Programming and Scripting
Lab Extra 11- Pandas

Introduction.

This is extra material I have put in, that you may find interesting.

I would suggest that you create another folder in labs called **Topic11-Pandas**.
Sample code and data on github.

I/O

1. Create a data from with data in it. You may use the Dict with lists, dict with dict or list with list method, and print out the dataFrame. (I am using list with list, but setting the column names). You can use head to just print ou the first 3 rows

```
import pandas as pd

listDAta= [
    ['John', 'math',        23],
    ['John', 'programming', 66],
    ['Mary', 'math',        45],
    ['Mary', 'programming', 33],
    ['Mark', 'SIEM',        57],
    ['Mark', 'programming', 70],
    ['Mark', 'math',        73],
    ['John', 'programming', 61]
]

df = pd.DataFrame(listDAta, columns=['name','subject','grade'])
print(df.head(3))
```

2. Use the describe function to see the overall stats on the grades (use the type function to see that this output is a dataframe

```
print(df.describe())
print(type(df.describe()))
```

3. Write this dataframe to a csv file called grades.csv (look at the csv file, why does the first line start with a comma

```
path = "./data/"
csvFilename = path + 'grades.csv'
df.to_csv(csvFilename)
```

4. And write it to an excel file called grades.xlsx (in sheet 'data') without the index column

```
excelFilename = path +'grades.xlsx'
df.to_excel(excelFilename, index=False, sheet_name='data')
```

5. Add the description to another sheet called 'summary'. You will need to change the writer to open from the default writer. (if you get a permission denied, close the workbook in excel)

```
with pd.ExcelWriter(excelFilename, engine='openpyxl', mode='a') as writer:
    df.describe().to_excel(writer, sheet_name="summary")
```

6. Calculate and print the mean of the grades

```
mean =  df.describe().loc['mean','grade']
print(mean)
# or
mean = df['grade'].mean()
print (mean)
```

Extra Extra: Read from log file and clean data

This is for cybersecurity and really going way beyond what you need for now.
This section I am looking at a log file from a webserver a sample of it can be downloaded from my github here (click download)
[pands-course-material/access.log.demo at main · andrewbeattycourseware/pands-course-material (github.com)](https://github.com/andrewbeattycourseware/pands-course-material/blob/main/access.log.demo)
file is in this weeks sample code

7. Read an assess.log file into a dataFrame (assume the delimiter is ‘ ‘is

```
import pandas as pd

path = './data/'
logFilename = path + 'access.log.demo'

df = pd.read_csv(logFilename, sep=' ', header= None)


print(df)
```

8. Mmmmmmmmm……. Did this work lets check by printing the first line (index 0)

```
print(df.iloc[0])
```

9. It would be good to have meaningful column names, so change the read_csv (I looked up what each column means at apache.org, as you can see the file has ip addresses, timestamps, urls and lots of data from users accessing a web site

```
colNames= ('ip',
    'dash1',
    'userId',
    'time',
    'url',
    'status code',
    'size of response',
    'referer',
    'user agent',
    'dunno'
)
df = pd.read_csv(logFilename, sep=' ', header= None, names=colNames)
```

10. Drop the columns that contain just dashes, (if the webserver is set up for userIds you may not want to do this)

```
df.drop(columns=['dash1', 'userId'], inplace=True)
```

11. The time looks badly formatted let's remove the [] and change the type of the column to data time. The apply function is handy for this

```
# remove the [] from time
# there is a lot in this line
# apply with apply the function to each element in the column and return
a series
# which I make the column equal to
# I think I covered lamba function in the functions topic

df['time'] = df['time'].apply(lambda x: re.search('[\w:/]+', x).group())
# for the task you may want to use a normal function instead of lambda
'''
def getNewValue(x):
    newvalue = re.search('[\w:/]+', x).group()
    # do your stuff
    return newvalue

df['time'] = df['time'].apply(getNewValue)
'''
```

12. Lets check the column types

```
print (df.dtypes)
```

13. Change the type of the time column to dateTime

```
# this is not a normal date time format so we need to specify it
# https://docs.python.org/3/library/datetime.html#strftime-and-
strptime-behavior
# https://pandas.pydata.org/pandas-
docs/stable/reference/api/pandas.to_datetime.html?highlight=to_date
time
df['time'] = pd.to_datetime(df['time'], format='%d/%b/%Y:%H:%M:%S')
```

We can now do a bit of analysis.

14. Get the events that happened between two times.

```
# way one use the loc function
#newdf = df.loc[(df['time'] > start_date) & (df['time'] < end_date)
]
# way 2 use the series function between
#newdf = df[df.time.between(start_date, end_date)]
# way three set the index to the date column
# this give a whole pile of handy features
df = df.set_index(['time'])
newdf = df['2021/02/15 23:00':'2021/02/15 23:59:59']
#newdf = df.between_time('23:00', '23:59') # this is times every da
y
print (newdf)
```

15. Get mean amount of data that was downloaded every half an hour

```
# sample the download sizes say every 30 Minutes
downloadSamples = df['size of response'].resample(rule='30Min').mea
n()
print(downloadSamples)
```

16. Plot this in line plot

```
# ok I cheated a bit here
# more information on the documents
# https://pandas.pydata.org/pandas-
docs/stable/getting_started/intro_tutorials/04_plotting.html
downloadSamples.plot()
plt.show()
```