Python Tutorial – Part I: Introduction

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 688P, Spring Semester 2022

January 18, 2023

Overview

Part 5

Data and

Dataset Transformation

(Pandas)

Working with Pandas

Introduction to Pandas

Pandas is an open source Python Library that supports working and analysis of tabular data sets.

Benefits:

- Pandas can clean messy data sets, and make them readable and relevant.
- Pandas allows us to analyze large data sets and make conclusions based on statistical theories.
- Three data structures: Series, DataFrame and Panel.

Installation:

```
prompt >> pip3 install pandas
```

What can Pandas do?

Basic Operations:

- Create series and dataframes.
- Read CSV and JSON files.
- Plot data.

Clean Data:

- Clean empty cells.
- Fix wrong format.
- Remove duplicates.

Advanced Operations:

- Combine (concatenate, join, merge) Panda objects.
- Compute correlations.

Panda Series and DataFrames

Panda Series

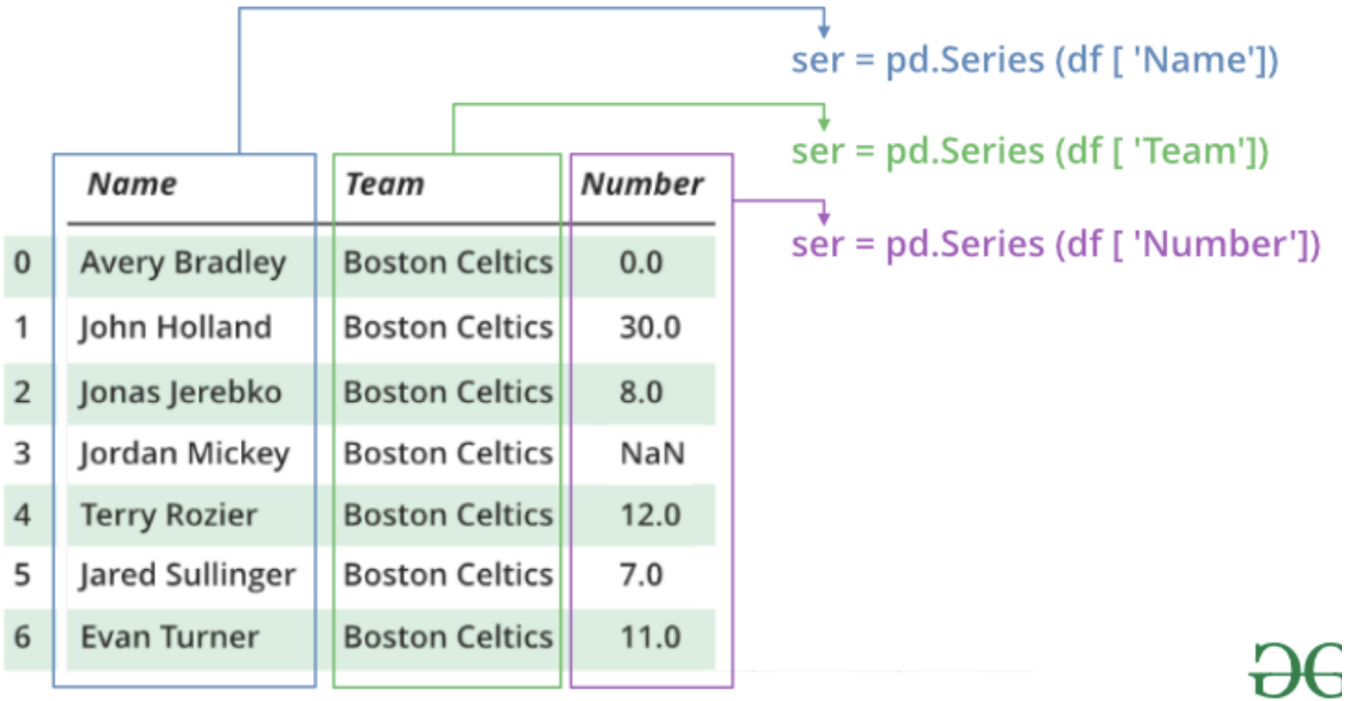
A Panda Series is a one-dimensional ... labeled array capable of holding data of any type (integer, string, float, python objects, etc.).

Panda DataFrame

A Panda DataFrame is a two-dimensional (potentially heterogeneous) tabular data structure with labeled axes for the rows and columns.

Panda Series

Panda Series Elements: columns, data ...



	Name	Team	Number
0	Avery Bradley	Boston Celtics	0.0
1	John Holland	Boston Celtics	30.0
2	Jonas Jerebko	Boston Celtics	8.0
3	Jordan Mickey	Boston Celtics	NaN
4	Terry Rozier	Boston Celtics	12.0
5	Jared Sullinger	Boston Celtics	7.0
6	Evan Turner	Boston Celtics	11.0

Basic Operations:

- Create a series; access elements; index and select data; binary operations; conversion operations.

Panda Series

Example 1: Manually create series from list:

```
# Part 1: Manually create series ...

a = [1, 2, 3, 4, 3, 2, 1 ]
myvar = pd.Series(a)
print(myvar)

# Part 2: Create series from a list with labels ...

myvar = pd.Series(a, index = ["a", "b", "c", "d", "c", "b", "a" ])
print(myvar)
```

Abbreviated Output: Parts 1 and 2 ...

```
Part 01
0    1
1    2
.....
5    2
6    1
dtype: int64
```

```
Part 02
a    1
b    2
.....
b    2
a    1
dtype: int64
```

Panda Series

Example 2: Manually create series from dictionary:

```
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories)
print(myvar)
```

Output:

```
day1    420
day2    380
day3    390
dtype: int64
```

Panda Series

Example 3: Create series from NumPy functions

```
# series01 = pd.Series(np.arange(2,8)) ... ");

series01 = pd.Series(np.arange(2,8))
print(series01)
```

Output:

```
0    2
1    3
2    4
3    5
4    6
5    7
dtype: int64
```

Panda Series

Example 4: Create series from NumPy functions

```
series02 = pd.Series( np.linspace(0,10,5) )
print(series02)

print( series02.size)
print( len(series02) )
print( series02.values )
```

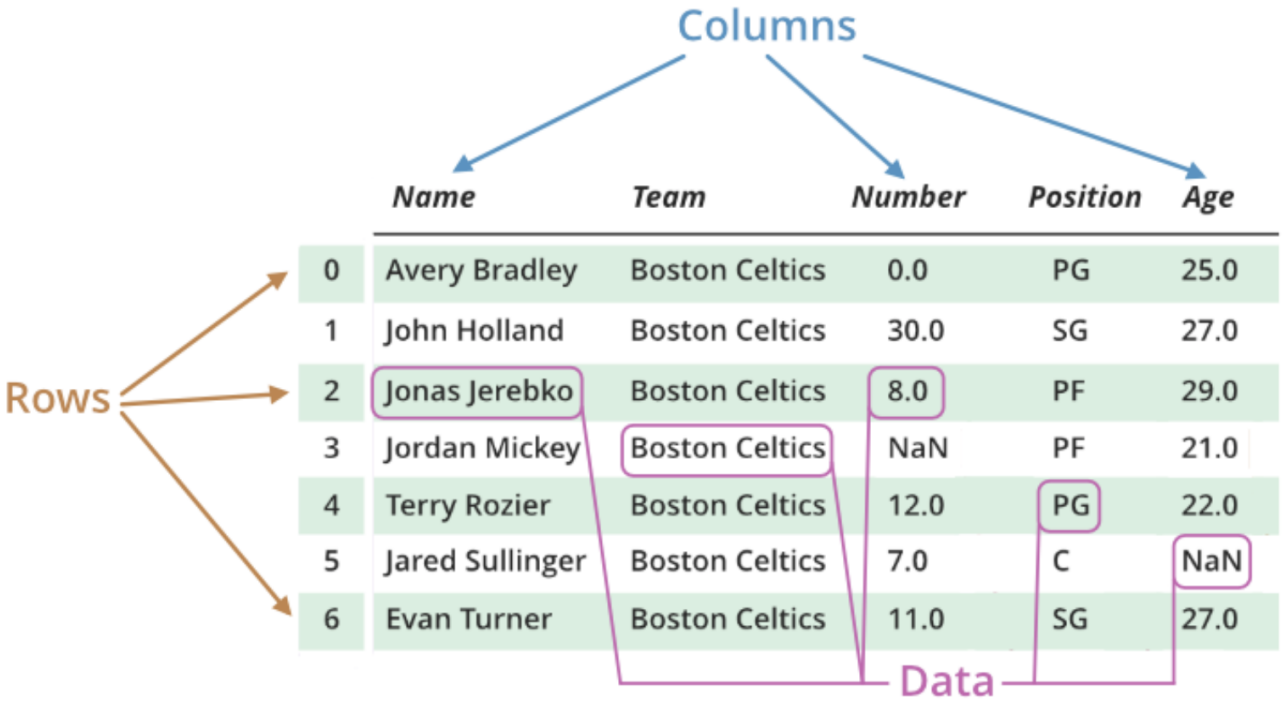
Output:

```
0     0.0
1     2.5
2     5.0
3     7.5
4    10.0
dtype: float64

5                           # <-- series02.size ...
5                           # <-- series02 length ...
[ 0.   2.5  5.   7.5 10. ] # <-- series02 values ...
```

Panda DataFrames

Panda DataFrame Elements: rows, columns, data ...



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Basic Operations:

- Create dataframe; deal with rows and columns; index and select data; iterate over rows and columns.

Working with Panda DataFrames

Example 1: Manually create small dataset ...

```
mydataset = {
   'cars': [ "BMW", "Honda", "Acura"],
   'year': [ 2013,     2017,    2022]
}

myvar = pd.DataFrame(mydataset)
print(myvar)
```

Output:

```
     cars  year
0     BMW  2013
1   Honda  2017
2   Acura  2022
```

Working with Panda DataFrames

Example 2: Create dataframes from 1-d and 2-d arrays ...

```
myvar = pd.DataFrame( np.arange(1,8) ) # <-- dataframe from 1-d array
print(myvar)

df = pd.DataFrame( [ [1,2],
                     [3,4],
                     [5,6] ] )          # <-- dataframe from 2-d array
print(df)
```

Abbreviated Output:

```
Dataframe from 1-d np array
---------------------------
   0
0  1
1  2
2  3
....
5  6
6  7
```

```
Dataframe from 2-d np array
---------------------------
      0  1
   0  1  2
   1  3  4
   2  5  6
```

Working with Panda DataFrames

Example 3: Create simple dataframe from multiple series ...

```
data = {                              # <-- Create dataframe from
   "calories": [520, 480, 400],       #     multiple series.
   "duration": [ 50,  48,  40]
}

myvar = pd.DataFrame(data)
print(myvar)

index = ["day1", "day2", "day3"]  # <-- give each ros a new name.
myvar = pd.DataFrame(data, index)
print(myvar)
```

Output:

```
Part 1: dataframe from series
   calories  duration
0       520        50
1       480        48
2       400        40
```

```
Part 2: rename rows
      calories  duration
day1       520        50
day2       480        48
day3       400        40
```

Working with Panda DataFrames

Example 4: Create dataframe from JSON object ...

```
# Create JSON object (same format as Python dictionary) ...

data = {
  "Duration":{ "0":60,  "1":60, "2":60,   "3":45,  "4":45,  "5":60 },
  "Pulse":{    "0":110, "1":117, "2":103, "3":109, "4":117, "5":102 },
  "Maxpulse":{ "0":130, "1":145, "2":135, "3":175, "4":148, "5":127 },
  "Calories":{ "0":409, "1":479, "2":340, "3":282, "4":406, "5":300 }
}

df = pd.DataFrame(data)
print(df)
```

Output:

```
   Duration  Pulse  Maxpulse  Calories
0        60    110       130       409
1        60    117       145       479
2        60    103       135       340
3        45    109       175       282
4        45    117       148       406
5        60    102       127       300
```

Working with Panda DataFrames

Example 5: Select rows and columns from dataframe ...

```
# Select columns of a dataframe ...

print( df[ [ 'Duration','Calories'] ].head() )

# Selecting rows of a dataframe ...

print( df.loc['1'].head() )     # <-- extract and print row 1
print( df.loc['2'].head() )     # <-- extract and print row 2
```

Output:

```
Columns of dataframe
--------------------

    Duration  Calories
0         60       409
1         60       479
2         60       340
3         45       282
4         45       406
```

```
Row 1
-----------------

Duration     60
Pulse       117
Maxpulse    145
Calories    479
Name: 1, dtype: int64
```

```
Row 2
-----------------

Duration     60
Pulse       103
Maxpulse    135
Calories    340
Name: 2, dtype: int64
```

Working with Pandas

Example 6: Read and plot CSV data file.

```
df = pd.read_csv('../data/AirPassengers.csv')
print(df.head())

print(df.info())  # <-- print dataframe info and shape ...
print(df.shape)
```

Output:

```
     Month  #Passengers
0  1949-01          112
1  1949-02          118
2  1949-03          132
3  1949-04          129
4  1949-05          121
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 2 columns):
#   Column       Non-Null Count  Dtype
---  ------       --------------  -----
0   Month        144 non-null    object
1   #Passengers  144 non-null    int64
dtypes: int64(1), object(1)
memory usage: 2.4+ KB
None
(144, 2)
```

Working with Pandas

Example 6: (continued)

```
import matplotlib.pyplot as plt

ax = plt.gca()
df.plot(kind='line',x='Month',y='#Passengers',color='blue',ax=ax)
plt.show()
```

Output: