1. **Project : Doctor Portal**

This application is related to social networking for doctors and medical students. Here doctors can upload the case studies/queries/articles related to the medicals. In the system also we integrated the polls and surveys.

**Domain:** Healthcare

**Tools & Environment:** Python, Django, JavaScript, and MySQL Server.

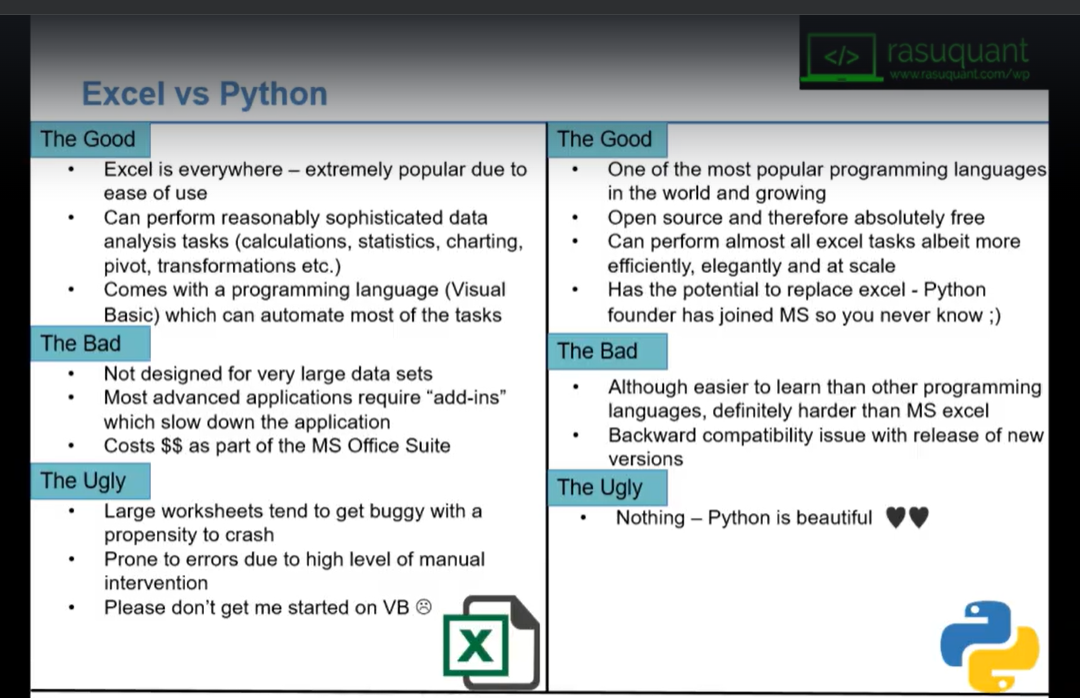
**Duration:** Nov 2015 to Dec 2017

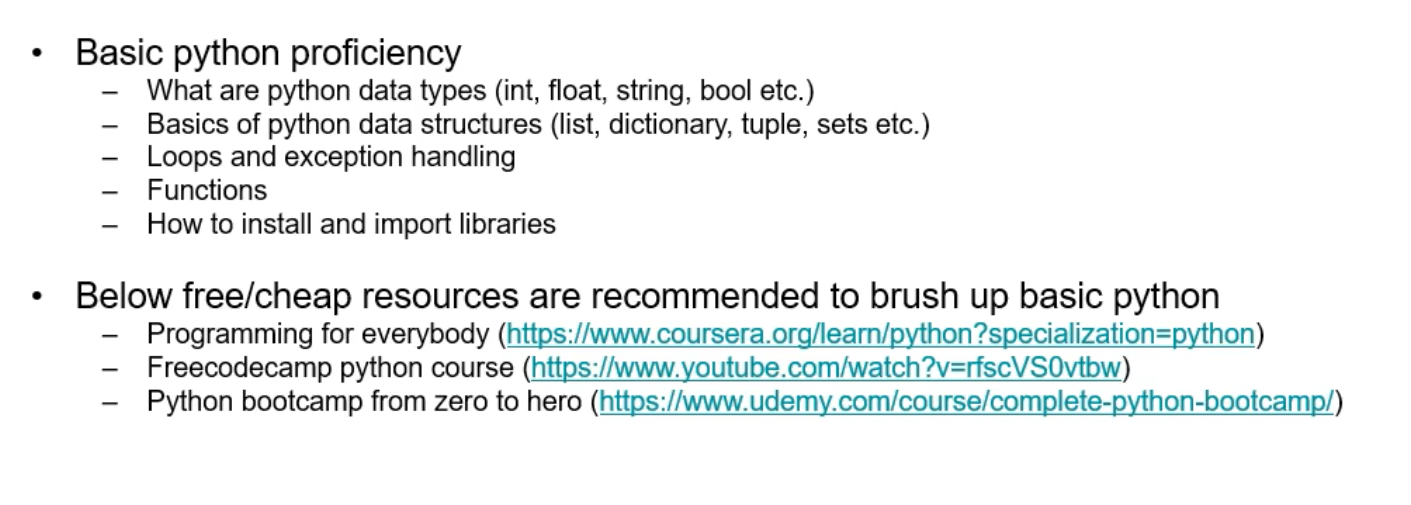
**Role & Responsibilities:** Requirement, Development, Deployment and Bug Fixing

