How to read data from different text based and non-text based sources

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491h

Enter your authorization code:
··········
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
```

Exercise 1: Read data from a CSV

```
df1 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_1.csv")
```

```
df1
```

	Bedroom	Sq. foot	Locality	Price ($)

Exercise 2: Read data from a CSV where headers are missing

```
df2 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_2.csv")
df2
```

	2	1500	Good	300000
0	3	1300	Fair	240000
1	3	1900	Very good	450000
2	3	1850	Bad	280000
3	2	1640	Good	310000

```
df2 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_2.csv",header=None)
df2
```

	0	1	2	3
0	2	1500	Good	300000
1	3	1300	Fair	240000
2	3	1900	Very good	450000
3	3	1850	Bad	280000
4	2	1640	Good	310000

```
df2 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_2.csv",header=None, names=['Be
df2
```

	Bedroom	Sq.ft	Locality	Price($)
0	2	1500	Good	300000
1	3	1300	Fair	240000
2	3	1900	Very good	450000
3	3	1850	Bad	280000

Exercise 3: Read data from a CSV where delimiters/separators are not comma

```
df3 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_3.csv")
df3
```

	Bedroom; Sq. foot; Locality; Price ($)
0	2; 1500; Good; 300000
1	3; 1300; Fair; 240000
2	3; 1900; Very good; 450000
3	3; 1850; Bad; 280000
4	2; 1640; Good; 310000

```
df3 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_3.csv",sep=';')
df3
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2	1500	Good	300000

Exercise 4: How to bypass given headers with your own?

2	3	1900	Very good	450000

```
df4 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_1.csv",names=['A','B','C','D']
df4
```

	A	B	C	D
0	Bedroom	Sq. foot	Locality	Price ($)
1	2	1500	Good	300000
2	3	1300	Fair	240000
3	3	1900	Very good	450000
4	3	1850	Bad	280000
5	2	1640	Good	310000

```
df4 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_1.csv",header=0,names=['A','B'
df4
```

	A	B	C	D
0	2	1500	Good	300000
1	3	1300	Fair	240000
2	3	1900	Very good	450000
3	3	1850	Bad	280000
4	2	1640	Good	310000

Exercise 5: Skip initial rows

```
df5 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_skiprows.csv")
df5
```

	Filetype: CSV	Unnamed: 1	Unnamed: 2	Unnamed: 3
0	NaN	Info about some houses	NaN	NaN
1	Bedroom	Sq. foot	Locality	Price ($)
2	2	1500	Good	300000
3	3	1300	Fair	240000
4	3	1900	Very good	450000
5	3	1850	Bad	280000
6	2	1640	Good	310000

```
df5 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_skiprows.csv",skiprows=2)
df5
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2	1500	Good	300000
1	3	1300	Fair	240000
2	3	1900	Very good	450000
3	3	1850	Bad	280000
4	2	1640	Good	310000

Exercise 6: Skip footers

```
df6 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_skipfooter.csv")
df6
```

	Filetype: CSV	Unnamed: 1	Unnamed: 2	Unnamed: 3
0	NaN	Info about some houses	NaN	NaN
1	Bedroom	Sq. foot	Locality	Price ($)
2	2	1500	Good	300000
3	3	1300	Fair	240000
4	3	1900	Very good	450000
5	3	1850	Bad	280000
6	2	1640	Good	310000
7	NaN	This is the end of file	NaN	NaN

```
df6 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_skipfooter.csv",skiprows=2,ski
df6
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2	1500	Good	300000
1	3	1300	Fair	240000
2	3	1900	Very good	450000
3	3	1850	Bad	280000
4	2	1640	Good	310000

Exercise 7: Read only first *n* rows (especially useful for large files)

```
df7 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_1.csv",nrows=2)
```

```
df7
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2	1500	Good	300000
1	3	1300	Fair	240000

Exercise 8: How to combine `skiprows` and `nrows` to read data in small chunks

```
# List where DataFrames will be stored
list_of_dataframe = []
# Number of rows to be read in one chunk
rows_in_a_chunk = 10
# Number of chunks to be read (this many separate DataFrames will be produced)
num_chunks = 5
# Dummy DataFrame to get the column names
df_dummy = pd.read_csv("/content/drive/My Drive/pandas/Boston_housing.csv",nrows=2)
colnames = df_dummy.columns
# Loop over the CSV file to read only specified number of rows at a time
# Note how the iterator variable i is set up inside the range
for i in range(0,num_chunks*rows_in_a_chunk,rows_in_a_chunk):
    df = pd.read_csv("/content/drive/My Drive/pandas/Boston_housing.csv",header=0,skipr
    list_of_dataframe.append(df)

list_of_dataframe[0]
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9

```
list_of_dataframe[1]
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
1	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
2	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7
3	0.62976	0.0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4
4	0.63796	0.0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2
5	0.62739	0.0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21.0	395.62	8.47	19.9
6	1.05393	0.0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21.0	386.85	6.58	23.1
7	0.78420	0.0	8.14	0	0.538	5.990	81.7	4.2579	4	307	21.0	386.75	14.67	17.5
8	0.80271	0.0	8.14	0	0.538	5.456	36.6	3.7965	4	307	21.0	288.99	11.69	20.2
9	0.72580	0.0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21.0	390.95	11.28	18.2

Exercise 9: Setting the option `skip_blank_lines`

```
df9 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_blankline.csv")
df9
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2	1500	Good	300000
1	3	1300	Fair	240000
2	3	1900	Very good	450000
3	3	1850	Bad	280000
4	2	1640	Good	310000

```
df9 = pd.read_csv("/content/drive/My Drive/pandas/CSV_EX_blankline.csv",skip_blank_line
df9
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2.0	1500.0	Good	300000.0
1	3.0	1300.0	Fair	240000.0
2	NaN	NaN	NaN	NaN
3	3.0	1900.0	Very good	450000.0
4	3.0	1850.0	Bad	280000.0
5	NaN	NaN	NaN	NaN
6	2.0	1640.0	Good	310000.0

Exercise 10: Read CSV from inside a compressed (.zip/.gz/.bz2/.xz) file

```
df10 = pd.read_csv('/content/drive/My Drive/pandas/CSV_EX_1.zip')
df10
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2	1500	Good	300000
1	3	1300	Fair	240000
2	3	1900	Very good	450000
3	3	1850	Bad	280000
4	2	1640	Good	310000

Exercise 11: Reading from an Excel file - how to use `sheet_name`

```
df11_1 = pd.read_excel("/content/drive/My Drive/pandas/Housing_data.xlsx",sheet_name='D
df11_2 = pd.read_excel("/content/drive/My Drive/pandas/Housing_data.xlsx",sheet_name='D
df11_3 = pd.read_excel("/content/drive/My Drive/pandas/Housing_data.xlsx",sheet_name='D
```

```
df11_1.shape
```

```
(9, 14)
```

```
df11_2.shape
```

```
(4, 14)
```

```
df11_3.shape
```

```
(16, 14)
```

Exercise 12: If `sheet_name` is set to `None` then an Ordered Dictionary of DataFrame is returned if the Excel

```
dict_df = pd.read_excel("/content/drive/My Drive/pandas/Housing_data.xlsx",sheet_name=N

dict_df.keys()
```

```
dict_keys(['Data_Tab_1', 'Data_Tab_2', 'Data_Tab_3'])
```

Exercise 13: General delimated text file can be read same as a CSV

```
df13 = pd.read_table("/content/drive/My Drive/pandas/Table_EX_1.txt")
df13
```

	Bedroom, Sq. foot, Locality, Price ($)
0	2, 1500, Good, 300000
1	3, 1300, Fair, 240000
2	3, 1900, Very good, 450000
3	3, 1850, Bad, 280000
4	2, 1640, Good, 310000

```
df13 = pd.read_table("/content/drive/My Drive/pandas/Table_EX_1.txt",sep=',')
df13
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2	1500	Good	300000

```
df13 = pd.read_table("/content/drive/My Drive/pandas/Table_tab_separated.txt",)
df13
```

	Bedroom	Sq. foot	Locality	Price ($)
0	2	1500	Good	300000
1	3	1300	Fair	240000
2	3	1900	Very good	450000
3	3	1850	Bad	280000
4	2	1640	Good	310000

Exercise 14: Read HTML tables directly from an URL

```
#! pip3 install lxml

url = 'http://www.fdic.gov/bank/individual/failed/banklist.html'
list_of_df = pd.read_html(url)

df14 = list_of_df[0]
df14.head()
```

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date
0	The First State Bank	Barboursville	WV	14361	MVB Bank, Inc.	April 3, 2020
1	Ericson State Bank	Ericson	NE	18265	Farmers and Merchants Bank	February 14, 2020

Exercise 15: Mostly, `read_html` returns more than one table and further wrangling is needed to get the desired data

```
list_of_df = pd.read_html("https://en.wikipedia.org/wiki/2016_Summer_Olympics_medal_tab
```

```
len(list_of_df)
```

```
7
```

```
for t in list_of_df:
    print(t.shape)
```

```
(1, 1)
(87, 6)
(10, 9)
(0, 2)
(1, 2)
(4, 2)
(1, 2)
```

```
df15=list_of_df[1]
df15.head()
```

	Rank	NOC	Gold	Silver	Bronze	Total
0	1	United States (USA)	46	37	38	121
1	2	Great Britain (GBR)	27	23	17	67

Exercise 16: Read in a JSON file

4	5	Germany (GER)	17	10	15	42

```
df16 = pd.read_json("/content/drive/My Drive/pandas/movies.json")
```

```
df16.head()
```

	title	year	cast	genres
0	After Dark in Central Park	1900	[]	[]
1	Boarding School Girls' Pajama Parade	1900	[]	[]
2	Buffalo Bill's Wild West Parad	1900	[]	[]
3	Caught	1900	[]	[]
4	Clowns Spinning Hats	1900	[]	[]

```
df16.tail()
```

	title	year	cast	genres
28790	Bumblebee	2018	[Hailee Steinfeld, John Cena, Jorge Lendeborg ...	[Action, Adventure, Science Fiction]
28791	Welcome to Marwen	2018	[Steve Carell, Leslie Mann, Diane Kruger, Falk...	[Fantasy, Drama]
28792	Holmes and Watson	2018	[Will Ferrell, John C. Reilly, Rebecca Hall, R...	[Action, Mystery, Comedy]
28793	On the Basis of Sex	2018	[Felicity Jones, Armie Hammer, Justin Theroux,...	[Biography, Drama]
28794	Destroyer	2018	[Nicole Kidman, Tatiana Maslany, Sebastian Sta...	[Crime, Thriller]

```
df16[df16['title']=="The Avengers"]['cast']
```

```
13519                          [Adele Mara, John Carroll]
23778    [Ralph Fiennes, Uma Thurman, Sean Connery, Jim...
27195    [Robert Downey, Jr., Chris Evans, Mark Ruffalo...
Name: cast, dtype: object
```

```
cast_of_avengers=df16[(df16['title']=="The Avengers") & (df16['year']==2012)]['cast']
```

```
print(list(cast_of_avengers))
```

```
[['Robert Downey, Jr.', 'Chris Evans', 'Mark Ruffalo', 'Chris Hemsworth', 'Scarlett Johansson', 'Jeremy Renner', 'Tom H
```

Exercise 17: Read Stata file (.dta)

```
df17 = pd.read_stata("/content/drive/My Drive/pandas/rscfp2016.dta")
```

```
df17.head()
```

	YY1	Y1	wgt	hhsex	age	agecl	educ	edcl	married	kids	lf	lifecl	famstruct	racecl	RACECL4	race	OCC
0	1	11	6427.136676	2	71	5	10	3	2	0	0	6	3	1	1	1	
1	1	12	6428.350592	2	71	5	10	3	2	0	0	6	3	1	1	1	
2	1	13	6414.477294	2	71	5	10	3	2	0	0	6	3	1	1	1	
3	1	14	6428.487972	2	71	5	10	3	2	0	0	6	3	1	1	1	
4	1	15	6425.256822	2	71	5	10	3	2	0	0	6	3	1	1	1	

5 rows × 348 columns

Exercise 18: Read tabular data from PDF file

```
! pip install tabula-py
```

```
Collecting tabula-py
  Downloading https://files.pythonhosted.org/packages/8d/ed/20655a47a603430272c995d908d0dd96f93c2aa8973c8a55a66c8f3b8df
     |████████████████████████████████| 10.4MB 570kB/s
Collecting distro
  Downloading https://files.pythonhosted.org/packages/25/b7/b3c4270a11414cb22c6352ebc7a83aaa3712043be29daa05018fd5a5c95
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from tabula-py) (1.18.5)
Requirement already satisfied: pandas>=0.25.3 in /usr/local/lib/python3.6/dist-packages (from tabula-py) (1.0.5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.25.3->tabula-py)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.25.3->t
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.6.1->pandas>
Installing collected packages: distro, tabula-py
Successfully installed distro-1.5.0 tabula-py-2.1.1
```

```
from tabula import read_pdf
```

```
df18_1 = read_pdf('/content/drive/My Drive/pandas/Housing_data.pdf',pages=[1],pandas_op
```

```
#df18_1 =pd.DataFrame(df18_1)
df18_1[0]
```

	0	1	2	3	4	5	6	7	8	9
0	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311
1	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311
2	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311
3	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311

```
df18_2 = read_pdf('/content/drive/My Drive/pandas/Housing_data.pdf',pages=[2],pandas_op
```

```
df18_2[0]
```

	0	1	2	3
0	15.2	386.71	17.10	18.9
1	15.2	392.52	20.45	15.0
2	15.2	396.90	13.27	18.9
3	15.2	390.50	15.71	21.7

```
df18=pd.concat([df18_1[0],df18_2[0]],axis=1)
```

```
df18
```

	0	1	2	3	4	5	6	7	8	9	0	1	2	3
0	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9
1	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
2	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
3	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7

With PDF extraction, most of the time, headres will be difficult to extract automatically. You have to pass on the list of headres as the `names` argument in the `read-pdf` function as `pandas_option`,

```
names=['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO','B','LST
```

```
df18_1 = read_pdf('/content/drive/My Drive/pandas/Housing_data.pdf',pages=[1],pandas_op
```

```
df18_2 = read_pdf('/content/drive/My Drive/pandas/Housing_data.pdf',pages=[2],pandas_op
df18=pd.concat([df18_1[0],df18_2[0]],axis=1)

df18
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9
1	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
2	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
3	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7

Exercise 19: In a complex page, you may have multiple tables and use Tabula to extract a list of DataFrames and then process the DataFrames further

5 cells hidden