**Guided Lab - 343.3.2 - Read CSV into Panda DataFrame**

# 

# Introduction:

Use the pandas **read\_csv()** function to read a CSV file (comma-separated) into a Python pandas DataFrameDataFrame. which supports options to read any delimited file. In this pandas article,

# 

# Lab Overview:

In this lab, we will demonstrate how to read a CSV file with or without a header, skip rows, skip columns, set columns to index, handle missing data, and many more with examples.

**Learning Objective**:

By the end of this lab, learners will be able to utilize the CSV file using Panda dataframe.

**Dataset:**

In this lab we will utilize the dummy employee dataset.

* [Click here to download employee dataset (employee.csv)](https://drive.google.com/file/d/14RV1xKIRzWS166LtGqnPC1Wg7eTlI_y1/view?usp=share_link).

# 

# Example 1: Reading Data from CSVs

| import pandas as pd  df = pd.read\_csv('employee.csv')  df |
| --- |

***Note: if you get error, use the line below:***

#df = pd.read\_csv('employee.csv', on\_bad\_lines='skip')

**Result:**

**Name Age Weight Salary**

0 James 36.0 75.0 5428000.0

1 Villers 38.0 74.0 3428000.0

2 VKole 31.0 70.0 8428000.0

3 Smith 34.0 80.0 4428000.0

4 Gayle 40.0 100.0 4528000.0

5 Adam 40.0 NaN 4528000.0

6 Rooter 33.0 72.0 7028000.0

7 Peterson 42.0 85.0 2528000.0

8 lynda 42.0 85.0 NaN

9 NaN 42.0 85.0 NaN

10 Jenny NaN 100.0 25632.0

11 Kenn NaN 110.0 25632.0

12 Aly NaN 90.0 25582.0

13 John 41.0 85.0 1528000.0

14 Ali 26.0 69.0 NaN

Note: Use the **sep** or **delimiter** argument to specify the separator of the columns. By default, it uses a comma.

# 

# Example 2: Viewing or Explore your Data

The first thing to do when opening a new dataset is to print out a few rows to keep as a visual reference. We accomplish this with .head():

| df.head() |
| --- |

.head() outputs the **first five rows** of your DataFrame by default, but we could also pass a number as well. df.head(10) would output the top ten rows.

| df.head(10) |
| --- |

To see the **last five rows**, use df.tail(), which also accepts a number and prints the bottom two rows in this case.

| df.tail(2) |
| --- |

**Result:**

| **Name** | **Age** | **Weight** | **Salary** |
| --- | --- | --- | --- |
| John | 41.0 | 85.0 | 1528000.0 |
| Ali | 26.0 | 69.0 | NaN |

# 

# Example 3: Getting Information About your Data

.info() should be one of the very first commands you run after loading your data:

| df.info() |
| --- |

**Result:**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 15 entries, 0 to 14

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Name 14 non-null object

1 Age 12 non-null float64

2 Weight 14 non-null float64

3 Salary 12 non-null float64

dtypes: float64(3), object(1)

memory usage: 488.0+ bytes

.info() provides the essential details about your dataset such as the number of rows and columns, the number of non-null values, what type of data is in each column, and how much memory your DataFrame is using.

Another fast and useful attribute is .shape, which returns just a tuple of (rows, columns):

| df.shape |
| --- |

**Result:**

(15, 4)

Note that .shape has no parentheses and is a simple tuple format (rows, columns). So, we have 15 rows and 4 columns in our employeeDataFrame.

# 

# Example 4: **Skip Rows**

Sometimes, you may need to skip the first row or skip the footer rows. To do this, use **skiprows** and **skipfooter** params, respectively.

| *# Skip first few rows*  df = pd.read\_csv('employee.csv', header=None, skiprows=5)  print(df) |
| --- |

**Result:**

0 1 2 3

0 Gayle 40.0 100.0 4528000.0

1 Adam 40.0 NaN 4528000.0

2 Rooter 33.0 72.0 7028000.0

3 Peterson 42.0 85.0 2528000.0

4 lynda 42.0 85.0 NaN

5 NaN 42.0 85.0 NaN

6 Jenny NaN 100.0 25632.0

7 Kenn NaN 110.0 25632.0

8 Aly NaN 90.0 25582.0

9 John 41.0 85.0 1528000.0

10 Ali 26.0 69.0 NaN

# 

# Example 5: Load Only Selected Columns

There are two common ways to use this argument:

**Method 1:** Use usecols with Column Names

**df = pd.read\_csv('my\_data.csv', usecols=['column name one', 'column name two'])**

**Method 2:** Use usecols with Column Positions

**df = pd.read\_csv('my\_data.csv', usecols=[0, 2])**

| df = pd.read\_csv('employee.csv', usecols =[0,3])  print(df) |
| --- |

**Result:**

**Name Salary**

0 James 5428000.0

1 Villers 3428000.0

2 VKole 8428000.0

3 Smith 4428000.0

4 Gayle 4528000.0

5 Adam 4528000.0

6 Rooter 7028000.0

7 Peterson 2528000.0

8 lynda NaN

9 NaN NaN

10 Jenny 25632.0

11 Kenn 25632.0

12 Aly 25582.0

13 John 1528000.0

14 Ali NaN

# 

# Example 6: **Set DataTypes to Columns**

By default, read\_csv() assigns the data type that best fits the data. We can find data type of the columns by using **df.dtypes**

| df = pd.read\_csv('employee.csv')  print(df.dtypes) |
| --- |

**Result:**

Name object

Age float64

Weight float64

Salary float64

dtype: object

Let’s change the ***Name*** columns to the ***String*** type.

| *# Set column data types*  df = pd.read\_csv('employee.csv', dtype={'Name':'string' })  print(df.dtypes) |
| --- |

**Result:**

Name string

Age float64

Weight float64

Salary float64

dtype: object

# 

# Example 7: Handling Missing Data or NaN VALUE

As we have learned in the previous lesson, the fillna() method can be used to deal with NaN values.

| df = pd.read\_csv('employee.csv', dtype={'Name':'string' })  df2 = df.fillna(value={'Name':'Verification Pending','Age':"Unknown", 'Weight': "pending", 'Salary': 0.0})  print(df2) |
| --- |

**Result:**

Name Age Weight Salary

0 James 36.0 75.0 5428000.0

1 Villers 38.0 74.0 3428000.0

2 VKole 31.0 70.0 8428000.0

3 Smith 34.0 80.0 4428000.0

4 Gayle 40.0 100.0 4528000.0

5 Adam 40.0 pending 4528000.0

6 Rooter 33.0 72.0 7028000.0

7 Peterson 42.0 85.0 2528000.0

8 lynda 42.0 85.0 0.0

9 Verification Pending 42.0 85.0 0.0

10 Jenny Unknown 100.0 25632.0

11 Kenn Unknown 110.0 25632.0

12 Aly Unknown 90.0 25582.0

13 John 41.0 85.0 1528000.0

14 Ali 26.0 69.0 0.0

# 

# Example 8: Pandas Read Multiple CSV Files into DataFrame

# 

Sometimes you may need to read or import multiple CSV files from a folder or from a list of files and convert them into a Pandas DataFrame. You can do this by reading each CSV file into a DataFrame and appending or concatenating the DataFrames to create a single DataFrame with data from all files.

When you want to read multiple CSV files that exist in different folders, first create a list of strings with absolute paths and use it as shown below to load all CSV files and create one big Pandas DataFrame.

| # Read CSV files from List  df = pd.concat(map(pd.read\_csv, ['car\_data1.csv', 'car\_data2.csv','car\_data3.csv']))  df |
| --- |

**Result:**

Car MPG Cylinders Displacement Horsepower \

0 AMC Ambassador Brougham 13.0 8 360.0 175

1 AMC Ambassador DPL 15.0 8 390.0 190

2 AMC Ambassador SST 17.0 8 304.0 150

3 AMC Concord 19.4 6 232.0 90

4 AMC Concord 24.3 4 151.0 90

.. ... ... ... ... ...

156 Mercury Capri v6 21.0 6 155.0 107

157 Mercury Cougar Brougham 15.0 8 302.0 130

158 Mercury Grand Marquis 16.5 8 351.0 138

159 Mercury Lynx l 36.0 4 98.0 70

160 Mercury Marquis 11.0 8 429.0 208

Weight Acceleration Model Origin quantity city

0 3821 11.0 73 US 25 NewYork

1 3850 8.5 70 US 2 NJ

2 3672 11.5 72 US 4 DALLAS

3 3210 17.2 78 US 52 TEXAS

4 3003 20.1 80 US 42 OH

.. ... ... ... ... ... ...

156 2472 14.0 73 US 158 NewYork

157 4295 14.9 77 US 27 NJ

158 3955 13.2 79 US 332 DALLAS

159 2125 17.3 82 US 425 TEXAS

160 4633 11.0 72 US 112 OH

[260 rows x 11 columns]

*Note that by default, the concat() method performs an append operation, which means it appends each DataFrame at the end of another DataFrame and creates a single DataFrame.*

**Submission Instructions:**

Include the following deliverables in your submission:

* Submit your code and screenshots using the Start Assignment button in the top-right corner of the assignment page in Canvas.

**For Canvas official Team**

| **Instructions for Canvas Assignment Creation** |
| --- |
| **Assignment Name: GLAB - 343.1.5 - Read CSV into Panda DataFrame**  **Points:** **100**  **Assignment Group: Module 343 - Data Analytics with Python**  **Display Grade As: Complete/Incomplete**  **Do not count this assignment towards the final grade: Checked**  **Submission Types: File Upload**  **Everything else is the default.** |