<https://hexaware.udemy.com/course/excel-automation-using-python/>

**Description**

Excel spreadsheets are ubiquitous and no corporate job is possible without them. Like you, I have been working with them since I started my career. However, I rarely use excel now since I have automated most of my excel based tasks using python which has done wonders to my productivity and I want to help you do the same. I have created this course to help you automate your excel spreadsheets based tasks using python and improve your productivity manifold.





# -\*- coding: utf-8 -\*-

"""

Excel Automation - Numpy 1D Arrays

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import numpy as np

py\_list = [3,4,5] #python list

3\*py\_list #not what you want

np\_array = np.array(py\_list)

3\*np\_array #this is what we want - vectorized operation

np\_array\*np.array([2,4,6]) #vectortized multiplication

###############creating numpy 1D arrays###########################

np\_array = np.array([12,2,17,6,24])

zero\_array = np.zeros(5)

ones\_array = np.ones(5)

spaced\_array = np.linspace(10, 50, 30)

random\_array = np.random.randint(4, 12, 8)

py\_list = np\_array.tolist() #converting array back to list

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Numpy higher dimension arrays

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import numpy as np

###############creating higher dimension arrays#####################

np\_2d\_array = np.array([[3,4,5],

                        [1,8,9],

                        [2,6,16],

                        [4,3,1]])

np\_3d\_array = np.array([[[3,4,5],

                         [6,7,8]],

                        [[9,10,11],

                         [13,17,18]],

                        [[19,20,21],

                         [23,27,28]],

                        ])

##############manipulating array shape##########################

np.shape(np\_2d\_array)

np.shape(np\_3d\_array)

np.reshape(np\_2d\_array,(2,6))

np\_3d\_array.flatten()

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Numpy functions and operations

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import numpy as np

np\_array = np.array([12, 6, 34, -2, 7, 28])

np\_2d\_array = np.array([[3,14,5],

                        [11,8,19],

                        [2,6,16],

                        [1,13,7]])

###############important numpy functions#######################

np\_array.max()

np\_array.argmax()

np\_array.sort() #this sorts the array in-place so careful!

np.sort(np\_array) #this returns copy of the sorted array - safer option

np.sort(np\_array)[::-1] #hack to sort descending

np.sort(np\_2d\_array, axis = 1) #sorting 2-d array along axes

np\_array.mean()

np\_array.std() #standard deviation

###############numpy operations#######################

array1 = np.array([3,6,17,21])

array2 = np.array([-1,3,-5,1])

array1 + array2

array1 \* 5

array1 \*\* 3

array1 \*\* (1/2)

np.sqrt(array1)

np.log(array1)

array1 \* array2

array1 - array2

array1 / array2

array1.dot(array2)

# -\*- coding: utf-8 -\*-

"""

Excel Automation - numpy indexing

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import numpy as np

np\_array = np.array([14,2,4,8,-2,11,30])

np\_2d\_array = np.array([[3,7,10],

                        [12,2,1],

                        [-3,12,7]])

slice\_view = np\_array[1:4] #view of the subarray starting from index 1 all the way (not including) index 4

slice\_copy = np\_array[1:4].copy() #copy of the subarray starting from index 1 all the way (not including) index 4

slice\_view[1:4] = 5 #change elements with index 1, 2 and 3 of the original array

slice\_copy[1:4] = 5 #does not impact the original array

np\_array[np\_array > 5] #indexing using boolean array

np\_2d\_array[np\_2d\_array>5] #this will return a 1-d array

np\_2d\_array[:2,:2] #returns 2x2 array with rows 0,1 and columns 0,1 of original array

np\_2d\_array[[0,2],[1,2]] #returns 1-d array with elements 0,1 and 2,2 from the original array

np\_2d\_array[[[0],[2]],[1,2]] #returns 2x2 array with rows 0,2 and columns 0,1 of original array

