**Working with Excel Spreadsheets:**

1. Excel is a popular and powerful spreadsheet application for Windows.
2. The openpyxl module allows your Python programs to read and modify Excel spreadsheet files.
3. OpenOffice Calc work with Excel’s *.xlsx* file format for spreadsheets.

For example, you might have the boring task of copying certain data from one spreadsheet and pasting it into another one. Or you might have to go through thousands of rows and pick out just a handful of them to make small edits based on some criteria. Or you might have to look through hundreds of spreadsheets of department budgets, searching for any that are in the red. These are exactly the sort of boring, mindless spreadsheet tasks that Python can do for you.

**Excel Documents:**

1. An Excel spreadsheet document is called a workbook.
2. Each workbook can contain multiple sheets (also called worksheets).
3. The sheet the user is currently viewing (or last viewed before closing Excel) is called the active sheet.
4. Each sheet has columns (addressed by letters starting at A) and rows (addressed by numbers starting at 1).
5. A box at a particular column and row is called a cell.

**Reading Excel Documents:**

1. create the spreadsheet. (Which contains one/many sheets)
2. add some data to it
3. now let’s see how we can manipulate the sheets or workbook using openpyxl module in python.
4. Opening excel document with openpyxl.
5. Once u have imported openpyxl module u can use “**openpyxl.load\_workbook()”** function. (Data type workbook)
6. Remember that the file is in the format of .xlsx
7. Getting sheets from the workbook.
8. You can get a list of all the sheets names in the workbook by calling “**get\_sheet\_names()”** method. (Data type worksheet)
9. You can use “**get\_sheet\_by\_name()”** method to get a certain worksheet.
10. You can use “**.active”** method to get the active sheet.
11. Getting cells from the sheets.
12. You can access a cell object by its name.
13. Just mention the cell name inside the square brackets with in the single quotes
14. For example, sheet[‘A1’], sheet[‘B4’]
15. You can use **.value** method to access the value in the given cell
16. For example, sheet[‘A1’].value
17. You can use **”.cell”** method and using row and column to call a cell
18. Sheet.cell(row=1, column=2)
19. You can use **“max\_row() and max\_column() and min\_row() and min\_column()”** method to get the max and min number of rows and columns present in worksheet.
20. Converting between column letters and numbers
21. To convert firstly import **“from openpyxl.utils import get\_column\_letter, column\_index\_from\_string”** use get\_column\_letter() function for to convert from numbers to letters, column\_index\_from\_string() function to convert letters to numbers.
22. Getting rows and columns from the sheets
23. You can slice the worksheet to get all the cell object in a row , column and rectangular area of the spreadsheet.
24. To get rectangular area from ‘A1’ and ‘C3’, and we get a generator object containing the cell object in that area. To generate the objects we use tuple() on it to display the cell objects in a tuple.
25. To get the values in the area we use two for loops. The outer loop goes over each row and the nested for loop goes over cell in that row.
26. Tuple(sheet[‘A1’,’C3’])
27. For r in sheet[‘A1’,C3’]:
28. For c in r:
29. Print(c.coordinate , c.value)
30. To access the values of cell of a particular row or column we can use .colums[] method.
31. For c in sheet.column[1]
32. Print(c.value)
33. As a quick review, here’s a rundown of all the functions, methods, and datatypes involved in reading a cell out of a spreadsheet file.
    1. Import the openpyxl module.
    2. Call the openpyxl.load\_workbook() function.
    3. Get a workbook object
    4. Call the .active or get\_sheet\_by\_name() workbook method.
    5. Get a worksheet object
    6. Use indexing or cell() sheet method wit row and column keyword arguments.
    7. Get a cell object.
    8. Read the cell object’s value attribute.

**Reading data from a Spreadsheet:**

1. For example, we will use a excel sheet name censuspopdata.xlsx in which we have data from the 2010-US census and in the spreadsheet, we have date of census-tract, state, county and population.
2. So, in this project we will learn how to write a script that can read from the census spreadsheet file and calculate statistics (like we will sum up) from each county with seconds.
3. This what our program will do.
   1. Reads the data from the excel spreadsheet.
   2. Counts the number of census tract in each county
   3. Counts the total population of each county
   4. Then prints the result.
4. This means your code will need to do the following.
5. Open and read the cells of an excel doc with openpyxl module.
6. Calculate all the tract and populations data and store it in a data structure.
7. Write the data structure to a text file with .py extension using the pprint module.
8. Step 1: **read the spreadsheet data**
9. There is just one sheet in the censuspopdata.xlsx spreadsheet, named 'Population by Census Tract', and each row holds the data for a single census tract. The columns are the tract number (A), the state abbreviation (B), the county name (C), and the population of the tract (D). Open a new file editor window and enter the following code. Save the file as readCensusExcel.py.
10. You also use pprint module that we will use to print the final data
11. Before u can store anything in it, you should determine exactly how you will structure the data inside it.
12. Import openpyxl, pprint
13. Wb=openpyxl.load\_workbook(‘censuspopdata.xlsx’)
14. Sheet =wb.get\_sheet\_by\_name(‘Population by Census Tract’)
15. Countdata={}
16. For row in range(2,sheet.max\_row()+1):
17. State=sheet[‘B’ + str(row)].value
18. County=sheet[‘C’+str(row)].value
19. Pop = sheet [‘D’+str(row)].value
20. Step 2: **populate the data structure**
21. The data structure stored in countdata will be a dictionary with state abbreviation as its keys, whose keys are string of county, each county name will in turn into a dictionary with just two keys 'tracts’ and ‘pop’.
22. countyData.setdefault(state,{})
23. countyData[state].setdefault(county,{‘tracts’:0,’pop’:0})
24. countyData[state][county][‘tracts’]+=1
25. countyData[state][county][‘pop’]+=1
26. Step3: **writing the result to a file**
27. Print(‘writing result….’)
28. resultFile=open(‘census2010.py’,’w’)
29. resultFile.write(‘allData=’+ pprint.pformat(countyData))
30. resultFile.close()
31. print(‘Done..’)

