Pandas Python Module

Computing for Data Analytics (CPSC 4800)

Mourad Bouguerra
mbouguerra@langara.ca

Langara College

June, 2022

Lesson's Outline

1. **Lesson's Learning Objectives**

2. **Pandas Overview**
   - Pandas Installation
   - Pandas Series
   - Pandas DataFrame
     - Reading/Writing Data
     - Basic Data Exploratory Analysis (DEA)
     - Data Filtering

Learning Objectives

Learning Objectives

✎ Upon completion of this lesson, you will be able to

- ❒ generate Pandas Series and DataFrames
- ❒ use read data from different sources into a Pandas DataFrame
- ❒ access and filter Pandas DataFrame rows and columns using different methods

Pandas Module

- Pandas Python module provides
  - fast, flexible, and efficient data structures that enables
    - ✔ easy anlysis & manipulation of relational/tabular[a] data
- Pandas Python module provides two data structures
  - Series (1-dimensional)
  - DataFrame (2-dimensional)

[a] Data in the format rows x columns.

Pandas Installation

- Pandas can be installed from terminal using one of the following
  - command-lines

```
python -m pip install pandas
pip install pandas
conda install pandas
```

Using Pandas

Pandas Version

- To check Pandas version

```
pd.__version__
```

- To check Pandas dependencies versions

```
pd.show_versions()
```

Pandas Series

- Pandas series is one-dimensional object that consists of
  - index
  - values
- The index identifies each data point

```
import pandas as pd
series = pd.Series(data=[1,2,3])
print(series)
# Output
# 0    1
# 1    2
# 2    3
# dtype: int64
```

Pandas Series

- Pandas series is one-dimensional object that consists of
  - index
  - values
- The index identifies each data point

```
import pandas as pd
series = pd.Series(data=[1,2,3])
print(f'The index is {series.index}')
print(f'The values are {series.values}')
# Output
# The index is RangeIndex(start=0, stop=3, step=1)
# The values are [1 2 3]
```

Pandas Series

- Series index identifies each data point
  - row names
- If NOT specified, a default index of integers
  - 0 to $len(Series) - 1$

```
import pandas as pd
series = pd.Series(data=[1,2,3],index=list('abc'),\
    dtype='uint8')
print(series)
# Output
# a    1
# b    2
# c    3
# dtype: uint8
```

Pandas Series

- Series index identifies each data point
  - row names
- Can be used to access Series elements

```
import pandas as pd
series = pd.Series(data=[1,2,3],index=list('abc'),\
                   dtype='uint8')
print(f'The index is {series.index}')
print(f'The values are {series.values}')
# Output
# The index is Index(['a', 'b', 'c'], dtype='object')
# The values are [1 2 3]
```

Pandas Series

- To acces Series elements, we can
  - Series index/index slicing

```
print(f'series[1:2]={series[1:2]}')
# Output
# series[1:2]=b    2
# dtype: uint8
print(f"series['a':'b']={series['a':'b']}")
# Output
# a    1
# b    2
# dtype: uint8
```

Class Activity

- Given the following series
  - ✎ What is the Series index?
  - ✎ What is the Series values?
  - ✎ What is the data type of the series elements?

```
import pandas as pd
import random as r
series = pd.Series(data=[r.random() for i in range(10)])
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

- Given the following series
  - ✎ What is the Series index?
  - ✎ What is the Series values?
  - ✎ What is the data type of the series elements?

