.ExcelFile(file_path)
total_sheets = len(data.sheet_names)
print("total number of sheets in excel: ", total_sheets)
```

- **Program to print all the sheet names in excel**

```
import pandas as PD

file_path = "C:\\Users\\PriyaPramod\\Desktop\\DataSource\\TestData.xlsx"

data = PD.ExcelFile(file_path)
sheets = data.sheet_names

for sheet in sheets:
    print(sheet)
```

DataFrame

- **DataFrame** is a 2-dimensional labelled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object.

Reading the data from excel based on the sheet name

- Excel files quite often have multiple sheets and the ability to read a specific sheet or all of them is very important.
- To make this easy, the Pandas read_excel method takes an argument called **"sheet_name"** that tells pandas which sheet to read in the data from.
- For this, you can either use the sheet name or the sheet number.
- Sheet numbers start with zero. If the sheet_name argument is not given, it defaults to zero and pandas will import the first sheet.
- By default, pandas will automatically assign a numeric index or row label starting with zero.
- Instead of index, if you want to read the sheet based on his sheet name, than assign the sheet name to the sheet_name variable

Program to read the sheet using the sheet number.

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 xl = PD.read_excel(file_path, sheet_name=0)
5
6 print(xl)
```

Console PyUnit

<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe

```
  UserName Password              Expected_result     Result
0    admin  manager  actiTIME - Enter Time-Track  TEST_PASS
1    admin  mnagher              actiTIME - Login  TEST_PASS
2     user  manager              actiTIME - Login  TEST_PASS
3    admin  manager              actiTIME - Login  TEST_PASS
4    user2   user23              actiTIME - Login  TEST_PASS
```

Program to read the sheet using sheet name

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 print(xl)
```

Console PyUnit

<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]

```
  UserName Password              Expected_result     Result
0    admin  manager  actiTIME - Enter Time-Track  TEST_PASS
1    admin  mnagher              actiTIME - Login  TEST_PASS
2     user  manager              actiTIME - Login  TEST_PASS
3    admin  manager              actiTIME - Login  TEST_PASS
4    user2   user23              actiTIME - Login  TEST_PASS
```

Program to read the single column

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 print("Reading the column user name from the sheet1")
7 column = xl["UserName"]
8 print(column)
```

Console PyUnit

```
<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Reading the column user name from the sheet1
0    admin
1    admin
2     user
3    admin
4    user2
Name: UserName, dtype: object
```

Program to read the multiple columns

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 print("Reading the user name & Password columns from the sheet1")
7 columns = xl[["UserName", "Password"]]
8 print(columns)
```

Console PyUnit

```
<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Reading the user name & Password columns from the sheet1
  UserName Password
0    admin  manager
1    admin  mnagher
2     user  manager
3    admin  manager
4    user2   user23
```

Program to read the single row

- In order to read the rows from the data frame object we need to perform the slicing operation.
- I.E we need to mention the starting row number and the ending row number.
- Below is the Syntax.

```
Df [Starting Row Number : Ending Row Number]
```

Example programs:

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 print("Reading the first row from the sheet1")
7 row = xl[0:1]
8 print(row)
```

Console PyUnit

```
<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe
Reading the first row from the sheet1
  UserName Password              Expected_result     Result
0    admin  manager  actiTIME - Enter Time-Track  TEST_PASS
```

Program to read multiple rows

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 print("Reading the multiple rows from the sheet1")
7 row = xl[0:5]
8 print(row)
```

Console PyUnit

```
<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Reading the multiple rows from the sheet1
  UserName Password              Expected_result     Result
0    admin  manager  actiTIME - Enter Time-Track  TEST_PASS
1    admin  mnagher             actiTIME - Login  TEST_PASS
2     user  manager             actiTIME - Login  TEST_PASS
3    admin  manager             actiTIME - Login  TEST_PASS
4    user2   user23             actiTIME - Login  TEST_PASS
```

Exploring the data

- Now that we have read the data set from our Excel file, we can start exploring it using pandas. A pandas DataFrame stores the data in a tabular format, just like the way Excel displays the data in a sheet. Pandas has a lot of built-in methods to explore the DataFrame we created from the Excel file we just read in.

Let's look at the methods that come in handy while exploring the data set.

Head:

- Pandas has a built-in “head ()” method that we can use to easily display the first few rows of our DataFrame.
- If no argument is passed, it will display first five rows.
- If a number is passed, it will display the equal number of rows from the top.

Program to display the first 5 rows using the head function

```
Excel_using_Pandas_Read_excel
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/Sample.xlsx"
4 #Load Spread Sheet
5 xl = PD.read_excel(file_path)
6
7 print(xl.head())
8
```

```
Console  Debug
<terminated> Excel_using_Pandas_Read_excel.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
  UserName  Password
0    Admin   manager
1   Admin1  Manager1
2    Admin   Manager
3   Admin1  Manager1
4    Admin   Manager
```

- Passing the argument for the head method as 2.
- Now it will display two rows from the top.