 -\*- coding: utf-8 -\*-

"""

Excel Automation - sumif(s) and countif(s)

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

#countif

len(sales\_data[sales\_data["Quantity"] > 5])

sales\_data[sales\_data["Quantity"] > 5].shape[0]

#countifs

len(sales\_data[(sales\_data["State"]=="Kentucky") & (sales\_data["Quantity"] > 5)])

sales\_data[(sales\_data["State"]=="Kentucky") & (sales\_data["Quantity"] > 5)].shape[0]

#sumif

sales\_data[sales\_data["City"].str[:4]=="Fort"]["Profit"].sum()

sales\_data.loc[sales\_data["City"].str[:4]=="Fort","Profit"].sum()

#sumifs

sales\_data[(sales\_data["City"].str[:4]=="Fort") & (sales\_data["Quantity"] > 5)]["Profit"].sum()

sales\_data.loc[(sales\_data["City"].str[:4]=="Fort") & (sales\_data["Quantity"] > 5),"Profit"].sum()

# -\*- coding: utf-8 -\*-

"""

Excel Automation - string functions

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

#extract substring from a text column - left, right, mid

sales\_data["Order ID"].str[3:7] # mid function

sales\_data["Order ID"].str[:2] # left function

sales\_data["Order ID"].str[-2:] # right function

sales\_data["Order ID"].str.split("-").str[0]

#trim leading and trailing spaces

sales\_data["Order ID"] = sales\_data["Order ID"].str.strip()

#concatenate

sales\_data["location"] = sales\_data["State"]+"\_"+sales\_data["City"]

#capitalize

sales\_data["State"] = sales\_data["State"].str.upper()

sales\_data["State"] = sales\_data["State"].str.lower()

#find a string

sales\_data["City"].str.find("Fort")

# -\*- coding: utf-8 -\*-

"""

Excel Automation - If function

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

import numpy as np

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

sales\_data["Profit Net Tax"] = np.where(sales\_data["Category"]=="Furniture",0.8\*sales\_data["Profit"],

                                        np.where(sales\_data["Category"]=="Office Supplies",0.7\*sales\_data["Profit"],

                                                 np.where(sales\_data["Category"]=="Technology",0.6\*sales\_data["Profit"],0)))

# -\*- coding: utf-8 -\*-

"""

Excel Automation - pivot table function

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import glob

import pandas as pd

import numpy as np

os.chdir("D:\\Udemy\\Excel Automation\\assignment") # changes working directory

cwd = os.getcwd() # store current working directory in a variable

filenames = glob.glob(cwd+"\\\*\\\*xlsx")

#aggregate all excel files into one dataframe

consolidated = pd.DataFrame(columns = pd.read\_excel(filenames[0]).columns)

for file in filenames:

    temp = pd.read\_excel(file)

    consolidated = consolidated.append(temp, ignore\_index = True)

pivot\_df =   pd.pivot\_table(consolidated, values="Profit",

                           index = ["Segment","Category","Sub-Category"],

                           columns = ["Region","State"],

                           aggfunc = np.sum)

#####################################pivot parameters####################################

columns = ["Region"]

rows = ["Segment","Category","Sub-Category"]

values = ["Profit"]

#########################################################################################

#removing columns which are not required

relevant\_df = consolidated.loc[:,columns+rows+values]

#creating pivot

pivot\_df = relevant\_df.groupby(rows+columns).sum().unstack()

#exporting pivot to file

pivot\_df.to\_excel("pivot.xlsx")

pivot\_df.to\_html("pivot.html")

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Pandas series object

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import numpy as np

import pandas as pd

np\_array = np.array([2,6,1,3])

pd\_series = pd.Series(np\_array,index=["a","b","c","d"]) #creating series by passing ndarray, default indices created

pd\_series = pd.Series([2,6,1,3],index=["a","b","c","d"]) #creating series by passing list of values and list of indices

pd\_series2 = pd.Series({"alpha":2,"beta":6,"cappa":1,"delta":3}) #creating series by passing dictionary

pd\_series["b"] #accessing series element using index

pd\_series2[["alpha","cappa"]]

pd\_series2[2:] #series can be sliced using numeric index despite having non numeric indices

pd\_series2[1:3]