**Writing Excel Documents:**

1. openpyxl also provides ways of writing data, meaning that your programs can create and edit spreadsheet files.
2. **Creating and saving excel documents:**
3. Call the **‘openpyxl. Workbook()’** function to create a new, blank workbook object.
4. Import openpyxl
5. Wb=openpyxl.Workbook()
6. Wb.get \_sheet\_names()
7. Sheet=wb.active
8. Sheet.title
9. Sheet.title=”spam bacon eggs sheet”
10. Wb.get\_sheet\_names()
11. Any time you modify the workbook object or its sheets and cells, the spread sheet fill will not be save until we call the “save()” workbook method.
12. Import openpyxl
13. Wb.openpyxl.load\_workbook(‘example.xlsx’)
14. Sheet=wb.active
15. Sheet.title=’spam spam spam’
16. Wb.save(‘example\_copy.xlsx’)
17. don’t forget to save the file after every update or changes u have done in the code or table.
18. **Creating and removing sheets:**
19. Sheets can be added and removed from the workbook with the **‘create\_sheet()’** and **‘remove\_sheet()**’ methods .
20. Import openpyxl
21. Wb=openpyxl.workbook()
22. Wb.create\_sheet()
23. Wb.create\_sheet(index=0,title=’first sheet’)
24. Wb.create\_sheet(index=2,title=’middle sheet’)
25. Wb.get\_sheet\_names()
26. Output>>>>> [‘first sheet’, ’sheet’, ’middle sheet’, ‘sheet1’]
27. We have seen how we can add sheet and we have used index to assign a particular position.
28. Now we will see how to remove a sheet from workbook.
29. Wb.remove\_sheet(wb.get\_sheet\_by\_name(‘middle sheet’))
30. Wb.remove\_sheet(wb.get\_sheet\_by\_name(‘sheet1’))
31. Wb.get\_sheet\_names()
32. Output>>>> [‘first sheet’, ‘sheet’]
33. **Writing values to cells:**
34. Writing values to cell is much like writing values to keys in a dictionary.
35. Import openpyxl
36. Wb.openpyxl.wokbook()
37. Sheet=wb.get\_sheet\_by\_name(‘sheet)
38. Sheet[‘A1’]=’pruthvi’
39. Sheet[‘A1’].value
40. Output>>>>> ‘pruthvi’

**Updating a spreadsheet:**

1. In this project we will write a program to update cells in spreadsheet of produce sales.
2. Our program will look through the spreadsheet, find specific kinds of produce, and update their prices.
3. Your program does the following :
4. Loops over all the rows
5. If the row if for garlic, celery or lemons, change the price.
6. This means your code will need to do the following:
7. Open the spreadsheet file
8. For each row check whether the value in column A is garlic, celery or lemon.
9. If it is update the price in column B
10. Save the spreadsheet to a new file
11. Step 1: **Set up a data structure with the update information**
12. The prices that we need to update are
13. Celery = 1.19
14. Garlic= 3.07
15. Lemon= 1.27
16. You can write your code like
17. If produceName==’Celery’:
18. Cellobj=1.19
19. If produceName==’Garlic’:
20. Cellobj=3.07
21. If produceName==’Lemon’:
22. Cellobj=1.27
23. To skip the errors and changing values or adding a new produceName
24. It is better to create a dictionary of the price update
25. Import openpyxl
26. Wb.opepyxl.load\_workbook(‘produceSales.xlsx’)
27. Sheet=wb.get\_sheet\_by\_name(‘sheet’)
28. PRICE\_UPDATES={‘Garlic’: 3.07,’Celery’: 1.19,’Lemon’:1.27}
29. Step2: **check all rows and update incorrect prices:**
30. The next part will loop through the rows to find the produceName
31. For rownum in range(2, sheet.max\_rwo()):
32. produceName=sheet.cell(row=rownum, column=1).value
33. If produceName in PRICE\_UPDATES:
34. Sheet.cell(row=rownum, column=2).value=PRICE\_UPDATES[produceName]
35. Wb.sheet(‘updatedProduceSales.xlsx’)