Learning Objectives

1. Pandas Recap

2. Working with CSV files

DPhi

Pandas Session Recap
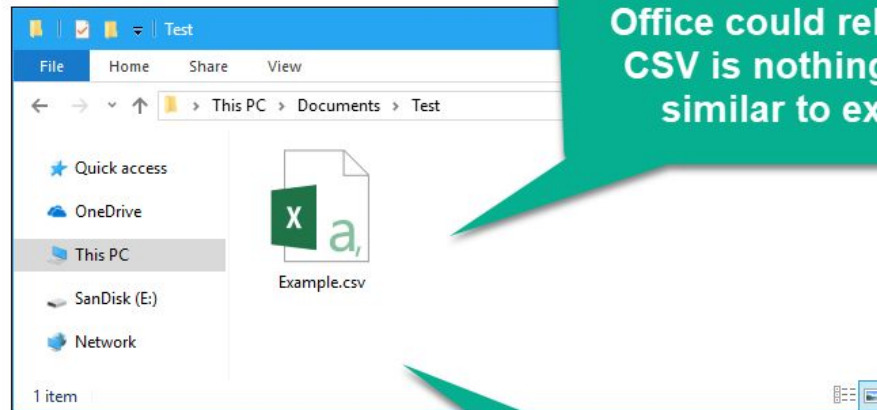
What we learnt so far?

- Pandas Objects/Data Structures (refer pre-session slides for clear explanation)

    - Series
    - Dataframe

- Data indexing and selection

    - **Use case:** Helps fetch a data record when we are dealing with large volumes of data. For ex: If you have a data with 10 million records, you can easily fetch information of a particular serial no/index no with pandas. This operation may not be feasible with MS Excel while dealing with such large volumes

Pandas Session Recap

- Data Wrangling & Handling Missing Values - read the notebook.

  - **Use case:** Often Data Scientists get unclean data with a lot of missing values and we need to have a solution to deal with it. Pandas facilitate in handling this issue

- Pandas String Operations

  - **Use case:** Helps in Handling Missing Values

DPhi

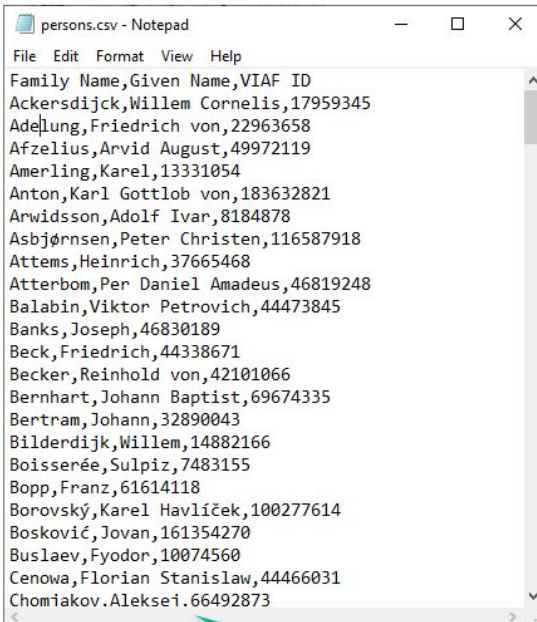# Working with CSVs

DPhi

What is a CSV file?

Windows users using MS Office could relate to this. CSV is nothing but a file similar to excel files

Only catch is, in CSV you store data with comma separated values
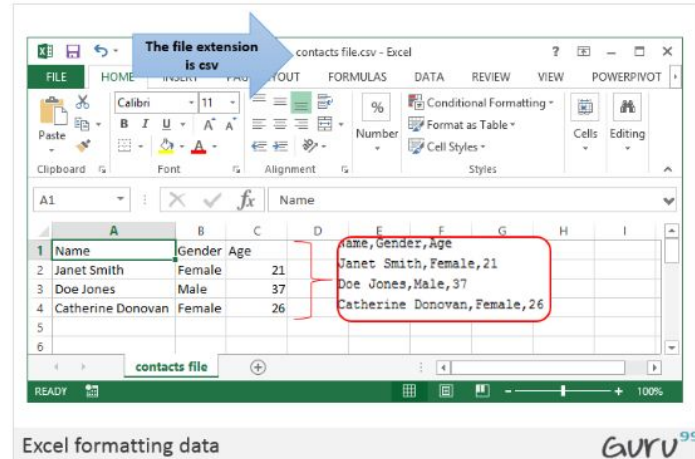
DPhi

What is a CSV ?