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Pandas dataframe object intro

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import pandas as pd

import numpy as np

df = pd.DataFrame(np.array([[1,2,3],

                            [4,5,6],

                            [7,8,9]]),

                  columns=["alpha","beta","cappa"]) #creating dataframe using 2-d array

df.columns = ["a","b","c"] #changing the names of columns

df.index = ["m","n","o"] #changing the names of indices

df\_dict = pd.DataFrame({"a":[1,2,3],

                        "b":[4,5,6],

                        "c":[7,8,9]}) #creating dataframe using dictionary

type(df["alpha"]) #Type of each column (and row) is series

df[["alpha","cappa"]] #extracting multiple columns

df.loc["m"] #extracting row using index names

df.iloc[1] #extracting row using numeric index

df.loc[["m","o"],["beta","cappa"]]

df.iloc[[0,2],[1,2]]

**Importing/Exporting Data from Excel Spreadsheet**

"""

Excel Automation - Pandas dataframe importing and exporting external datasets

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import pandas as pd

#reading excel data from a given sheet

data = pd.read\_excel("data.xlsx",sheet\_name="Orders",header=3,index\_col="Row ID")

data.head(10) #inspect first 10 rows of the dataframe

data.tail() #inspect last 10 rows of the dataframe

data.dtypes #inspect the types of each column of the dataframe

data.describe() #basic statistical analysis on numeric columns

data.columns #lists all the columns by their names

data.drop("Product Name",axis=1,inplace=True) #drop a column in the original dataframe

data.drop(["Product Name","Ship Date"],axis=1) #drop multiple columns in the original dataframe

data["Per Unit Sale"] = data["Sales"]/data["Quantity"] #create a new column by performing operation on other columns

data.to\_excel("data\_2.xlsx",index=False) #export dataframe to excel

**DATA Type**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - datatype handling

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import pandas as pd

#reading excel data from a given sheet

data = pd.read\_excel("data.xlsx",sheet\_name="Orders",header=3,index\_col="Row ID")

data.columns

data["Postal Code"] = data["Postal Code"].astype(str)

data["Order Date"] = data["Order Date"].astype(str)

data["Order Date"] = pd.to\_datetime(data["Order Date"], errors = "raise", format = "%Y-%m-%d")

data["month"] = data["Order Date"].dt.month #please refer documentation for pd.Series.dt

data["year"] = data["Order Date"].dt.year

data["processing time"] = data["Ship Date"] - data["Order Date"]

data["formatted date"] = data["Order Date"].dt.strftime("%b-%Y") #the output will be in string format and not datetime

**Merge :**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Pandas dataframe (merge,concatenate and join)

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import pandas as pd

menu = pd.DataFrame({"items":["pizza","pasta","salad","burritto","taco","burger"],

                    "price":[14.99,12.99,7.99,10.99,6.99,5.99],

                    "popularity":["high","medium","low","high","medium","high"]})

nutrition = pd.DataFrame({"item":["pizza","pastry","burritto","salad","pasta"],

                         "avg\_calorie":[3200,800,940,240,740],

                         "protein":["12%","4%","16%","6%","10%"]})

#####concatenate#########

pd.concat([menu,nutrition],axis=1,ignore\_index=False)

#####merge#######

menu.merge(nutrition,how="inner",left\_on="items",right\_on="item")

menu.merge(nutrition,how="outer")

menu.merge(nutrition,how="left")

menu.merge(nutrition,how="right")

#####join#####

menu.set\_index("items",inplace=False)

menu.reset\_index(inplace=False)

menu.set\_index("items").join(nutrition.set\_index("item"))

**Group by**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - pandas groupby function

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import pandas as pd

#reading excel data from a given sheet

data = pd.read\_excel("data.xlsx",sheet\_name="Orders",header=3,index\_col="Row ID")

data.groupby("Region").sum() #groupby one field

data.groupby(["Region","Category"]).sum() #groupby multiple fields

data.groupby(["Region","Category"]).sum().unstack()

gp = data.groupby("Region") #storing groupby object in a variable

gp.groups #dictionary of group entities and the corresponding indices

gp.get\_group("South") #similar to apply filter in excel

**Numpy assignment solution**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Numpy assignment solution (commute.xlsx)

