Pandas Practice Set for Beginners

August 17, 2025

Contents

1 Introduction

This document provides a beginner-friendly guide to learning Pandas, a powerful Python library for data manipulation and analysis. It includes code and explanations from three Jupyter notebooks: `Pandas-Series-Practice1.ipynb`, `Pandas-DataFrames-1.ipynb`, and `Pandas-DataFrame-Part-2.ip` Each section covers key concepts like creating Series, manipulating DataFrames, and filtering data, with code examples and their outputs.

2 Pandas Series Basics

This section covers the creation and manipulation of Pandas Series, a one-dimensional labeled array, using the `Pandas-Series-Practice1.ipynb` notebook.

2.1 Importing Libraries

```
1 import numpy as np
2 import pandas as pd
```

Description: Imports the NumPy and Pandas libraries, which are essential for numerical operations and data manipulation, respectively.

2.2 Setting Up Data

```
1 labels = ['a','b','c']
2 my_data = [10,20,30]
3 arr = np.array(my_data)
4 d = {'a':10,'b':20,'c':30}
```

Description: Defines a list of labels, a list of data values, a NumPy array from the data, and a dictionary for use in creating Series.

2.3 Creating a Series from a List

```
1 pd.Series(data=my_data)
```

Output:

```
0    10
1    20
2    30
dtype: int64
```

Description: Creates a Pandas Series from the list `my_data` with default integer indices.

2.4 Creating a Series with Custom Indices

```
1 pd.Series(my_data, labels)
```

Output:

```
a    10
b    20
c    30
dtype: int64
```

Description: Creates a Series using `my_data` with custom indices from the `labels` list.

2.5 Alternative Syntax for Series with Indices

```
pd.Series(data=my_data, index=labels)
```

Output:

```
a    10
b    20
c    30
dtype: int64
```

Description: Demonstrates an alternative syntax for creating a Series with custom indices, equivalent to the previous example.

2.6 Creating a Series from a NumPy Array

```
pd.Series(arr, labels)
```

Output:

```
a    10
b    20
c    30
dtype: int64
```

Description: Creates a Series from a NumPy array `arr` with custom indices.

2.7 Creating a Series from a Dictionary

```
pd.Series(d)
```

Output:

```
a    10
b    20
c    30
dtype: int64
```

Description: Creates a Series directly from a dictionary `d`, where keys become indices and values become data.

2.8 Displaying the Dictionary

```
d
```

Output:

```
{'a': 10, 'b': 20, 'c': 30}
```

Description: Displays the dictionary `d` to confirm its contents.

2.9 Series with Functions as Data

```
pd.Series(data=[sum, print, len])
```

Output:

```
0      <built-in function sum>
1    <built-in function print>
2      <built-in function len>
dtype: object
```

Description: Shows that a Series can hold arbitrary objects, such as Python built-in functions.

2.10 Creating a Series with Custom Indices

```
ser1 = pd.Series([1, 2, 3, 4], ['Guna', 'Indore', 'America', 'Shivpuri'])
```

Output (after displaying `ser1`):

```
Guna        1
Indore      2
America     3
Shivpuri    4
dtype: int64
```

Description: Creates a Series with numerical data and location-based indices.

2.11 Creating Another Series

```
ser2 = pd.Series([1, 2, 5, 4], ['Guna', 'Noida', 'Italy', 'Shivpuri'])
```

Output (after displaying `ser2`):

```
Guna        1
Noida       2
Italy       5
Shivpuri    4
dtype: int64
```

Description: Creates another Series with different indices for comparison.

2.12 Accessing a Series Element

```
ser2['Noida']
```

Output:

```
2
```

Description: Retrieves the value associated with the index `'Noida'` from `ser2`.

2.13 Creating a Series from Labels

```
ser3 = pd.Series(data=labels)
```

Output (after accessing `ser3[0]`):

```
'a'
```

Description: Creates a Series from the `labels` list and demonstrates accessing the first element.

