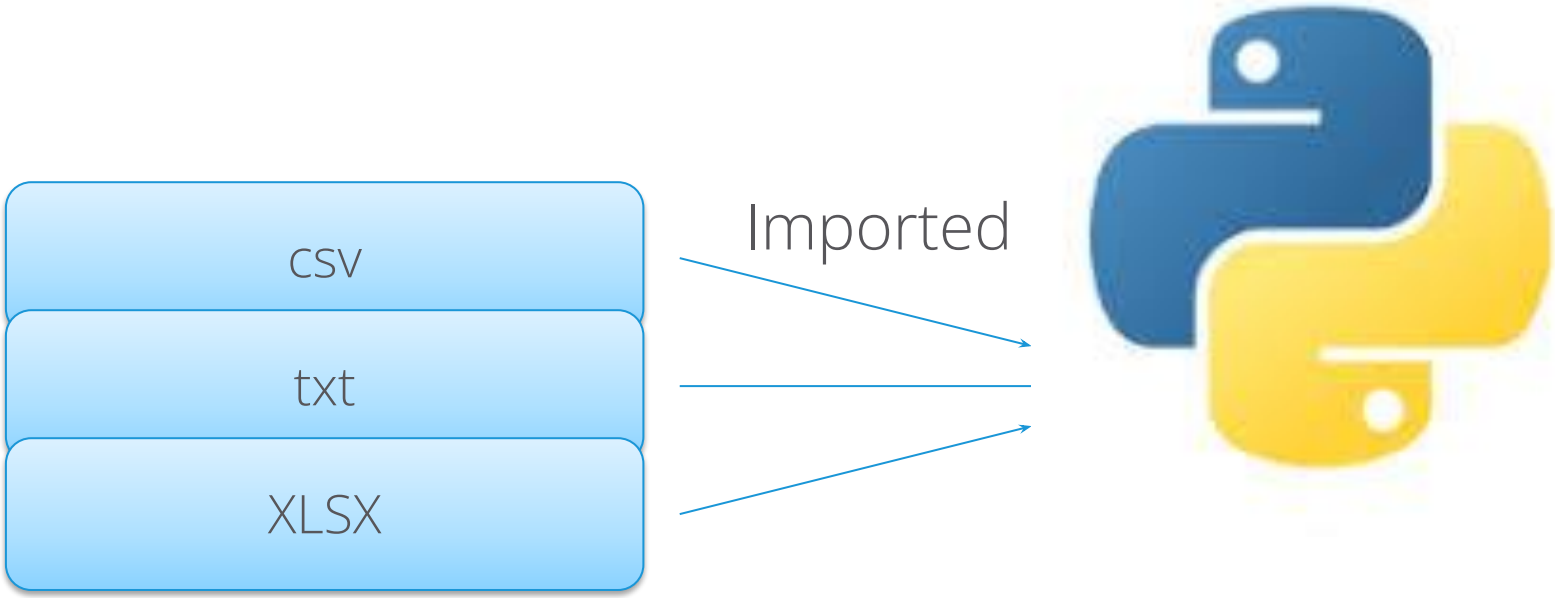
Data Management in Python -

Importing & Exporting Data

(CSV, TXT, XLSX, SAS, STATA, SPSS, MySQL, PostgreSQL and Oracle)

Contents

Your Data



Common File Formats

The most commonly used file formats for storing data

CSV:

Comma Separated Values file. Allows data to be saved in a table structured format; with commas acting as separators.

TXT:

Text file. Stores data in plain text format, separated by different types of delimiters such as blank space, tab ("\t"), comma (","), pipe ("|"), etc.

File Edit Format View Help

```
First_Name,Last_Name,Grade,Location,ba,ms
Alan,Brown,GR1,DELHI,17990,16070
Agatha,Williams,GR2,MUMBAI,12390,6630
Rajesh,Kolte,GR1,MUMBAI,19250,14960
Ameet,Mishra,GR2,DELHI,14780,9300
Neha,Rao,missing,MUMBAI,19235,15200
Sagar,Chavan,GR2,MUMBAI,13390,6700
Aaron,Jones,GR1,MUMBAI,23280,13490
John,Patil,GR2,MUMBAI,13500,10760
Sneha,Joshi,GR1,DELHI,20660,missing
Gaurav,Singh,GR2,DELHI,13760,13220
Adela,Thomas,GR2,DELHI,13660,6840
Anup,Save,GR2,MUMBAI,11960,7880
```

File Edit Format View Help

```
First_Name      Last_Name       Grade   Location        ba      ms
Alan    Brown   GR1     DELHI   17990   16070
Agatha  Williams        GR2     MUMBAI  12390   6630
Rajesh  Kolte   GR1     MUMBAI  19250   14960
Ameet   Mishra  GR2     DELHI   14780   9300
Neha    Rao     missing MUMBAI  19235   15200
Sagar   Chavan  GR2     MUMBAI  13390   6700
Aaron   Jones   GR1     MUMBAI  23280   13490
John    Patil   GR2     MUMBAI  13500   10760
Sneha   Joshi   GR1     DELHI   20660   missing
Gaurav  Singh   GR2     DELHI   13760   13220
Adela   Thomas  GR2     DELHI   13660   6840
Anup    Save    GR2     MUMBAI  11960   7880
```

Using Pandas Library

- To import files of different formats
- pandas is a Python Package providing **high-performance, easy-to-use data structures**
- **data analysis tools**
- Pandas will be our preferred library for **data management**.