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import pandas as pd

import numpy as np

#reading excel data from sheet

commute = pd.read\_excel("commute.xlsx", sheet\_name = "Sales", index\_col="Date")

commute.replace(["Yes","No"],[1,0],inplace=True) #replace Yes and No with 1 and 0

x = np.array(commute) #trasnform daily commute data into 2d array

y = np.array([8,3,0.5,12]) #store pricing information in a 1d array

daily\_expenses = x.dot(y) #calculating dot product

daily\_expenses.sum() #calculating sum of the product

daily\_expenses.argmax() #identifying index with maximum value

commute.index[daily\_expenses.argmax()].strftime("%Y-%m-%d") #using max index to identify the relavant date

**Panda Assignment**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Pandas Assignment

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import pandas as pd

data = pd.read\_excel("sales\_data.xlsx", index\_col = "Row ID") #import data

data["month\_year"] = data["Order Date"].dt.strftime("%m-%Y") #create another column having month year combination to perform grouping

gb = data.groupby("month\_year") #store groupby object in a variable

ls = [] #empty list which will be used to store grouped dataframes

for df in gb.groups:

    ls.append(gb.get\_group(df)) #appending grouped data to the list one by one

#exporting grouped dataframes into excel files - ensure that the current drive is set as required

for df in ls:

    file\_name = df["month\_year"].iloc[0]

    df.drop("month\_year",axis=1, inplace=True)

    df.to\_excel("{}.xlsx".format(file\_name), index=False)

**OS Library**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - os library intro

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

cwd = os.getcwd() # Stores the current working directory

os.mkdir("Test Folder") # Create a folder in the working directory; provide the full path if do not want to create folder in thr working directory

os.rmdir("Test Folder") # delete a folder in the working directory

os.system("data.xlsx") # launch a file in the working directory

os.path.join(cwd,os.mkdir("Test Folder")) # adding subpath to a path variable

# -\*- coding: utf-8 -\*-

"""

Excel Automation - file handling

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import shutil

os.chdir("D:\\Udemy\\Excel Automation\\assignment") #set working directory

cwd = os.getcwd() #store current working directory in a variable

#move file using os.rename

os.rename(os.path.join(cwd,"01-2014.xlsx"),os.path.join(cwd,"2014","01-2014.xlsx"))

os.rename("D:\\Udemy\\Excel Automation\\assignment\\01-2014.xlsx",

          "D:\\Udemy\\Excel Automation\\assignment\\2014\\01-2014.xlsx")

#move/rename folder using os.renames

os.renames("2015","2014")

#move file using os.replace

os.replace(os.path.join(cwd,"02-2014.xlsx"),os.path.join(cwd,"2014","02-2014.xlsx"))

#delete files using os.remove

os.remove("filename.xlsx")

#copy files using shutil

shutil.copy("03-2014.xlsx",os.path.join(cwd,"2014","03-2014.xlsx"))

#copy folder using shutil

shutil.copytree("2014", "2014\_copy")

#delete folder using shutil

shutil.rmtree("2014\_copy")

#move files using shutil

shutil.move("D:\\Udemy\\Excel Automation\\assignment\\01-2014.xlsx",

            "D:\\Udemy\\Excel Automation\\assignment\\2014\\01-2014.xlsx")

# -\*- coding: utf-8 -\*-

"""

Excel Automation - glob library intro

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import glob

os.chdir("D:\\Udemy\\Excel Automation\\data") # changes working directory

cwd = os.getcwd() # store current working directory in a variable

filenames = glob.glob("\*") # get all file/folder names in a given path

filenames = glob.glob(cwd+"\\\*\\\*\\\*.xls\*") # get all filenames having string .xls in their filenames

#get all filenames ending with 2017.xlsx

filenames = glob.glob("\*2017.xlsx")

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Assignment 1 solution

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import glob

os.chdir("D:\\Udemy\\Excel Automation\\assignment")

cwd = os.getcwd()

filenames = glob.glob("\*.xlsx")

for file in filenames:

    year = file.split(".")[-2][-4:]

    try:

        int(year)

    except:

        continue

    if os.path.isdir(year) == False:

        os.mkdir(year)

    os.rename(file,os.path.join(cwd,year,file))

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Assignment 1 solution

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import glob

os.chdir("D:\\Udemy\\Excel Automation\\assignment")

cwd = os.getcwd()

filenames = glob.glob("\*.xlsx")

for file in filenames:

    year = file.split(".")[-2][-4:]

    try:

        int(year)

    except:

        continue

    if os.path.isdir(year) == False:

        os.mkdir(year)

    os.rename(file,os.path.join(cwd,year,file))