2.14 Adding Two Series

```
ser1 + ser2
```

Output:

```
America     NaN
Guna        2.0
Italy       NaN
Noida       NaN
Shivpuri    8.0
dtype: float64
```

Description: Adds two Series, aligning by index. Non-matching indices result in `NaN`.

3 Pandas DataFrames: Part 1

This section, based on `Pandas-DataFrames-1.ipynb`, introduces DataFrame creation and basic operations.

3.1 Setting Up the Environment

```
import numpy as np
import pandas as pd
from numpy.random import randn
np.random.seed(101)
```

Description: Imports libraries and sets a random seed for reproducibility.

3.2 Creating a DataFrame

```
df = pd.DataFrame(randn(5, 4), ['A', 'B', 'C', 'D', 'E'], ['W', 'X', 'Y', '
    Z'])
```

Output:

```
          W         X         Y         Z
A  2.706850  0.628133  0.907969  0.503826
B  0.651118 -0.319318 -0.848077  0.605965
C -2.018168  0.740122  0.528813 -0.589001
D  0.188695 -0.758872 -0.933237  0.955057
E  0.190794  1.978757  2.605967  0.683509
```

Description: Creates a 5x4 DataFrame with random values, labeled rows (A-E) and columns (W-Z).

3.3 Selecting a Column

```
df['W']
```

Output:

```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
```

```
E    0.190794
Name: W, dtype: float64
```

Description: Selects the `W` column, returning a Series.

3.4 Checking Column Type

```
type(df['W'])
```

Output:

```
pandas.core.series.Series
```

Description: Confirms that a single column is a Pandas Series.

3.5 Checking DataFrame Type

```
type(df)
```

Output:

```
pandas.core.frame.DataFrame
```

Description: Verifies that `df` is a DataFrame.

3.6 Selecting Multiple Columns

```
df[['W', 'Z']]
```

Output:

```
          W         Z
A  2.706850  0.503826
B  0.651118  0.605965
C -2.018168 -0.589001
D  0.188695  0.955057
E  0.190794  0.683509
```

Description: Selects columns `W` and `Z`, returning a DataFrame.

3.7 Adding a New Column

```
df['new'] = df['W'] + df['Z']
```

Description: Creates a new column `new` by adding columns `W` and `Z`.

3.8 Displaying the Updated DataFrame

```
df
```

Output:

```
          W         X         Y         Z       new
A  2.706850  0.628133  0.907969  0.503826  3.210676
B  0.651118 -0.319318 -0.848077  0.605965  1.257083
C -2.018168  0.740122  0.528813 -0.589001 -2.607169
D  0.188695 -0.758872 -0.933237  0.955057  1.143752
E  0.190794  1.978757  2.605967  0.683509  0.874303
```

Description: Shows the DataFrame with the new column.

3.9 Dropping a Column

```
df.drop("new", axis=1, inplace=True, errors='ignore')
```

Description: Removes the `new` column permanently from the DataFrame.

3.10 Displaying the DataFrame After Dropping

```
df
```

Output:

```
          W         X         Y         Z
A  2.706850  0.628133  0.907969  0.503826
B  0.651118 -0.319318 -0.848077  0.605965
C -2.018168  0.740122  0.528813 -0.589001
D  0.188695 -0.758872 -0.933237  0.955057
E  0.190794  1.978757  2.605967  0.683509
```

Description: Confirms the column `new` has been removed.

3.11 Checking DataFrame Shape

```
df.shape
```

Output:

```
(5, 4)
```

Description: Returns the dimensions of the DataFrame (5 rows, 4 columns).

3.12 Selecting Columns Y and X

```
df[['Y', 'X']]
```

Output:

```
          Y         X
A  0.907969  0.628133
B -0.848077 -0.319318
C  0.528813  0.740122
D -0.933237 -0.758872
E  2.605967  1.978757
```

Description: Selects columns `Y` and `X` in that order.

3.13 Selecting a Row with loc

```
df.loc['A']
```

Output:

```
W    2.706850
X    0.628133
Y    0.907969
Z    0.503826
Name: A, dtype: float64
```

Description: Selects row `A` using label-based indexing, returning a Series.