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.

read_excel() Function

Importing a .xlsx file

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

sheetname= defines the excel sheet to import data from. Takes Sheet1 by default. ”Sheet_name” indicates the name of the sheet to be imported. number indicates the index number of the sheet to be imported. Starts from 0.
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.
index_col= number, helpful if you want to define a specific column as data index.

*

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_excel() Function

Importing multiple sheets from .xlsx file

- To facilitate working with multiple sheets from the same file, the **pd.ExcelFile** can be used to wrap the file and can be passed into **read_excel()**

ExcelFile class is best for importing sheets using different arguments as specification.

```
salary_data={}
with pd.ExcelFile("C:/Users/Documents/basic_salary.xlsx") as xls:
salary_data['Sheet1'] = pd.read_excel(xls, 'Sheet1', index_col=None)
salary_data['Sheet2'] = pd.read_excel(xls, 'Sheet2', index_col=1)
```

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')
```

Using SQLAlchemy in Pandas

- To import files from different database management systems that require different API's such as MySQL, PostgreSQL and Oracle
- Pandas uses a database abstractor SQLAlchemy
- SQLAlchemy uses the `create_engine` function to create an Engine object
- To create this object, SQLAlchemy uses the most prominent DBAPI driver
-

create_engine() Function

Create a MySQL, PostgreSQL and Oracle file

```
import pandas as pd
from sqlalchemy import create_engine

#Create Engine
#PostgreSQL
engine =
create_engine('postgresql://scott:tiger@localhost/mydatabase')

#MySQL
engine = create_engine('mysql://scott:tiger@localhost/foo')

#Oracle
engine = create_engine('oracle://scott:tiger@127.0.0.1:1521/sidname')
```

Remember, the create function should point to the address of your database which includes your username for the database, the IP/hosting address, password as well as your database name.

SQL Functions in SQLAlchemy

Importing a MySQL, PostGreSQL, Oracle database file

```
salary_data = pd.read_sql_table('data', engine)
salary_data = pd.read_sql_query('SELECT * FROM data', engine)
```

- **'data'** here is the name of the table being imported.
- **'SELECT * FROM data'** is the SQL query being passed to get the data.

Exporting a file to an SQL database

```
salary_data.to_sql('name of database', engine)
```

Importing SAS files in Pandas

- Pandas can read SAS files but not write SAS xport (.XPT) and SAS7BDAT (.sas7bdat) format files.
- SAS files only contains two value types: ASCII text and floating point values (usually 8 bytes but sometimes truncated). For xport files, there is no automatic type conversion to integers, dates, or categoricals. For SAS7BDAT files, the format codes may allow date variables to be automatically converted to dates.

```
import pandas as pd
salary_data = pd.read_sas('sas_data.sas7bdat')
```

Importing STATA files in Pandas

Just like in the case of SAS files, Pandas can read STATA files but not write them, thus they cannot be exported as a STATA file.

```
import pandas as pd
salary_data = pd.read_stata('file.stata')
```

- **convert_categoricals** indicates whether value labels should be read and used to create a Categorical variable from them.
- **convert_missing** indicates whether missing value representations in Stata should be preserved. If **False** (the default), missing values are represented as np.nan. If **True**, missing values are represented using **StataMissingValue** objects, and columns containing missing values will have object data type.

Quick Recap

Import Files Using Pandas	• **`read_table`** is used for importing csv and txt files • **`read_csv`** is used specifically for importing csv files and is more efficient than **`read_table()`** • **`read_excel`** is used to import an Excel file.
Handle Missing Values	• Use **`na_values`** to explicitly define what values are to be considered null. • Use **`skip_blank_lines`** to skip rows with blank spaces.
Export Files Using Pandas	• **`to_csv`** exports csv files • **`to_csv(sep= '\t')`** exports tab delimited txt files • **`to_excel`** exports excel file.

Quick Recap

Importing & Exporting MySQL, PostGreSQL and Oracle files	• DBAPI wrapper SQLAlchemy is used • **`read_sql_table()`** imports the database while **`read_sql_query()`** imports the results of the specified query on the database. • **`data.to_sql()`** is used to read a file to an sql database
Import SAS files	• Use **`read_sas()`** function in native pandas. These files cannot be exported.
Import STATA files	• Use **`read_stata()`** function in native pandas. These files cannot be exported.