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Assignment 2 solution

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import glob

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation\\assignment")

cwd = os.getcwd()

filenames = glob.glob(cwd+"\\\*\\\*xlsx")

consolidated = pd.DataFrame(columns = pd.read\_excel(filenames[0]).columns)

for file in filenames:

    temp = pd.read\_excel(file)

    consolidated = consolidated.append(temp, ignore\_index = True)

consolidated.to\_excel("consolidated.xlsx", index = False)

**Implementing Excel Functions in Python**

**VLookup**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - vlookup

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation")

sales = pd.read\_excel("sales\_data.xlsx") #importing left dataframe

zip\_income = pd.read\_csv("zipcode\_income.csv", engine='python') # importing right dataframe

temp = sales.merge(zip\_income.loc[:,["Zip\_Code","Mean"]].rename(columns={"Zip\_Code":"Postal Code", "Mean":"Mean Income"})

                   ,how="left",on="Postal Code") # merging the left df with relavant columns of the right df

temp.drop\_duplicates(subset=["Row ID"], keep ="first", inplace=True) # dropping duplicates

**VLookup-11**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - vlookup function

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation")

def vlookup(left\_df,right\_df,left\_key,right\_key,right\_val):

    """

    left\_df : path to excel data which requires vlookup values

    right\_df: path to excel data which is going to provide vlookup values

    left\_key: the key column in the left dataset

    right\_key: the key column in the right dataset

    right\_val: the column in the right dataset whose values need to be moved to the right dataset

    """

    left = pd.read\_excel(left\_df)

    left.reset\_index(inplace=True)

    right = pd.read\_excel(right\_df)

    right = right.loc[:,[right\_key,right\_val]].rename(columns={right\_key:left\_key})

    temp = left.merge(right, how="left", on=left\_key)

    temp.drop\_duplicates(subset=["index"], keep ="first", inplace=True)

    return temp.set\_index("index")

vlookup\_df = vlookup("sales\_data.xlsx","zipcode\_income.csv","Postal Code","Zip\_Code","Mean")

# -\*- coding: utf-8 -\*-

"""

Excel Automation - pivot table function

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import glob

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation\\assignment") # changes working directory

cwd = os.getcwd() # store current working directory in a variable

filenames = glob.glob(cwd+"\\\*\\\*xlsx")

#aggregate all excel files into one dataframe

consolidated = pd.DataFrame(columns = pd.read\_excel(filenames[0]).columns)

for file in filenames:

    temp = pd.read\_excel(file)

    consolidated = consolidated.append(temp, ignore\_index = True)

#####################################pivot parameters####################################

columns = ["Region"]

rows = ["Segment","Category","Sub-Category"]

values = ["Profit"]

#########################################################################################

#removing columns which are not required

relevant\_df = consolidated.loc[:,columns+rows+values]

#creating pivot

pivot\_df = relevant\_df.groupby(rows+columns).sum().unstack()

#exporting pivot to file

pivot\_df.to\_excel("pivot.xlsx")

pivot\_df.to\_html("pivot.html")

# -\*- coding: utf-8 -\*-

"""

Excel Automation - pivot table function

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import glob

import pandas as pd

import numpy as np

os.chdir("D:\\Udemy\\Excel Automation\\assignment") # changes working directory

cwd = os.getcwd() # store current working directory in a variable

filenames = glob.glob(cwd+"\\\*\\\*xlsx")

#aggregate all excel files into one dataframe

consolidated = pd.DataFrame(columns = pd.read\_excel(filenames[0]).columns)

for file in filenames:

    temp = pd.read\_excel(file)

    consolidated = consolidated.append(temp, ignore\_index = True)

pivot\_df =   pd.pivot\_table(consolidated, values="Profit",

                           index = ["Segment","Category","Sub-Category"],

                           columns = ["Region","State"],

                           aggfunc = np.sum)

#####################################pivot parameters####################################

columns = ["Region"]

rows = ["Segment","Category","Sub-Category"]

values = ["Profit"]

#########################################################################################

#removing columns which are not required

relevant\_df = consolidated.loc[:,columns+rows+values]

#creating pivot

pivot\_df = relevant\_df.groupby(rows+columns).sum().unstack()

#exporting pivot to file

pivot\_df.to\_excel("pivot.xlsx")

pivot\_df.to\_html("pivot.html")

**If**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - If function

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

import numpy as np

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

sales\_data["Profit Net Tax"] = np.where(sales\_data["Category"]=="Furniture",0.8\*sales\_data["Profit"],

                                        np.where(sales\_data["Category"]=="Office Supplies",0.7\*sales\_data["Profit"],

                                                 np.where(sales\_data["Category"]=="Technology",0.6\*sales\_data["Profit"],0)))

# -\*- coding: utf-8 -\*-

"""

Excel Automation - string functions

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

#extract substring from a text column - left, right, mid

sales\_data["Order ID"].str[3:7] # mid function

sales\_data["Order ID"].str[:2] # left function

sales\_data["Order ID"].str[-2:] # right function

sales\_data["Order ID"].str.split("-").str[0]

#trim leading and trailing spaces

sales\_data["Order ID"] = sales\_data["Order ID"].str.strip()

#concatenate

sales\_data["location"] = sales\_data["State"]+"\_"+sales\_data["City"]

#capitalize

sales\_data["State"] = sales\_data["State"].str.upper()

sales\_data["State"] = sales\_data["State"].str.lower()

#find a string

sales\_data["City"].str.find("Fort")

**SumIf and CountIf**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - sumif(s) and countif(s)

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

#countif

len(sales\_data[sales\_data["Quantity"] > 5])

sales\_data[sales\_data["Quantity"] > 5].shape[0]

#countifs

len(sales\_data[(sales\_data["State"]=="Kentucky") & (sales\_data["Quantity"] > 5)])

sales\_data[(sales\_data["State"]=="Kentucky") & (sales\_data["Quantity"] > 5)].shape[0]

#sumif

sales\_data[sales\_data["City"].str[:4]=="Fort"]["Profit"].sum()

sales\_data.loc[sales\_data["City"].str[:4]=="Fort","Profit"].sum()

#sumifs

sales\_data[(sales\_data["City"].str[:4]=="Fort") & (sales\_data["Quantity"] > 5)]["Profit"].sum()

sales\_data.loc[(sales\_data["City"].str[:4]=="Fort") & (sales\_data["Quantity"] > 5),"Profit"].sum()

**Visualization**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Pandas visualization basics

@author: Mayank Rasu (http://rasuquant.com/wp/)

@author: User

"""

import os

import pandas as pd

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

sales\_data.groupby("Region")[["Sales","Profit"]].sum().plot(kind="bar",

                                                            title="Regional Sales & Profits",

                                                            subplots = True,

                                                            layout = (1,2),

                                                            sharey = True)

sales\_data.groupby("Region")[["Sales","Profit"]].sum().plot(kind="bar",

                                                            title="Regional Sales & Profits",

                                                            stacked = True)

**Matplotlib Object Oriented Interface Intro**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Matplotlib Object Oriented Interface Intro

@author: Mayank Rasu (http://rasuquant.com/wp/)

@author: User

"""

import matplotlib.pyplot as plt

import numpy as np

x = np.arange(1,10)

# creating side by side plots

fig = plt.figure()

ax1 = fig.add\_axes([0,0,0.48,1])

ax2 = fig.add\_axes([0.53,0,0.48,1])

ax1.plot()

ax2.plot()

#creating plot within another plot and tweaking plot features

fig = plt.figure()

ax1 = fig.add\_axes([0,0,1,1])

ax2 = fig.add\_axes([0.15,0.7,0.25,0.25])

ax1.plot(x,x\*\*2, color = "r")

ax2.plot(x,np.log(x), ls = "--")

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Matplotlib Object Oriented Interface Intro - II

@author: Mayank Rasu (http://rasuquant.com/wp/)

@author: User

"""

import matplotlib.pyplot as plt

import numpy as np

x = np.arange(1,10)

# creating side by side plots

fig = plt.figure()

fig.suptitle("Major Title")

ax1 = fig.add\_axes([0,0,0.45,0.9])

ax1.set\_title("x squared")

ax1.set\_xlabel("x values")

ax1.set\_ylabel("y values")

ax2 = fig.add\_axes([0.5,0,0.45,0.9])

ax2.set\_title("log of x")

ax2.set\_xlabel("x values")

ax2.set\_yticks([0,1,2])

ax1.plot(x, x\*\*2)

ax2.plot(x, np.log(x), color = "r")

**Combining pandas visualization with matplotlib OO interface**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Combining pandas visualization with matplotlib OO interface