- CSV files are normally created by programs that handle large amounts of data. They are a convenient way to export data from spreadsheets and databases as well as import or use it in other programs.

- CSV (Comma Separated Values) is a simple file format used to store tabular data, such as a spreadsheet or database.

- A CSV file stores tabular data (numbers and text) in plain text.

- Each line of the file is a data record.

- Each record consists of one or more fields, separated by commas.

- The use of the comma as a field separator is the source of the name for this file format.

How does it look like?

Looks like this when opened on Notepad

Looks like this when opened on MS Excel

Excel formatting data

DPhi

Working with CSV files in Python

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

- However, a common method for working with CSV files is using Pandas. It makes importing and analyzing data much easier.

- One crucial feature of Pandas is its ability to write and read Excel, CSV, and many other types of files.

DPhi

Pandas read_csv

- Functions like the Pandas read_csv() method enable you to work with files effectively.

- The read_csv() function reads the CSV file into a DataFrame object.

- A CSV file is similar to a two-dimensional table and the DataFrame object represents two dimensional tabular view.

- The most basic way to read a csv file in Pandas:

```
# Import pandas
import pandas as pd
```

```
# reading csv file
pd.read_csv("filename.csv")
```

- Read the next slide to understand how to provide filename

Pandas read_csv

```
In [4]: # Import pandas
        import pandas as pd
```

```
In [5]: pwd #Let's get to know which directory/folder we are working with
```
```
Out[5]: 'C:\\Users\\chanukya\\Documents'
```

```
In [9]: # reading csv file
        pd.read_csv("test1.csv") # we have a file with name test1.csv in documents folder so it is easy to directly call the name
```

Out[9]:

	username	password	firstname	lastname	email	cohort1
0		NaN	Chanukya	Patnaik	Learner		Beginner

```
In [10]: #Let's say the file is in a different directory Eg: on my desktop
```

```
In [11]: pd.read_csv("C:/Users/chanukya/Desktop/test1.csv") #we can directly put the folder location as mentioned here within parentheses
```

Out[11]:

	username	password	firstname	lastname	email	cohort1
0		NaN	Chanukya	Patnaik	Learner		Beginner

Pandas read_csv

- There are many other things one can do through this function only to change the returned object completely.

- For instance, one can read a csv file not only locally, but from a URL through read_csv or one can choose what columns needed to export so that we don't have to edit the array later.

- These modifications can be done by the various arguments it takes.

- We don't need to memorise all the arguments though, let's have a look at few important ones in the next two slides.

DPhi

Pandas to_csv

- The easiest way to write DataFrames to CSV files is using the Pandas to_csv function

- <u>Syntax:</u>

```python
# DataFrame to CSV file
# df is the name of the DataFrame here
df.to_csv('file_name.csv')
```

Where df is the name of your DataFrame

- If you want to export without the index, simply add index=False

```python
# Specify index as False to import without index
df.to_csv('file_name.csv', index=False)
```

DPhi

Pandas to_csv with example

Read the
comments
carefully

```
In [15]: a = pd.read_csv("C:/Users/chanukya/Desktop/test1.csv") #I am just importing a dataframe and storing it as "a"
```

```
In [17]: a.to_csv("C:/Users/chanukya/Desktop/test2.csv") #coverting the dataframe into csv and storing it with a new filename test2.csv

         # here "a" is the name of the dataframe which was imported, however, you can create your own dataframe with pandas
         # and export it to your desired local folder using the above line of code
```

DPhi

Comprehensive Tutorial

- ==Must read-==
  A comprehensive tutorial on pandas for beginners.

  https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/

DPhi

Let's Practice!

Now let's practice what we've learnt on a real dataset

The dataset can be directly accessed with the link (Yes, read_csv accepts links too!):

Data Source:
https://data.un.org/Data.aspx?q=greenhouse+gas+co2+emissions&d=GHG&f=seriesID%3aGHG

DPhi

Let's Practice!

- Load the CSV file into a variable - greenhouse_data

- Use the description function to understand how the data looks like

- Print its first 10 rows using head()

- Print its last 10 rows using tail()

- Check if there are any null values

- Store the DataFrame to a CSV file named 'file2.csv'

DPhi

That's it for this unit. Thank you!

Feel free to post any queries on [Discuss](.).

DPhi