accenture technology

Learning and Knowledge Management

Module 6: openpyxl for MS Excel

Module Objectives

At the end of this module, you will be able to:

- Explain installing and configuring openpyxl in python project
- Illustrate processing excel documents
- Describe extracting rows and columns
- Demonstrate extracting data from excel sheet using openpyxl and integrate with selenium python API

Topic List

- Overview of openpyxl
- Configure openpyxl in Python Project
- Processing Excel Documents
- Extracting Rows and Columns
- Integrating openpyxl with Selenium Python API

Topic List

- **Overview of openpyxl**
- Configure openpyxl in Python Project
- Processing Excel Documents
- Extracting Rows and Columns
- Integrating openpyxl with Selenium Python API

Overview of openpyxl

Overview

- openpyxl is a comprehensive library to read and write MS Excel files in python
- openpyxl is a module in python often used to handle excel files.
- To do
  - Read data from excel
  - Write data or draw some charts
  - Accessing sheets
  - Renaming / Adding / Deleting sheets
  - Formatting and styling in sheets or any other task

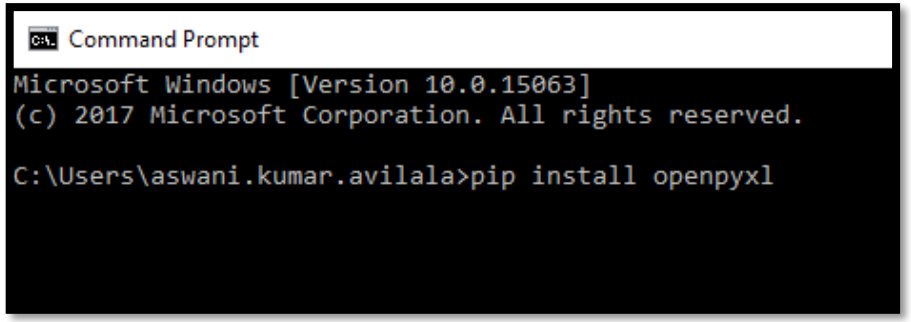
Topic List

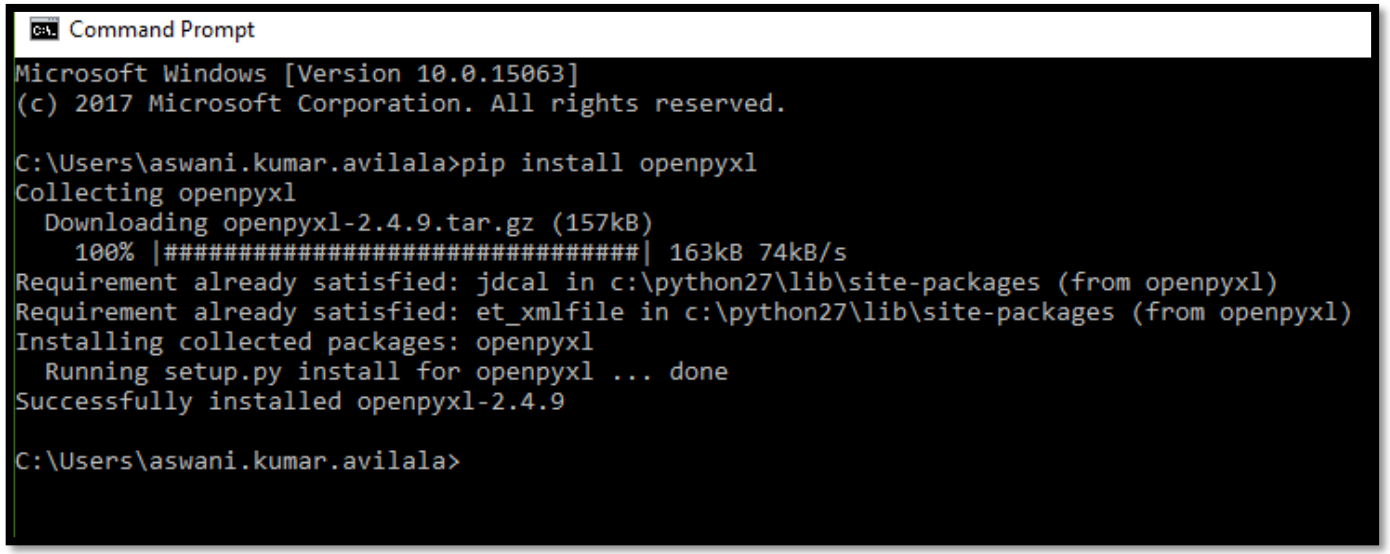
- Overview of openpyxl
- **Configure openpyxl in Python Project**
- Processing Excel Documents
- Extracting Rows and Columns
- Integrating openpyxl with Selenium Python API

Configure openpyxl in Python Project (1)