```
Excel_using_Pandas_Read_excel
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/Sample.xlsx"
4 #Load Spread Sheet
5 xl = PD.read_excel(file_path)
6
7 print(xl.head(2))
8
```

```
Console  Debug
<terminated> Excel_using_Pandas_Read_excel.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
  UserName  Password
0    Admin   manager
1   Admin1  Manager1
```

Tail Function

- We can use the tail method to view the bottom rows. If no parameter is passed, only the bottom five rows are returned.

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/Sample.xlsx"
4 xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 print("Getting the bottom rows: ")
7 print(xl.tail())
```

```
Console  Debug
<terminated> tail_function.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Getting the bottom rows:
  UserName  Password
3   Admin1  Manager1
4    Admin   Manager
5   Admin1  Manager1
6    Admin   Manager
7   Admin1  Manager1
```

- Passing the argument for the tail method as 2.
- Now it will display two rows from the bottom.

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 bottom_rows = xl.tail(2)
7 print("Printing the rows from the bottom")
8 print(bottom_rows)
```

```
Console  PyUnit
<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Printing the rows from the bottom
  UserName Password   Expected_result     Result
3    admin  manager  actiTIME - Login  TEST_PASS
4    user2   user23  actiTIME - Login  TEST_PASS
```

Shape:

- Shape method returns total number of rows and columns from the sheet.

Program to get the total number of rows and columns from the sheet

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  s = xl.shape
7  print("Total number of rows and columns in the sheet1 is: ", s)
```

```
Console  PyUnit
<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Total number of rows and columns in the sheet1 is:  (5, 4)
```

Note: Shape method returns the tuple.

- **First item in the tuple object represents the total number of rows in a sheet.**
- **Second item in the tuple object represents the total number of columns in a sheet.**

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  s = xl.shape
7  print("Total number of rows and columns in the sheet1 is: ", s)
8  print("Total number of rows: ", s[0])
9  print("Total number of columns: ", s[1])
```

```
Console  PyUnit
<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Total number of rows and columns in the sheet1 is:  (5, 4)
Total number of rows:  5
Total number of columns:  4
```

Program to print the column names?

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  print(xl.columns)
7
```

```
Console  PyUnit
<terminated> Practice.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Index(['UserName', 'Password', 'Expected_result', 'Result'], dtype='object')
```

- If your excel sheet has more number of sheets, we need to create the DataFrame for each sheet by loading the excel file every time.
- Consider, we have a two sheet in the sample excel file. If you want to add both the sheets, we need to create the DataFrame object for both the sheets and concatenate the two DataFrame using the “concat” function.

```
import pandas as PD

file_path = "C:/Users/PriyaPramod/Desktop/TestData/Sample.xlsx"

xl_sheet1 = PD.read_excel(file_path, sheet_name="Sheet1")
xl_sheet2 = PD.read_excel(file_path, sheet_name="Sheet2")

Xl_sheets = PD.concat([xl_sheet1, xl_sheet2])

print(Xl_sheets)
```

Sort Values

- In Excel, you're able to sort a sheet based on the values in one or more columns. In pandas, you can do the same thing with the sort_values method.
- Sort values method takes two arguments, first argument is “**Column Name**” & second argument is “**ascending**” keyword.
- If you assign “**True**” to ascending keyword, then sorting will happen in ascending order.
- If you assign “**False**”, then sorting will happen in descending order.

Program to sort the column in ascending order.

```
import pandas as PD

file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
xl = PD.read_excel(file_path, sheet_name="Sheet1")
print("Before sorting")
print(xl)
print("-----------------")
print("After sorting")
'''Sorting the column using the sort values method '''
ascending_sort = xl.sort_values("UserName", ascending=True)
print(ascending_sort)
```

```
Console  PyUnit
<terminated> Practice.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
0     admin   manager   actiTIME - Enter Time-Track   TEST_PASS
1     admin   mnagher               actiTIME - Login   TEST_PASS
2      user   manager               actiTIME - Login   TEST_PASS
3     admin   manager               actiTIME - Login   TEST_PASS
4     user2    user23               actiTIME - Login   TEST_PASS
----------------
After sorting
  UserName Password                Expected_result      Result
0     admin   manager   actiTIME - Enter Time-Track   TEST_PASS
1     admin   mnagher               actiTIME - Login   TEST_PASS
3     admin   manager               actiTIME - Login   TEST_PASS
2      user   manager               actiTIME - Login   TEST_PASS
4     user2    user23               actiTIME - Login   TEST_PASS
```

Program to sort the column in descending order

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5  print("Before sorting")
6  print(xl)
7  print("----------------")
8  print("After sorting")
9  '''Sorting the column using the sort values method '''
10 descending_sort = xl.sort_values("UserName", ascending=False)
11 print(descending_sort)
```

Console PyUnit

```
<terminated> Practice.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Before sorting
  UserName Password                Expected_result     Result
0    admin  manager  actiTIME - Enter Time-Track  TEST_PASS
1    admin  mnagher              actiTIME - Login  TEST_PASS
2     user  manager              actiTIME - Login  TEST_PASS
3    admin  manager              actiTIME - Login  TEST_PASS
4    user2   user23              actiTIME - Login  TEST_PASS
----------------
After sorting
  UserName Password                Expected_result     Result
4    user2   user23              actiTIME - Login  TEST_PASS
2     user  manager              actiTIME - Login  TEST_PASS
0    admin  manager  actiTIME - Enter Time-Track  TEST_PASS
1    admin  mnagher              actiTIME - Login  TEST_PASS
3    admin  manager              actiTIME - Login  TEST_PASS
```

Set_index method:

➔ This method is used to set any column as an index to the data frame.

For example, setting the index of our test data frame to the persons “Results”:

Selection_iLoc | Selection_iLoc_Slicing | Selection_iLoc_integerList | Multiple_rows | Multiple_columns

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/HTML Pages/TestData/TestData.xlsx"
4  data = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  data.set_index("Result", inplace=True)
7
8  print(data.head())
```

Console

```
<terminated> Selection_Loc_Label_Index_Bases.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
          UserName Password                Expected_result
Result
Test_PASS    admin  manager  actiTIME - Enter Time-Track
Test_FAIL    admin  mnagher  actiTIME - Enter Time-Track
Test_FAIL     user  manager  actiTIME - Enter Time-Track
Test_PASS    admin  manager  actiTIME - Enter Time-Track
Test_FAIL    user2   user23  actiTIME - Enter Time-Track
```

Pandas Data Selection

Using iloc & loc to select rows and columns in Pandas DataFrames

- There are multiple ways to select and index rows and columns from Pandas DataFrames.
- There are two main options to achieve the selection and indexing activities in Pandas. Below are the two methods.
    1. **iloc (Selecting data by row & column numbers)**
    2. **Loc (Selecting data by label)**

Iloc

- The iloc indexer for Pandas DataFrame is used for integer-location based indexing/selection by position.
- Iloc in pandas is used to select rows and columns by number, in the order that they appear in the data frame.
- There are two "arguments" to iloc – a row selector, and a column selector.

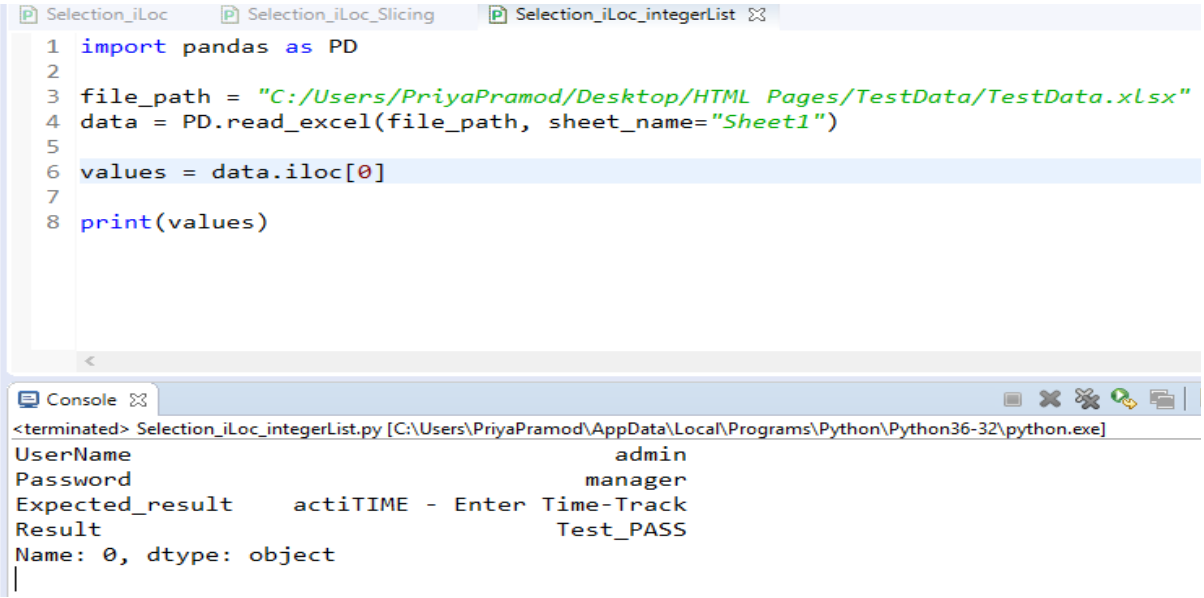
Syntax:

Data.iloc [<row selection>, <column selection>]

Single selections using iloc

 # Operations on Rows:

 # first row of data frame



data.iloc[1] # second row of data frame
data.iloc[-1] # last row of data frame

Program to select the last row

```
Selection_iLoc   Selection_iLoc_Slicing   Selection_iLoc_integerList
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/HTML Pages/TestData/TestData.xlsx"
4 data = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 values = data.iloc[-1]
7
8 print(values)
```

```
Console
<terminated> Selection_iLoc_integerList.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
UserName                                    user2
Password                                   user23
Expected_result    actiTIME - Enter Time-Track
Result                                  Test_FAIL
Name: 4, dtype: object
```

Slicing:

- We can perform the slicing operations to choose the rows between the ranges.

Example: Choose rows from 1st row to 3rd row.

Program to slice the rows from 2nd row to 4th row.

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6 rows = xl.iloc[2:5]
7 print("Printing the rows after slicing")
8 print(rows)
```

```
Console   PyUnit
<terminated> Practice.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Printing the rows after slicing
  UserName Password   Expected_result     Result
2     user  manager  actiTIME - Login  TEST_PASS
3    admin  manager  actiTIME - Login  TEST_PASS
4    user2   user23  actiTIME - Login  TEST_PASS
```

Program to select the rows

Note: In this program, selecting the 5, 2 & 1 rows.

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  rows = xl.iloc[[4, 2, 1]]
7  print(rows)
```

Console | PyUnit

```
<terminated> Practice.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
  UserName Password   Expected_result     Result
4    user2   user23  actiTIME - Login  TEST_PASS
2     user  manager  actiTIME - Login  TEST_PASS
1    admin  mnagher  actiTIME - Login  TEST_PASS
```

 # Operations on Columns:

data.iloc[:,0] # first column of data frame

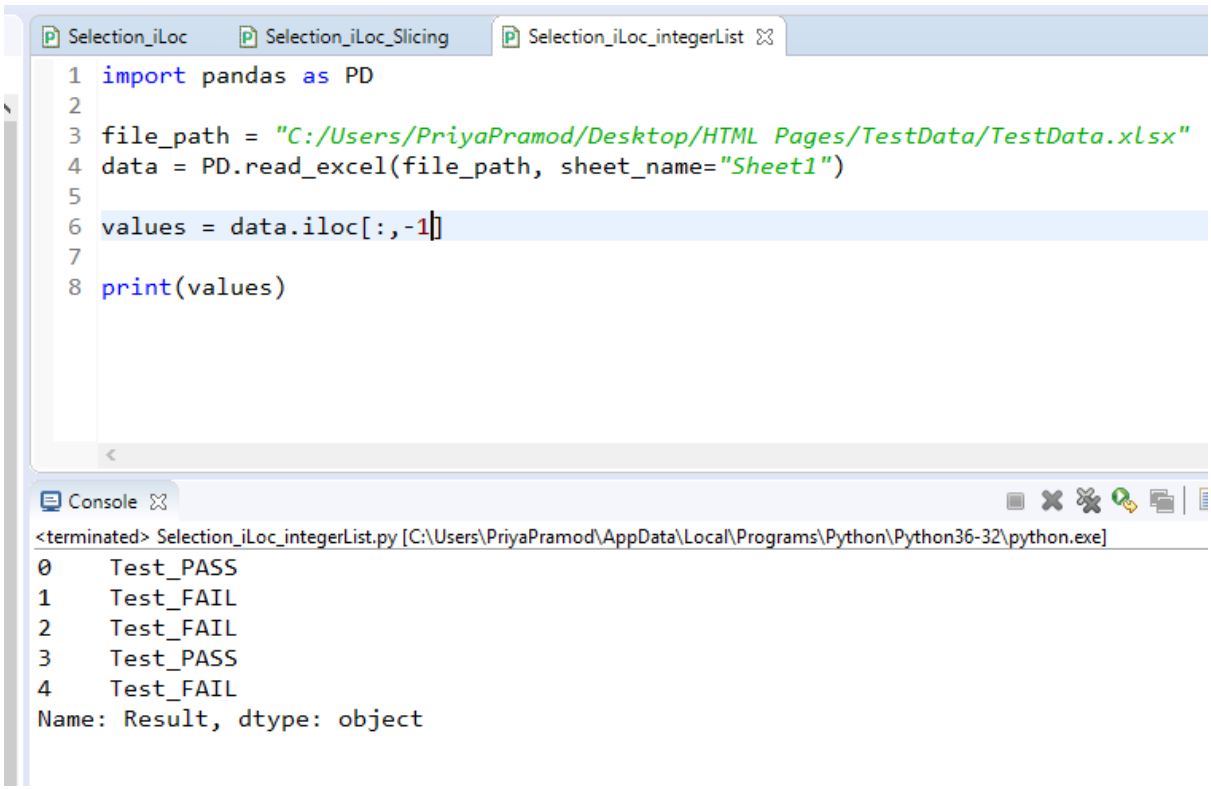
Selection_iLoc | Selection_iLoc_Slicing | Selection_iLoc_integerList

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/HTML Pages/TestData/TestData.xlsx"
4  data = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  values = data.iloc[:,0]
7
8  print(values)
```

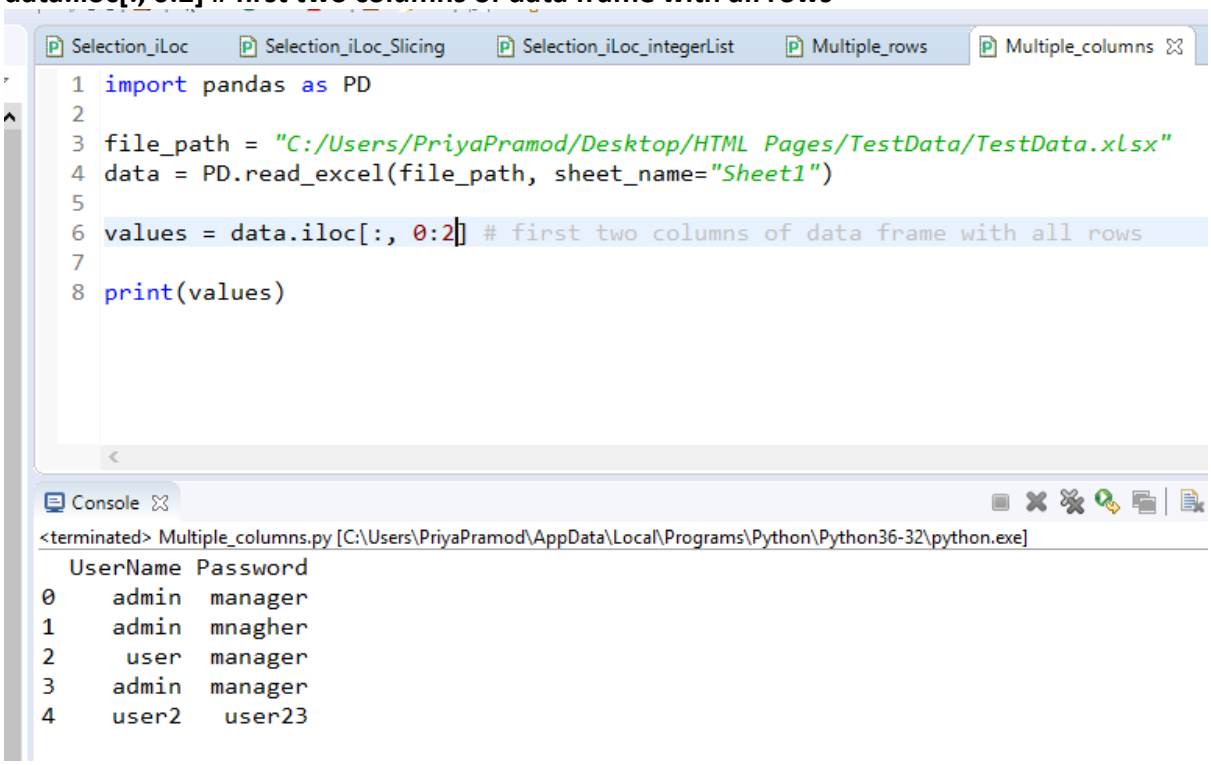
Console

```
<terminated> Selection_iLoc_integerList.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
0    admin
1    admin
2     user
3    admin
4    user2
Name: UserName, dtype: object
```

data.iloc [:,-1] # last column of data frame.



data.iloc[:, 0:2] # first two columns of data frame with all rows



Selecting the columns

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  rows = xl.iloc[:, [3,2,1]]
7  print(rows)
```

Console PyUnit

```
<terminated> Practice.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
      Result                Expected_result Password
0  TEST_PASS  actiTIME - Enter Time-Track  manager
1  TEST_PASS              actiTIME - Login  mnagher
2  TEST_PASS              actiTIME - Login  manager
3  TEST_PASS              actiTIME - Login  manager
4  TEST_PASS              actiTIME - Login   user23
```

Multiple columns and rows can be selected together using the .iloc indexer.

Program to select first 3 rows and first 3 columns using iloc?

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  rows = xl.iloc[0:3, 0:3]
7  print(rows)
```

Console PyUnit

```
<terminated> Practice.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
  UserName Password              Expected_result
0    admin  manager  actiTIME - Enter Time-Track
1    admin  mnagher              actiTIME - Login
2     user  manager              actiTIME - Login
```

Integer list of rows and columns:

We can also print the list of rows and columns using the below syntax:

Integer list

data.iloc [[1, 2], [3, 2]] & data.iloc [[1, 2, 3], [3, 2, 1]]

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  rows = xl.iloc[[0,3,1], [1, 2]]
7  print(rows)
```

Console PyUnit

```
<terminated> Practice.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
  Password              Expected_result
0  manager  actiTIME - Enter Time-Track
3  manager             actiTIME - Login
1  mnagher             actiTIME - Login
```

Program to print all the data in excel using the iloc?

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  xl = PD.read_excel(file_path, sheet_name="Sheet1")
5
6  s = xl.shape
7  rows = s[0]
8  columns = s[1]
9
10 for i in range(0, rows):
11     for j in range(0, columns):
12         value = xl.iloc[i, j]
13         print(value, end=" ")
14     print(" ")
```

Console PyUnit

```
<terminated> coloumn_selection.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
admin manager actiTIME - Enter Time-Track TEST_PASS
admin mnagher actiTIME - Login TEST_PASS
user manager actiTIME - Login TEST_PASS
admin manager actiTIME - Login TEST_PASS
user2 user23 actiTIME - Login TEST_PASS
```

LOC:

- The Pandas loc indexer can be used for selecting by label/index.
- The loc indexer is used with the same syntax as iloc:

data.loc [<row selection>, <column selection>]

Label-based / Index-based indexing using .loc

- We can directly select rows for different "column name" values using
  data.loc [RowName/RowNumber, ColumnName]

Consider the following spread sheet:

TestCaseID	UserName	Password	Expected_result	Result
TC001	admin	manager	actiTIME - Enter Time-Track	PASS
TC002	admin	mnagher	actiTIME - Login	PASS
TC003	user	manager	actiTIME - Login	PASS
TC004	admin	ghzxgdkjas	actiTIME - Login	PASS
TC005	user2	user23	actiTIME - Login	PASS

Note: Column name and row name are in bold and in blue.

Now, if you want to read the rows using the row name, then we can read the rows using the row name by using the loc method.

IMP:

1. Before reading the data using "Row name", we need to make a column as "Index column", otherwise you will get "Key Error". Since, by default pandas will not consider the first column as index column, programmer has to explicitly mention the column name as a row index.
2. To make any row as an index column, we will use "index_col" argument in read_excel method.
   Syntax:
   ```
   pd.read_excel(file_path, sheet_name="SheetName", index_col="ColumnName")
   ```
3. In this example programs, I will be considering the "**TestCaseID**" as an index column.
4. After setting any column as index column, we can the read the data using the column name

Program to read the data using the name of the row?

```
import pandas as pd

file_path = "D:/Data/TestData.xlsx"

sheet = pd.read_excel(file_path, sheet_name="Login", index_col="TestCaseID")
value = sheet.loc["TC001"]
print(value)
```

Output:

```
C:\Users\Jayapriyapramod\AppData\Local\Programs\
UserName                                    admin
Password                                  manager
Expected_result    actiTIME - Enter Time-Track
Result                                       PASS
Name: TC001, dtype: object

Process finished with exit code 0
```

Program to read multiple rows using the row names?

```
import pandas as pd

file_path = "D:/Data/TestData.xlsx"

sheet = pd.read_excel(file_path, sheet_name="Login", index_col="TestCaseID")
value = sheet.loc[["TC001", "TC002", "TC003"]]
print(value)
```

Output:

```
C:\Users\Jayapriyapramod\AppData\Local\Programs\Python\Python37-32\
           UserName Password              Expected_result Result
TestCaseID
TC001         admin  manager  actiTIME - Enter Time-Track   PASS
TC002         admin  mnagher             actiTIME - Login   PASS
TC003          user  manager             actiTIME - Login   PASS
```

Program to read the single column using the column name?

```
import pandas as pd

file_path = "D:/Data/TestData.xlsx"

sheet = pd.read_excel(file_path, sheet_name="Login", index_col="TestCaseID")
value = sheet.loc[:, "UserName"]
print(value)
```

Output:

```
C:\Users\Jayapriyapramod\AppData\Lc
TestCaseID
TC001     admin
TC002     admin
TC003      user
TC004     admin
TC005     user2
Name: UserName, dtype: object

Process finished with exit code 0
```

Program to read the multiple columns using the column names?

```
import pandas as pd

file_path = "D:/Data/TestData.xlsx"

sheet = pd.read_excel(file_path, sheet_name="Login", index_col="TestCaseID")
value = sheet.loc[:, ["UserName", "Password"]]
print(value)
```

Output:

```
C:\Users\Jayapriyapramod\AppData\Loc
            UserName    Password
TestCaseID
TC001          admin     manager
TC002          admin     mnagher
TC003           user     manager
TC004          admin  ghzxgdkjas
TC005          user2       user23

Process finished with exit code 0
```

Program to read the value using row name and column name?

```
import pandas as pd

file_path = "D:/Data/TestData.xlsx"

sheet = pd.read_excel(file_path, sheet_name="Login", index_col="TestCaseID")
value = sheet.loc["TC001", "UserName"]
print(value)
```

Note:

- Most of the cases, we will having the column name but we will be not having the row name, in those cases, you can read the values using the column name only, then you need to mention the row number instead of row name.

Example Sheet for the above case.

A	B	C
Customer	**Project**	**Task**
Architects Bureau	One-page web site	Page design
Boston Chocolate	Web site maintenance	HTML/CSS
Media Agency	Web site maintenance	Updating content

In the above case, we don't have a row names. Than below program is the solution for the above problem.

Program to read the values using row number and column name?

```
1 import pandas as PD
2
3 file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4 data = PD.read_excel(file_path, sheet_name="Sheet2")
5
6 row_values = data.loc[0, "Customer"]
7 print(row_values)
```

```
Console  PyUnit
<terminated> Selection_Loc_Label_Index_Bases.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32
Architects Bureau
```

Writing the data to the excel

- To write data to the excel spread sheet, pandas use “OpenPyXL” as an engine. Make sure you have installed “OpenPyXL”.

In order to write the data to the excel cell,

1. First identify the cell to which you want to write the data & assign the value to the cell by identifying the row and column.
   Example:
   2nd row, 4th column = value or xl.iloc[2, 4] = value or xl.loc[2, “Password”] = value
2. Create an object of “**ExcelWriter**” by passing the path of the file as an argument & assign the object to a variable.
   ```
   writer = PD.ExcelWriter(file_path)
   ```
3. Call “to_excel” method, which takes two argument, 1st object of “**ExcelWriter**” and 2nd “**SheetName**”
   ```
   df1.to_excel(writer, sheet_name='Sheet2')
   ```
4. Call the method “**Save**” to save the file.
   ```
   writer.save()
   ```

Write a program to write the data to excel?

```
import pandas as PD

print("Program starts")
file_path = "C:/Users/PriyaPramod/Desktop/TestData/Test.xlsx"

df1 = PD.DataFrame({'Data1': [10, 20, 30, 20, 15, 30, 45]})

# Create a Pandas Excel writer
writer = PD.ExcelWriter(file_path)

df1.to_excel(writer, sheet_name='Sheet1')

# Close the Pandas Excel writer and output the Excel file.
writer.save()
print("Program ends")

```

Note:

1. Above program works perfectly without any issues, but there is a problem in the above program.
2. Problem is, the pandas while writing the data to the excel file. Always overwrite the excel spreadsheet and as well delete other spreadsheets in the excel file.
3. In order to over come from the above problem, we will initialize the ExcelWriter object by loading all the spreadsheets using the OpenPyXL package.

Below is the program to write the data to the excel spreadsheet without deleting the existing spreadsheet and not overwriting.

```
import pandas as pd
from openpyxl import load_workbook

file_path = "D:/Data/writer.xlsx"
sheet = pd.read_excel(file_path, sheet_name="Sheet1")
sheet.iloc[1, 0] = "Python Selenium"

writer = pd.ExcelWriter(file_path)

# Loads all the spreadsheets to the variable
workbook = load_workbook(file_path)
# Initialising the the workbook object to book variable in ExcelWriter class
writer.book = workbook
# Initialising all the spreadsheet of the current workbook to sheets variable of ExcelWriter class
writer.sheets = dict((ws.title, ws) for ws in workbook.worksheets)

sheet.to_excel(writer, sheet_name="Sheet1", index=False)
writer.save()
writer.close()
```

Data Manipulation Operation on Excel using Pandas

Renaming the all columns
Example spread sheet:

Customer	Project	Task
Architects Bureau	One-page web site	Page design
Boston Chocolate	Web site maintenance	HTML/CSS
Media Agency	Web site maintenance	Updating content

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  data = PD.read_excel(file_path, sheet_name="Sheet2")
5
6  print("Column names Before renaming the columns")
7  print(data.columns)
8  print("----------------------------")
9  print("Column namesAfter renaming the columns")
10 data.columns = ["Qspider", "Jspiders", "Psipders"]
11 print(data.columns)
12
```

Console PyUnit

```
<terminated> column_rename_all_columns.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\
Column names Before renaming the columns
Index(['Customer', 'Project', 'Task'], dtype='object')
----------------------------
Column namesAfter renaming the columns
Index(['Qspider', 'Jspiders', 'Psipders'], dtype='object')
```

Renaming the single column

Syntax:

df.rename(columns={'old_name': 'new_name'})

```
1  import pandas as PD
2
3  file_path = "C:/Users/PriyaPramod/Desktop/TestData/TestData.xlsx"
4  data = PD.read_excel(file_path, sheet_name="Sheet2")
5
6  print("Before renaming the Customer column")
7  print(data.columns)
8  print("----------------------------")
9  print("After renaming the Customer column")
10 renamedColumns = data.rename(columns={'Customer': 'Qspiders'})
11 print(renamedColumns.columns)
12
```

Console PyUnit

```
<terminated> column_rename_all_columns.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
Before renaming the Customer column
Index(['Customer', 'Project', 'Task'], dtype='object')
----------------------------
After renaming the Customer column
Index(['Qspiders', 'Project', 'Task'], dtype='object')
```

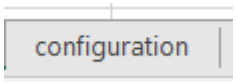
Data driven testing in selenium.

Step 1: create a excel file with the name “Testdata.xlsx.

Step2: Create a excel spreadsheet with the name "Login", and add the below data into that file.

TestCaseID	UserName	Password	Expected_result	Result
TC001	admin	manager	actiTIME - Enter Time-Track	PASS
TC002	admin	mnagher	actiTIME - Login	PASS
TC003	user	manager	actiTIME - Login	PASS
TC004	admin	ghzxgdkjas	actiTIME - Login	PASS
TC005	user2	user23	actiTIME - Login	PASS

Step 3: Create another excel spreadsheet with the name "configuration", and add the below data into that file.



Browser	URL
Chrome	https://demo.actitime.com/login.do

Step 4: Scenario:

a) Test the ActiTime login page with multiple set of user credentials.

Note: Run the script based on the number of rows present in the excel.

Step 5: Create the following folder structure in PyCharm.

Example:

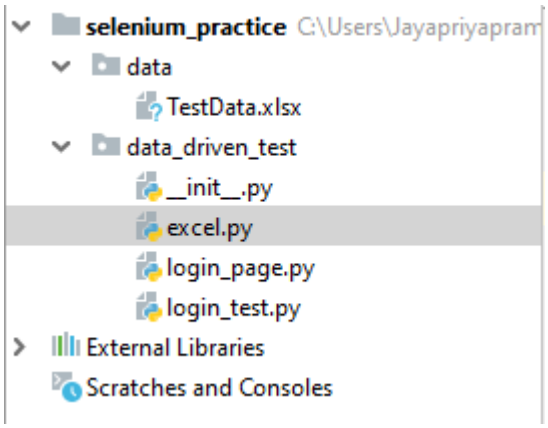
Project Name: selenium_practice

Package Name: data_driven_test

Python Files: under the data_driven_test package create 3 python files, namely "excel.py", "login_page.py" and "login_test.py"

Create a Directory "Data" to keep the excel file.

Project Structure should look like below,



Note: below are the reason for creating the above 3 files,

1. "excel.py" file is for developing the generic methods reading the data and writing the data to the excel file.
2. "login_page.py" file is for developing the reusable method to perform the actions like, launching the browser, closing the browser, entering the username and password values to the text boxes and validating the titles.
3. "login_test.py" file is for testing the login feature of antitime by calling the reusable functions developed in excel.py and login_page.py files.

In excel.py file develop reusable methods to get the data, write the data and get the total number rows from the application.

excel.py

```
1  import pandas as pd
2  from openpyxl import load_workbook
3
4
5  # To get the object of DataFrame
6  def get_sheet(file_path, sheet):
7      data_frame = pd.read_excel(file_path, sheet_name=sheet)
8      return data_frame
9
10
```

```
# To get the total number of rows from the excel spreadsheet
def get_row_number(file_path, sheet):
    data_frame = get_sheet(file_path, sheet)
    total_row = data_frame.shape[0]
    return total_row


# To get the data from the excel spreadsheet from the specific cell
def get_data(file_path, sheet, row_number, col_name):
    data_frame = get_sheet(file_path, sheet)
    value = data_frame.loc[row_number, col_name]
    return value


# To write the data to the specific cell in the excel spreadsheet
def write_to_excel(file_path, sheet, row_number, col_name, value_to_write):
    data_frame = get_sheet(file_path, sheet)
    data_frame.loc[row_number, col_name] = value_to_write
    writer = pd.ExcelWriter(file_path)

    workbook = load_workbook(file_path)
    writer.book = workbook
    writer.sheets = dict((ws.title, ws) for ws in workbook.worksheets)

    data_frame.to_excel(writer, sheet_name=sheet, index=False)
    writer.save()
    writer.close()
```

In login_page.py file develop reusable functions to launch the browser, close the browser, login to ActiTime and verifying the title.

login_page.py

```
from selenium import webdriver
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as ec


class LoginPage:

    # declaring the local class variable and initialising the variable, so that
    # we can use the same driver variable across different functions.
    def __init__(self):
        self.driver = None
```

```
    # this function will launch the browser, maximises the browser window, enters the url
    # and finally sets the implicit wait time.
    def launch_browser(self, browser_name, url):
        if browser_name == "Chrome":
            self.driver = webdriver.Chrome(executable_path=
                                           "C:\\Drivers\\chromedriver")
        else:
            print("Please enter the valid browser name")

        self.driver.maximize_window()
        self.driver.get(url)
        self.driver.implicitly_wait(30)

    # this function enters the username, password and clicks on the login button
    def login_to_application(self, username, password):
        self.driver.find_element(By.ID, "username").send_keys(username)
        self.driver.find_element(By.NAME, "pwd").send_keys(password)
        self.driver.find_element(By.ID, "loginButton").click()
```

```
    # this function ends the browser sessions
    def close_browser(self):
        self.driver.quit()

    # this function verifies the expected title with the actual title and returns boolean value
    def verify_title(self, expected_title):
        flag = False
        try:
            wait = WebDriverWait(self.driver, 15)
            flag = wait.until(ec.title_contains(expected_title))
            return flag
        except TimeoutException:
            print(expected_title + " is not loaded")
            return flag
```

In test login module, we call the reusable functions of excel.py and login_page.py file to test the ActiTime login page by passing the multiple set of data by reading it from the excel spreadsheet and write the result back to the excel spreadsheet.

login_test.py

```
from data_driven_test import excel
from data_driven_test.login_page import LoginPage

# Specifying the path of excel spreadsheet to file_path variable
file_path = "C:/Users/Jayapriyapramod/Desktop/selenium_practice/data/TestData.xlsx"

# Getting the total number of rows from excel, so that we can run the script multiple times
# based on the number of rows present in the excel spreadsheet.
rows = excel.get_row_number(file_path, "Login")

# Fetching the browser name and url from "configuration" spreadsheet
browser = excel.get_data(file_path, "configuration", 0, "Browser")
url = excel.get_data(file_path, "configuration", 0, "URL")
```

```
14
15  # Iterating multiple times based on the number of test data rows present in "Login" spreadsheet
16  # to test the login page of ActiTime application.
17  for i in range(0, rows):
18      # Getting the username, password & expected results data from the each row
19      user = excel.get_data(file_path, "Login", i, "UserName")
20      pass_word = excel.get_data(file_path, "Login", i, "Password")
21      expected_result = excel.get_data(file_path, "Login", i, "Expected_result")
22
23      # Entering the test data to ActiTime application
24      login = LoginPage()
25      login.launch_browser(browser, url)
26      login.login_to_application(user, pass_word)
27
28      # Validating whether the actual results is coming as expected and storing the status in the
29      # flag variable
30      flag = login.verify_title(expected_result)
31
```

```
32      # If actual and expected are matching, flag will be True else False.
33      # If flag is True, writing "PASS" to "Login" spreadsheet else writing "FAIL"
34      if flag:
35          excel.write_to_excel(file_path, "Login", i, "Result", "PASS")
36      else:
37          excel.write_to_excel(file_path, "Login", i, "Result", "FAIL")
38
39      # Finally ending the session by closing the browser.
40      login.close_browser()
```

Best Practices for Spreadsheet Data

Previous to reading in your spreadsheet in Python, you also want to consider adjusting your file to meet some basic principles, such as:

- The first row of the spreadsheet is usually reserved for the header, while the first column is used to identify the sampling unit;
- Avoid names, values or fields with blank spaces. Otherwise, each word will be interpreted as a separate variable, resulting in errors that are related to the number of elements per line in your data set. Consider using:
    - ✓ Underscores,

- ✓ Dashes,
- ✓ Camel case, where the first letter of each section of text is capitalized, or
- ✓ Concatenating words

- ➢ Short names are preferred over longer names;
- ➢ Try to avoid using names that contain symbols such as ?, $,%, ^, &, *, (,),-,#, ?,,,<,>, /, |, \, [,] ,{, and };
- ➢ Delete any comments that you have made in your file to avoid extra columns or NA's to be added to your file; and
- ➢ Make sure that any missing values in your data set are indicated with NA.