```
import pandas as pd
alphabet='abcdefghijklmnopqrstuvwxyz'
series = pd.Series([x for x in range(1,11)],\
                   index=list(alphabet[0:10]),\
                   dtype='uint8')
series['d':'i'] = 255
series[-1] = 100
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

- Given the following series
  - ✎ What is the Series index?
  - ✎ What is the Series values?
  - ✎ What is the data type of the series elements?

```
import pandas as pd
alphabet='abcdefghijklmnopqrstuvwxyz'
series = pd.Series([x for x in range(1,11)],\
                   index=list(alphabet[0:10]),\
                   dtype='uint8')
series[1:5] = 255
series[[-1,-3]] = 100
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Pandas Series

- Series can be generated from a dictionary
  - The series index is the dictionary keys set
  - The series values are the dictionary values

```
import pandas as pd
scores = {'quizzes':89.3,'assignments':97.7,\
    'midterm':75.45,'final':99.87}
series = pd.Series(data=scores)
print(series)
# Output
# quizzes        89.30
# assignments    97.70
# midterm        75.45
# final          99.87
# dtype: float64
```

Pandas Series Functions

- To get the unique values of a series use
  - unique() method

```
import pandas as pd
colors = pd.Series(data=['red','red','green','blue',\
    'blue','red','red','green','blue'])
print(f'{colors.unique()}')
# Output
# ['red' 'green' 'blue']
```

Pandas Series Functions

- To get the frequency of each value in a series use
  - value_counts() method

```
print(f'Frequencies\n{colors.value_counts()}')
# Output
# Frequencies
# red       4
# blue      3
# green     2
# dtype: int64
```

Pandas DataFrame

- Pandas DataFrame is two-dimensional object that consists of
  - rows and columns
  - Each column is a Pandas series
  - Each column elements must be of the data type
- Pandas DataFrame has both
  - row index
  - column index
- Pandas DataFrame is generally constructed by reading data

Pandas DataFrame

- Pandas DataFrame can be constructed from a
  - two-dimensial NumPy object

```
import pandas as pd
import string
alphabet = string.ascii_uppercase
df = pd.DataFrame(np.random.randn(1000,4),\
            columns=list('abcd'))
df = pd.DataFrame(np.random.randn(4,10),\
            index=list(alphabet)[:4],\
            columns=list(alphabet[:10]))
```

Class Activity

- Given the following DataFrames
  - ✎ What is the DataFrame rows index?
  - ✎ What is the DataFrame columns index?
  - ✎ What is the content of the DataFrame?

```
import pandas as pd
import string
alphabet = string.ascii_uppercase
df_1 = pd.DataFrame(np.random.randn(10,5))
df_2 = pd.DataFrame(np.random.randn(10,5),\
            index=list(alphabet)[:10])
df_3 = pd.DataFrame(np.random.randn(5,10),\
            index=list(alphabet)[:5],\
            columns=list(alphabet[:10]))
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

- What is the output of the following Python script?

```
df_1 = pd.DataFrame([np.arange(5),np.arange(5,10),np.arange
    (10,15)])
df_2 = pd.DataFrame(np.arange(15).reshape(3,5))
df_3 = pd.DataFrame(np.arange(15).reshape(5,3))
print(f'df_1=\n{df_1}')
print(f'df_2=\n{df_2}')
print(f'df_3=\n{df_3}')
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Pandas DataFrame

- To add a new column to a Pandas DataFrame use
  - df['NewColumnName']=List

```
df = pd.DataFrame(np.random.randn(4,10))
df['NewColumn']=['One','Two','Three','Four']
```

Pandas DataFrame

- To remove a column from Pandas DataFrame use
  - df.drop('ColumnName', inplace=True,axis=1)

```
df = pd.DataFrame(np.random.randn(100,4),\
        columns=['One','Two','Three','Four'])
# remove column Three
df.drop('Three', inplace=True, axis=1)
```

Class Activity

- What is the output of the following Python script?