3.14 Selecting a Row with iloc

```
df.iloc[0]
```

Output:

```
W    2.706850
X    0.628133
Y    0.907969
Z    0.503826
Name: A, dtype: float64
```

Description: Selects the first row using integer-based indexing.

3.15 Selecting a Specific Cell

```
df.loc["B", "Y"]
```

Output:

```
-0.8480769834036315
```

Description: Retrieves the value at row `B`, column `Y`.

3.16 Selecting a Subset of Rows and Columns

```
df.loc[['A', 'B'], ['X', 'Y']]
```

Output:

```
          X         Y
A  0.628133  0.907969
B -0.319318 -0.848077
```

Description: Selects a subset of the DataFrame with rows `A`, `B` and columns `X`, `Y`.

4 Pandas DataFrames: Part 2

This section, based on `Pandas-DataFrame-Part-2.ipynb`, covers conditional filtering and advanced DataFrame operations.

4.1 Setting Up the Environment

```
import numpy as np
import pandas as pd
from numpy.random import randn
np.random.seed(101)
```

Description: Imports libraries and sets a random seed for reproducibility.

4.2 Creating a DataFrame

```
df = pd.DataFrame(randn(5, 4), ['A', 'B', 'C', 'D', 'E'], ['W', 'X', 'Y', '
    Z'])
```

Output:

```
          W         X         Y         Z
A -0.993263  0.196800 -1.136645  0.000366
B  1.025984 -0.156598 -0.031579  0.649826
C  2.154846 -0.610259 -0.755325 -0.346419
D  0.147027 -0.479448  0.558769  1.024810
E -0.925874  1.862864 -1.133817  0.610478
```

Description: Creates a new 5x4 DataFrame with different random values.

4.3 Boolean DataFrame

```
df > 0
```

Output:

```
       W      X      Y      Z
A  False   True  False   True
B   True  False  False   True
C   True  False  False  False
D   True  False   True   True
E  False   True  False   True
```

Description: Creates a boolean DataFrame where `True` indicates values greater than 0.

4.4 Filtering with Boolean DataFrame

```
bool_df = df > 0
df[bool_df]
```

Output:

```
          W         X         Y         Z
A       NaN  0.196800       NaN  0.000366
B  1.025984       NaN       NaN  0.649826
C  2.154846       NaN       NaN       NaN
D  0.147027       NaN  0.558769  1.024810
E       NaN  1.862864       NaN  0.610478
```

Description: Filters the DataFrame to show only values where the condition `df > 0` is `True`, replacing others with `NaN`.

4.5 Conditional Selection on a Column

```
df['W'] > 0
```

Output:

```
A    False
B     True
C     True
D     True
E    False
Name: W, dtype: bool
```

Description: Checks which values in column `W` are greater than 0.

4.6 Displaying a Column

```
df['W']
```

Output:

```
A   -0.993263
B    1.025984
C    2.154846
D    0.147027
E   -0.925874
Name: W, dtype: float64
```

Description: Displays the `W` column for reference.

4.7 Filtering Rows Based on a Column

```
df[df['W'] > 0]
```

Output:

```
          W         X         Y         Z
B  1.025984 -0.156598 -0.031579  0.649826
C  2.154846 -0.610259 -0.755325 -0.346419
D  0.147027 -0.479448  0.558769  1.024810
```

Description: Filters the DataFrame to include only rows where `W` > 0.

4.8 Filtering Rows Based on Another Column

```
df[df['Z'] < 0]
```

Output:

```
          W         X         Y         Z
C  2.154846 -0.610259 -0.755325 -0.346419
```

Description: Filters rows where `Z` < 0.

4.9 Storing Filtered DataFrame

```
resultdf = df[df['W'] > 0]
```

Description: Stores the filtered DataFrame (where `W` > 0) in `resultdf`.

4.10 Selecting a Column from Filtered DataFrame

```
resultdf['X']
```

Output:

```
B   -0.156598
C   -0.610259
D   -0.479448
Name: X, dtype: float64
```

Description: Selects the `X` column from the filtered DataFrame.