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

import matplotlib.pyplot as plt

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

fig = plt.figure()

fig.suptitle("City level Performance", x = 0, y = 1)

ax1 = fig.add\_axes([0,0,1,0.9])

ax2 = fig.add\_axes([0.6,0.55,0.25,0.25])

ax1.set\_title("Top Cities")

ax2.set\_title("Laggard Cities")

sales\_data.groupby("City")["Sales"].sum().sort\_values(ascending=False).iloc[:15].plot(kind="barh", color = "green", ax=ax1)

sales\_data.groupby("City")["Sales"].sum().sort\_values(ascending=True).iloc[:5].plot(kind="bar", color = "red", ax=ax2)

**Visualization assignment – I**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Visualization assignment - I

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

import matplotlib.pyplot as plt

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

#side by side pie charts representing sales breakdown by category and by segment

plt.style.use("ggplot")

fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2)

plt.suptitle("Sales Breakdown by Category & Segment")

ax1.set\_title("Category Level Sales \n Breakdown")

data = sales\_data.groupby("Category")["Sales"].sum().to\_list()

labels = sales\_data.groupby("Category")["Sales"].sum().index

ax1.pie(data,labels=labels,autopct="%.2f%%")

ax2.set\_title("Segment Level Sales \n Breakdown")

data = sales\_data.groupby("Segment")["Sales"].sum().to\_list()

labels = sales\_data.groupby("Segment")["Sales"].sum().index

ax2.pie(data,labels=labels,autopct="%.2f%%")

**Visualization assignment (Advanced**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Visualization assignment - I (advanced)

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

import matplotlib.pyplot as plt

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

def fmt\_wdgs(data):

    def fmt\_values(pct):

        total = sum(data)

        abs\_value = round(pct\*total/100000,1)

        return "${}K".format(abs\_value)

    return fmt\_values

#side by side pie charts representing sales breakdown by category and by segment

plt.style.use("ggplot")

fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2)

plt.suptitle("Sales Breakdown by Category & Segment")

ax1.set\_title("Category Level Sales \n Breakdown")

data = sales\_data.groupby("Category")["Sales"].sum().to\_list()

labels = sales\_data.groupby("Category")["Sales"].sum().index

explode = [0 if x!=max(data) else 0.1 for x in data]

ax1.pie(data,labels=labels,explode=explode,autopct=fmt\_wdgs(data))

ax2.set\_title("Segment Level Sales \n Breakdown")

data = sales\_data.groupby("Segment")["Sales"].sum().to\_list()

labels = sales\_data.groupby("Segment")["Sales"].sum().index

explode = [0 if x!=max(data) else 0.1 for x in data]

ax2.pie(data,autopct="%.2f%%",explode=explode)

ax2.legend(labels,bbox\_to\_anchor=(1,0,0.5,0.5))

**Visualization assignment – II**

# -\*- coding: utf-8 -\*-

"""

Excel Automation - Visualization assignment - II

@author: Mayank Rasu (http://rasuquant.com/wp/)

"""

import os

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.ticker as tkr

os.chdir("D:\\Udemy\\Excel Automation") # changes working directory

sales\_data = pd.read\_excel("sales\_data.xlsx")

sales\_data["year"] = sales\_data["Order Date"].dt.year

timeseries\_data = sales\_data.groupby("Order Date")[["year","Sales"]].agg({"year":"first","Sales":"sum"})

timeseries\_data.sort\_index(inplace=True)

def new\_ticks(x,pos):

    return "${}K".format(round(x/1000))

#visualization representing cumulative sales as a timeseries

plt.style.use("ggplot")

fig = plt.figure()

fig.suptitle("Cumulative Sales", x = 0.5, y = 1)

ax1 = fig.add\_axes([0,0,1,0.9])

ax1.set\_title("2017 cumuative sales")

data\_2017 = timeseries\_data[timeseries\_data["year"]==2017]

data\_2017["cumsales"] = data\_2017["Sales"].cumsum()

ax1.plot(data\_2017["cumsales"])

ax1.set\_ylabel("Sales")

ax1.yaxis.set\_major\_formatter(tkr.FuncFormatter(new\_ticks))

ax2 = fig.add\_axes([0.2,0.55,0.25,0.25])

ax2.set\_title("Yearly Sales")

x = timeseries\_data.groupby("year").sum().index

height = timeseries\_data.groupby("year").sum()["Sales"].to\_list()

ax2.bar(x,height)

ax2.set\_yticks([0,200000,400000,600000,800000])

ax2.set\_xticks(x)

ax2.set\_ylabel("Sales")

ax2.yaxis.set\_major\_formatter(tkr.FuncFormatter(new\_ticks))