Data Management in Python - Importing & Exporting Data

Using Pandas Library

- To import files of these formats, we will be using the Pandas library.
- pandas is a Python Package providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
- pandas was created to replicate the data management capabilities of languages such as R that have been built for the purpose of analysis.
- Because it supports the use of various file formats, Pandas will be our preferred library for data management.
- It is built on the library NumPy, which can also be used for importing files of a single data type. However, in data science, most of our data sets include variables of different data types. Therefore, we will be ignoring NumPy for now.

Data Snapshot

basic_salary data consist salary of each employee with it's Location & Grade.

Variables

Observations

First_Name	Last_Name	Grade	Location	ba	ms
Alan	Brown	GR1	DELHI	17990	16070

Columns	Description	Type	Measurement	Possible values
First_Name	First Name	character	-	-
Last_Name	Last Name	character	-	-
Grade	Grade	character	GR1, GR2	2
Location	Location	character	DELHI, MUMBAI	2
ba	Basic Allowance	numeric	Rs.	positive values
ms	Management Supplements	numeric	Rs.	positive values

read_csv() Function

Importing a .csv file

```
import pandas as pd
salary_data = pd.read_csv("C:/Users/Documents/basic_salary.csv")
```

pd.read_csv() *assumes* ***header = TRUE*** *and* ***sep*** *= "," by default.*

*

First locate your data file, whether it is saved in the default working directory of Python or any other location in your system. If it is not stored in default working directory then you will have to give its path for importing it into Python. If you copy file path from the folder, ensure it uses forward slash (/).
Do not forget to accurately write the file name and extension.

read_table() Function

Importing a .txt file

```
import pandas as pd
salary_data = pd.read_table("C:/Users/Documents/basic_salary.txt")
```

header* = *infer *(default) indicates that the first row of the file contains the names of the columns. Pass 0 if you wish to explicitly define column names.*

sep* = *"/t" *(default) specifies that the data is separated by tab.*

delim_whitespace *= specifies whether whitespace is supposed to be considered as a delimiter. Default value is false.*

names *= array of column names you wish to define. Eg.* ***names*** *= ['A', 'B', 'C']*

*

First locate your data file, whether it is saved in the default working directory of Python or any other location in your system. If it is not stored in default working directory then you will have to give its path for importing it into Python. If you copy file path from the folder, ensure it uses forward slash (/).
Do not forget to accurately write the file name and extension.

How Does Python Handle Missing Observations?

- By default, Pandas interprets the following values as null values – '-1.#IND', '1.#QNAN', '1.#IND', '-1.#QNAN', '#N/A N/A', '#N/A', 'N/A', 'NA', '#NA', 'NULL', 'NaN', '-NaN', 'nan', '-nan', ''

- The read methods also accept the following arguments which deal with missing observations -

  — **`na_values`**= Accepts strings, dictionaries of additional values to be considered null.

  — **`na_filter`**= TRUE (default), detects missing value markers (like empty strings and the value of **`na_value`** values).

  — **`skip_blank_lines`**= TRUE (default), skips blank lines and moves on to the next data entry.

Sometimes data may contain random values which are considered as missing values. It is important to detect those values and overcome them. Therefore, we have a special tutorial for Handling missing values where you will learn to deal with different types of missing data.

Exporting CSV, Text and XLSX Files

- Sometimes you may want to export data saved as object from Python workspace to different file formats. Methods for Exporting Python objects into CSV, TXT and XLSX formats are given below:

```
# To a CSV
salary_data.to_csv('path_to_file/file_name.csv')
```

- ***path_to_file*** *here is the path where the file needs to be saved.*
- ***file_name*** *is the name you want to specify for that file.*

```
# To a Tab Delimited Text File
salary_data.to_csv('path_to_table/file_name.txt',sep='\t',
index=False)
```

```
# To an Excel Spreadsheet
salary_data.to_excel('path_to_file/file_name.xlsx')
```