Step 1 : Install openpyxl using pip

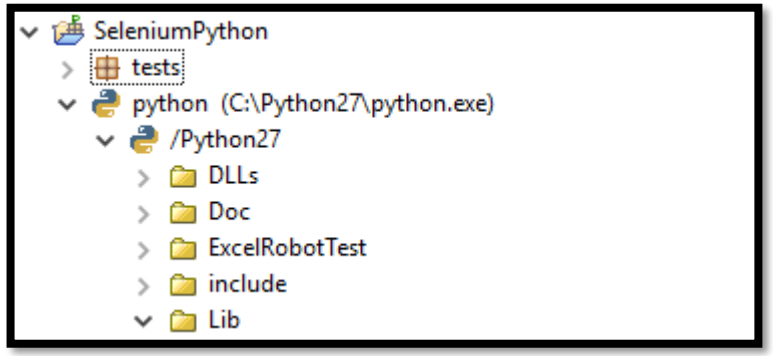
- Open command prompt , type ***pip install openpyxl*** and hit enter key

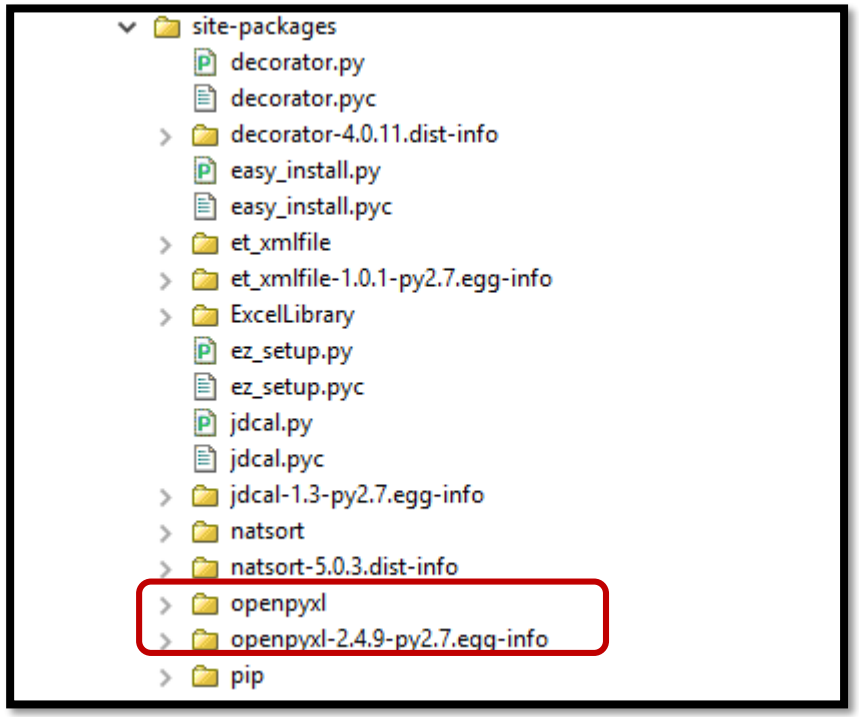




Configure openpyxl in Python Project (2)

Step 2 : Create a python project and observe the addition of library in the python directory





Topic List

- Overview of openpyxl
- Configure openpyxl in Python Project
- **Processing Excel Documents**
- Extracting Rows and Columns
- Integrating openpyxl with Selenium Python API

Processing Excel Documents (1)

Basics of Python Excel

- Spreadsheet, the excel file is called **Workbook** in openpyxl
  - A Workbook is usually saved as a file with extension .xls / .xlsx
  - A Workbook can have one or more work sheets
- The worksheet which a user is viewing or viewed before closing the file is called **Active sheet**
- Each sheet consists of
  - Vertical columns, known as **Column** starting from A.
  - Horizontal rows, called as **Row,** numbering starts from 1
- **Cell** is a box that intersects a Row and a column. Each cell has specific address in reference to Row and Column. The cell may contain number, formula or text.
- The grid of cells make the work area or worksheet in excel

Processing Excel Documents (2)

- Opening Excel files in Python
  - Accessing sheets from the loaded workbook

```
import unittest
import openpyxl
import os

class Test(unittest.TestCase):


    def testName(self):
        #To get the current working directory [Optional]
        path=os.getcwd()
        print(path)
        wb=openpyxl.load_workbook('data/Book1.xlsx')
        sheetnames=wb.get_sheet_names();
        print(sheetnames)
        pass


if __name__ == "__main__":
    #import sys;sys.argv = ['', 'Test.testName']
    unittest.main()
```

Processing Excel Documents (3)

Accessing a specific sheet from the loaded workbook

```
def testName(self):
    #To get the current working directory [Optional]
    path=os.getcwd()
    print(path)
    wb=openpyxl.load_workbook('data/Book1.xlsx')
    sheet1=wb.get_sheet_by_name('Sheet1')
    type(sheet1)
    print(sheet1.title)
    pass
```


Topic List

- Overview of openpyxl
- Configure openpyxl in Python Project
- Processing Excel Documents
- **Extracting Rows and Columns**
- Integrating openpyxl with Selenium Python API

Extracting Rows and Columns (1)

Accessing data in Cells from Worksheet

- For accessing data from sheet cells we refer by sheet and then the cell address.

```
def testName(self):
    #To get the current working directory [Optional]
    path=os.getcwd()
    print(path)
    wb=openpyxl.load_workbook('data/Book1.xlsx')
    sheet1=wb.get_sheet_by_name('Sheet1')
    cellvalue=sheet1['A3'].value
    print(cellvalue)
    pass
```

- Another way of accessing cell data is like

```
def testName(self):
    #To get the current working directory [Optional]
    path=os.getcwd()
    print(path)
    wb=openpyxl.load_workbook('data/Book1.xlsx')
    sheet1=wb.get_sheet_by_name('Sheet1')
    c=sheet1['A3']
    print(c.value,c.row,c.column)
    pass
```

Extracting Rows and Columns (2)

Accessing data in Cells from Worksheet

- Getting data from cells with the help of rows and columns:

```
def testName(self):
    #To get the current working directory [Optional]
    path=os.getcwd()
    print(path)
    wb=openpyxl.load_workbook('data/Book1.xlsx')
    sheet1=wb.get_sheet_by_name('Sheet1')
    cellvalue=sheet1.cell(row=3,column=1).value
    print(cellvalue)
    pass
```

- Instead of getting one value from a column, now we print all rows and columns

```
def testName(self):
    #To get the current working directory [Optional]
    path=os.getcwd()
    print(path)
    wb=openpyxl.load_workbook('data/Book1.xlsx')
    sheet1=wb.get_sheet_by_name('Sheet1')
    for x in range (2,6):
        print(x,sheet1.cell(row=x,column=1).value,
              sheet1.cell(row=x,column=2).value)
    pass
```


Extracting Rows and Columns (3)

Accessing data in Cells from Worksheet

- Extraction of rows and columns from sheet by slicing .

```
def testName(self):
    #To get the current working directory [Optional]
    path=os.getcwd()
    print(path)
    wb=openpyxl.load_workbook('data/Book1.xlsx')
    sheet1=wb.get_sheet_by_name('Sheet1')
    cells=sheet1['A2':'B5']
    for c1,c2 in cells:
        print(c1.value,c2.value)
    pass
```

Topic List

- Overview of openpyxl
- Configure openpyxl in Python Project
- Processing Excel Documents
- Extracting Rows and Columns
- **Integrating openpyxl with Selenium Python API**

Integrating openpyxl with Selenium Python API (1)

Step 1:

- Create a test method in unittest class which creates the instance of the webdriver. The test method in turn calls the readExcel method

```
import unittest
import openpyxl
import os
from selenium import webdriver

class Test(unittest.TestCase):

    def testMercuryTours(self):
        global driver
        driver=webdriver.Chrome();
        driver.get("http://newtours.demoaut.com");
        driver.implicitly_wait(10)
        driver.maximize_window();
        self.readExcel()
        driver.close()

    def login(self,username,password):

    def readExcel(self):

if __name__ == "__main__":
    #import sys;sys.argv = ['', 'Test.testName']
    unittest.main()
```

Integrating openpyxl with Selenium Python API (2)

Step 2:

- Create a login method with username and password as parameters. The login method interacts with the browser objects and send the data to web application

```
import unittest
import openpyxl
import os
from selenium import webdriver
class Test(unittest.TestCase):

    def testMercuryTours(self):

    def login(self,username,password):
        driver.find_element_by_name("userName").send_keys(username);
        driver.find_element_by_name("password").send_keys(password);
        driver.find_element_by_name("login").click()
        title=driver.title
        if title=='Find a Flight: Mercury Tours:' :
            self.assertEqual(driver.title,"Find a Flight: Mercury Tours:")
            driver.find_element_by_link_text("SIGN-OFF").click()

    def readExcel(self):

if __name__ == "__main__":
    #import sys;sys.argv = ['', 'Test.testName']
    unittest.main()
```

Integrating openpyxl with Selenium Python API (3)

Step 3:

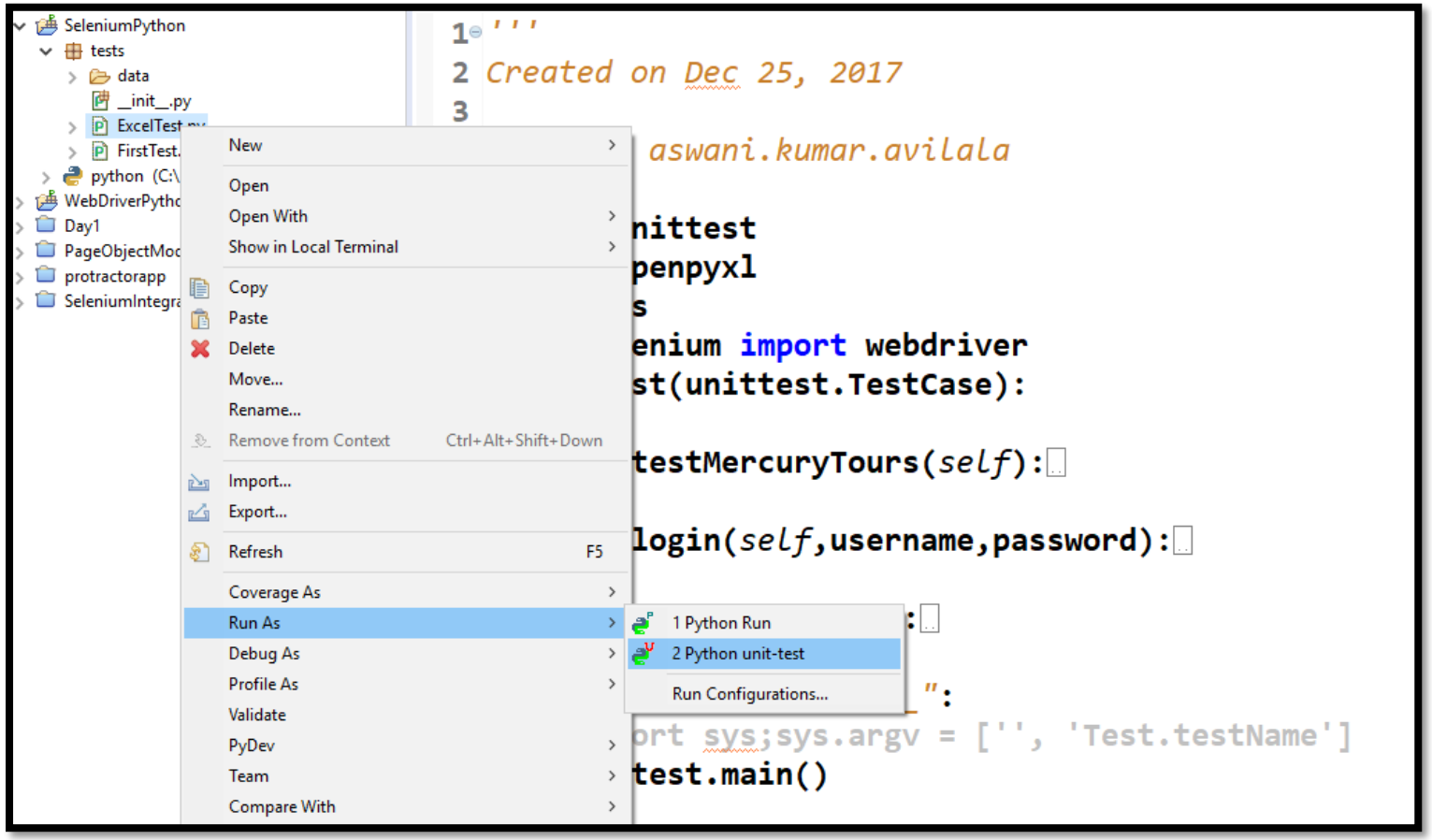
- Create readExcel method which reads the data as rows and columns from excel. The readExcel method in turn calls the login method based on number of rows.

```
import unittest
import openpyxl
import os
from selenium import webdriver
class Test(unittest.TestCase):

    def testMercuryTours(self):

    def login(self,username,password):

    def readExcel(self):
        #To get the current working directory [Optional]
        path=os.getcwd()
        print(path)
        wb=openpyxl.load_workbook('data/Book1.xlsx')
        sheet1=wb.get_sheet_by_name('Sheet1')
        cells=sheet1['A2':'B5']
        for c1,c2 in cells:
            self.login(c1.value,c2.value)

if __name__ == "__main__":
    #import sys;sys.argv = ['', 'Test.testName']
    unittest.main()
```

Integrating openpyxl with Selenium Python API (4)

Step 4:

- Execute the test as unittest.



Exercise 6.1: Reading Data from Excel

Read the data from excel and send to TestMeApp login page

To Do

1. Create a test function
2. Create a separate login function
3. Create a seperate readExcel function
4. Run the script as python unit test

Refer Exercise 6.1 in Selenium with Python_wbk.doc for the detailed steps.

Module Summary

Now, you should be able to:

- Explain installing and configuring openpyxl in python project
- Illustrate processing excel documents
- Describe extracting rows and columns
- Demonstrate extracting data from excel sheet using openpyxl and integrate with selenium python API

Thank You