KANAV BANSAL

WORKING WITH .CSV:

CSV - COMMA SEPERATED VALUES.

It is a simple file format used to store tabular data, such as spreadsheets or database. A CSV file stores tabular data (numbers and text) in a plain text. Each line of the file is a data record.

Each record consists of one or more fields, seperated by commas. The use of the comma as a field seperator is the source of the name for this file format.

For working CSV files in python, there is an inbuilt module called csv.

Because it's a plain text file, it can contain only actual text-data - in other

words, printable ASCII or unicode characters.

READING CSV FILES : (IN PYTHON)

To read the CSV file in python, we use csv.reader() function.

There are some modules under csv, they are

	FUNCTIONS	DESCRIPTION
1.	csv.field_size_limit	It returns the max. field size.
2.	csv.get_dialect	Fetches the dialect associated with the name.

DIALECT - A particular version of programming language.

3.	csv.list_dialect	Displays all the registered dialect.
4.	csv.register_dialect	Dialect associated with a name

5. CSV. writer - writes data to a CSV file.

6. CSV. unregister_dialect - It deletes the dialect associated with the name dialect registry.

7. CSV. QUOTE_ALL - Quotes everything irrespecti-ve of the type.

8. CSV. QUOTE_MINIMAL - Quotes special character field.

9. CSV. QUOTE_NONNUMERIC - Quotes fields that are not numerial.

10. CSV. QUOTE_NONE - Doesn't quote anything in output.

READING .CSV FILES : IN PANDAS

To import csv files in pandas, we use read_csv( ). and do operations on it.

PARAMETER'S OF ANY FONCTIONS :

PARAMETER	USE
1. filepath_or_buffer	- URL or Dir location of file.
2. sep	- stands for seperator, default is ',' as in csv
3. index_col	- Makes passed column as index instead of 0, 1, 2, 3..... n.
4. header	- Makes passed row/s [int/int list] as header.
5. use_cols	- Only uses the passed col [string list] to make data frame.
6. squeeze	- If true and only column is passed, returns pandas series.
7. skiprows	- skips passed rows in new data frame.

IRIS DATASET : CSV

df - dataframe

→ df = pd.read_csv('data/Iris.csv')

df.head()

↓

Gives the top '5' Iris data

→ df.tail()

↳ Gives the last '5' Iris data

→ print(df.shape) - Gives the shape

→ print(df.columns)

o/p - Index(['Id', 'sepallengthcm', 'sepalwidthcm',
'petallengthcm', 'petalwidthcm', 'species'],
dtype = 'object')

→ print(df.mean())

→ print(df.std())

→ df.describe()

→ df.describe(include = ['object'])

WEATHER DATA :

→ df = pd.read_csv('data/nyc_weather.csv')

df.head()

→ df['Temperature'].max() → max. temperature.

→ df[df.Temperature == df.Temperature.max()]

↳ select the row with max. temperature

tsv - tab seperated value

If the file is in different location, then the path should be given as r and then the file name.

```
r'filepath.csv'
```

→ df['EST'][df['Events'] == 'Rain'] → which day it rains.

→ df.EST[df.Temperature == df.Temperature.max()]

↳ selects the day with max. temperature.

→ df['windspeedMPH'].mean()

↳ Average windspeed.

→ df['Events'].value_counts()

↳ Gives the counts of the events.

→ df['Events'].unique()

↳ Gives the data of events which are unique.

→ df.info() - Summary of the data.

READING .XLSX FILES :

The simplest way to read excel files into pandas data frame

SYNTAX :

```
df = pd.read_excel('path_to_excel_file',
                   sheet_name='...')
```

→ df = pd.read_excel('data/weather_data.xlsx')

df.head()

DATAFRAME TO .CSV & .XLSX :

I/P - import numpy as np
import pandas as pd

```
dict = {'Name': pd.series(['Tom','Jack','steve',
        'Ricky','vin','James','smith']),
        'Age': pd.series([25,26,25,35,23,33,31]),
        'Rating': pd.series([4.23,4.1,3.4,5,2.9,4.7,3.1])}
df = pd.DataFrame(dict)
Print(df)
```

O/P -

	Name	Age	Rating
0	Tom	25	4.23
1	Jack	26	4.10
2	steve	25	3.40
3	Ricky	35	5.00
4	vin	23	2.90
5	James	33	4.70
6	Smith	31	3.10

Dataframe to csv

↳ df.to_csv('data/temp/new_csv_file.csv')

Dataframe to csv without index

↳ df.to_csv('data/temp/new_csv_file_1.csv',
index = False)

Dataframe to XLSX

↳ df.to_excel('data/temp/new_excel_file.xlsx',
Sheet_name = 'Stud_data')

Dataframe to XLSX without index

↳ df.to_excel('data/temp/new_excel_file_noIndex.
xlsx', sheet_name = 'Stud_data', index=False)