```
df = pd.DataFrame(np.arange(20).reshape(4,5))
df['Key']=['One','Two','Three','Four']
print(f'df=\n{df}')
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Reading/Writing Data

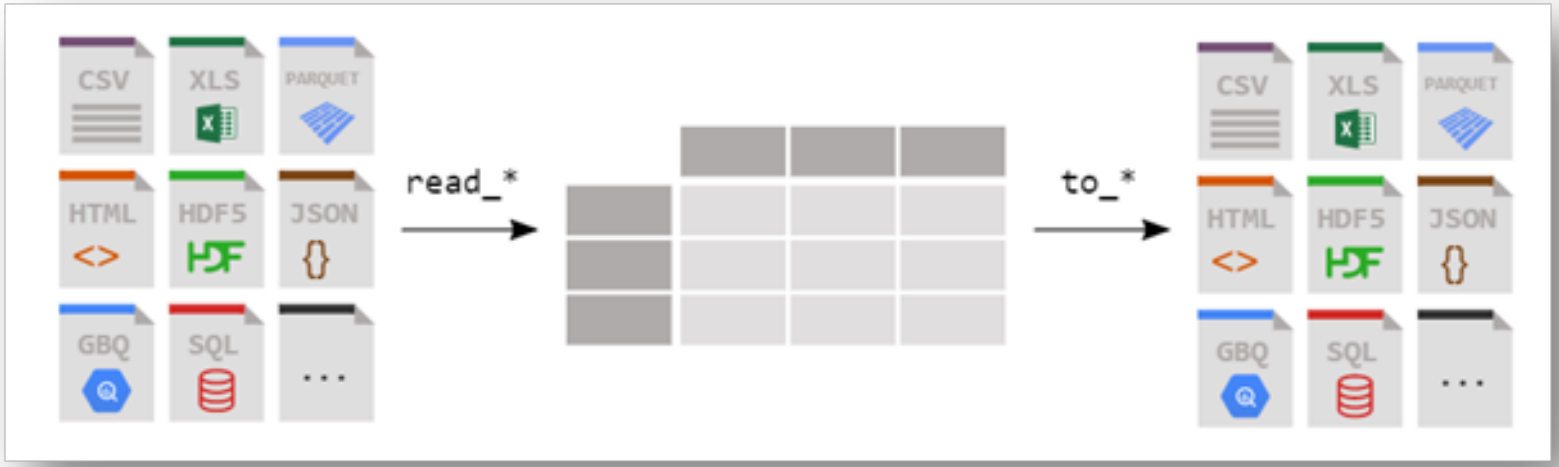

Figure: Pandas Reading/Writing*

*https://pandas.pydata.org/docs/getting_started/intro_tutorials/02_read_write.html

Class Activity

- Using **list comprehension**, display all the Pandas read functions
  - All Pandas read functions start with read
- Using **list comprehension**, display all the Pandas write functions
  - All Pandas write functions start with to

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Reading Data

Pandas DataFrame

- Pandas DataFrame can be constructed by reading different data format
  - CSV
  - Excel
  - JSON
  - HTML tables

```
import pandas as pd
df = pd.read_csv('data/arh.csv')
df = pd.read_table('data/arh.csv', sep=',')
df = pd.read_excel('data/arh.csv')
df = pd.pd.read_clipboard()
```

Class Activity

- Using **list comprehension**, display all the DataFrame write functions
  - All DataFrame write functions start with to

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Pandas DataFrame

- To check the size of the file in disk and memory

```
import pandas as pd
import os
MB = 2**20
filename = 'data/arh.csv'
df = pd.read_csv(filename)
size_in_disk = os.stat(filename).st_size/MB
size_in_ram = df.memory_usage('deep').sum()/MB
print(f'The size of the file in disk is {size_in_disk:.2f} MB.')
print(f'The size of the file in ram is {size_in_ram:.2f} MB.')
```

Pandas DataFrame

- To display rows/columns index of the Pandas DataFrame use
  - index property
  - columns property
- To get the numbers of rows and column of the Pandas DataFrame use
  - shape property

```
df.index
df.columns
df.shape
```

Reading Data

Pandas DataFrame

❒ To display the first/last rows of the Pandas DataFrame use

➡ head()

➡ tail

```
import pandas as pd
df = pd.read_csv('data/arh.csv')
df.head() # first five rows
df.tail() # last five rows
df.head(15) # first 15 rows
df.tail(20) # last 20 rows
```

Pandas DataFrame

- To get detailed information about each column of the Pandas DataFrame use
  - info() function
- To get RAM usage of each column of the Pandas DataFrame use
  - memory_usage('deep') function
- To get the five-number summary of the numerical columns of the Pandas DataFrame use
  - describe() function[a]

```
df.info()
df.memory_usage('deep')
df.describe()
```

[a] It also computes the count, mean, and standard deviation.

Pandas DataFrame

- To check for missing values in Pandas DataFrame use
  - isna() function
- To display the number of missing values in each column use
  - isna().sum() function[a]

```
df.isna()
df.isna().sum()
```

[a] It also computes the count, mean, and standard deviation.

Pandas DataFrame

- To check for duplicate rows in Pandas DataFrame use
  - duplicated() function
- To remove missing values rows use
  - dropna() function
- To remove duplicate rows use
  - drop_duplicates() function

```
df.duplicated()
clean_df = df.dropna()
clean_df = df.drop_duplicates()
```

Data Filtering

Pandas Data Filtering

- To select rows or columns from a Pandas DataFrame use the methods
  - loc()
  - iloc()

Pandas DataFrame iloc() Method

- Pandas loc() is used to select rows and/or columns by
  - integer position of the rows and columns index
- In slicing, the end is NOT included

```
# To select the first 5 rows and the second & third
    columns
movies.iloc[0:5,1:3]
# To select last three rows with all the columns
movies.iloc[-3:,:]
movies.iloc[-3:]
# This will generate an error
movies.iloc[:,['title','star_rating']]
# .iloc requires numeric indexers
```

Class Activity

- What does the following Python code accomplish?

```
movies.index
movies.columns
movies.iloc[-5:]
movies.iloc[:,2].unique()
movies.iloc[:,3].value_counts()
movies.iloc[[x for x in movies.index if x % 2 == 1],\
            [1,0]]
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Pandas DataFrame loc() Method

- Pandas loc() is used to select rows and/or columns by
  - the index label
- Both sides of the slicing are included in loc()

```
# To select the first fourth rows with all the columns
movies.loc[0:3,:]
# To select all the rows for the title column
movies.loc[:,'title']
# To select all the rows for the title and star_rating
    columns
movies.loc[:,['title','star_rating']]
# This will generate an error
movies.loc[:,[1,0]]
```

Class Activity

- What does the following Python code accomplish?

```
movies.loc[975:978]
movies.loc[:,'genre'].unique()
movies.loc[:,'content_rating'].value_counts()
movies.loc[[x for x in movies.index if x % 2 == 1],\
            ['title','star_rating']]
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Filtering Using Boolean Conditions

- To filter Pandas DataFrame by column value use
  - a Boolean condition on the column value
- dot operator can be only used if the column name
  - a valid Python variable name

```
# To filter only action movies
condition = movies.genre == 'Action'
# or
condition = movies['genre'] == 'Action'
movies[condition]
# or
movies.loc[condition,:]
```

Filtering Using Boolean Conditions

- To filter Pandas DataFrame by column value use
  - a Boolean condition on the column value
- dot operator can be only used if the column name
  - a valid Python variable name

```
# To filter only movies of more than 2 hours
long_movies = movies.duration >= 120
# or
long_movies = movies['duration'] >= 120
movies[long_movies] # with all columns
movies.loc[long_movies,'star_rating':'duration']
```

Filtering Using Boolean Conditions

- To combine multiple Boolean conditions
  - a Boolean condition on the column value
- and using the operator &
- or using the operator |
  - a valid Python variable name

```
# To filter only Drama genre movies
# with R content rating
condition = (movies.genre == 'Drama') &\
            (movies.content_rating == 'R')
movies[condition]
movies.loc[:,condition]
```

Class Activity

- ❒ Filter all the drama movies that have a duration more than 200 minutes
- ❒ Filter all the movies that have a duration more than 200 minutes or less than 90 minutes
- ❒ What are the top 10 rated movies?
- ❒ What are the worst 10 rated movies?

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn