**ESP 4a – Things We Must Remember**

**Some general remarks –**

Make sure you **only open the folder you are working on** in VSC Explorer

**Read the instructions very carefully**, do not start work until you know **exactly** what is required.

**Read the data file carefully**, make sure you can read all column headings, and that **you know what the data represents**.

**Look through the code you are given** – there is probably **code here you can use** e.g. importing, reading a CSV etc, check how they have formatted outputs and do similar in your own work

**For the actual work -**

I would **start each part of the work in a new file,** when complete, copy over to the working file you were given. Save all files and submit everything you did.

**#Comment everything!** Effectively use #comments to have a conversation with the assessor, where you explain your thinking and what you have done at every stage of your work.

To check things, **print()** as you go along, when happy, comment out and leave for examiner to see.

If you get a **Key Error**, check date formatting e.g., change **2022** to **22** or vice versa.

When outputting information, **use similar formatting to the code you have been given** – don’t forget you get marked on presentation.

If you cannot do exactly what you have been asked to do – well, what can you do? **At least do something, even if it is not exactly what they asked for** as this will evidence understanding for which you should get some marks. For example, if you cannot plot data using MatPlotLib, **print()** instead, using comments to explain what you’ve done.

**THE BASICS**

**Importing Matplotlib, Numpy and Pandas -**

import pandas as pd # pandas referred to as pd

import matplotlib.pyplot as plt # matplotlib referred to as plt

import numpy as np # numpy referred to as np

**Reading a file –**

df = pd.read\_csv('Task4a\_data\_November 22.csv') # access all data from CSV, store in variable called df

**To read an Excel file –**

df = pd.read\_**excel**('/Users/datagy/Desktop/Sales.xlsx')

**EXTRACT SPECIFIC DATA**

**Store data based on a specific criteria (“DUB”) from a specific column heading (“From”) into a variable, then print to confirm –**

dub\_df = df[df["From"] == "DUB"] # extract From DUB only from df, store in dub\_df

# print(dub\_df) # print to check this works, then comment out

**Remember this format to extract specific data out of a column –**

# dub\_df = df[df["COLUMN NAME"] == "CRITERIA"]

**To select against multiple columns and criteria –**

twoOnly = df[(df["From"] == "DUB") & (df["To"] =="LHR")] # Dublin to London Heathrow only departures

# print(twoOnly) # check this works

**Remember this format to extract data based on two column and two criteria –**

twoOnly = df[(df["COLUMN"] == "CRITERIA") & (df["COLUMN"] =="CRITERIA")]

**Create a virtual column (Total Passengers) from a sum of the other columns, and printing to confirm –**

df["Total Passengers"] = df["01/04/2022"] + df["02/04/2022"] + df["03/04/2022"] + df["04/04/2022"]

# creates 'virtual' column called "Total Passengers" which is the sum of each row left to right

# to confirm this we can print "Total Passengers" column ->

#print(df["Total Passengers"]) # print to check this works, then comment out - leave for verifier to enjoy!

**Store all data based on a specific criteria (“DUB”) from a specific column heading (“From”) into a variable, then print to confirm –**

dub\_df = df[df["From"] == "DUB"] # extract From DUB only from df, store in dub\_df

# print(dub\_df) # print to check this works, then comment out

**Sum up a total -**

dub\_df\_passenger\_count = dub\_df["Total Passengers"].sum()

# print(dub\_df\_passenger\_count)

**Print a total by joining text and a variable -**

print ("Total for Dublin = ", dub\_df\_passenger\_count)

Bar graph using X and Y axis, plus labels –

x = np.array(["Bungalows", "Semis", "Detached"])

y = np.array([bung\_df\_increase\_count , semi\_df\_increase\_count, det\_df\_increase\_count])

plt.title("Sum of % Increase")

plt.xlabel("Property Type")

plt.ylabel("% Increase")

plt.grid()

plt.bar(x,y)

plt.show()

To work out an average of a range of figures, use **mean()** rather than sum() –

lon\_df\_av\_increase = lon\_df["Total % Increase"].mean()

# print(lon\_df\_av\_increase)