

Python Pandas: Dataframes

Programming for Data Science with Python

Overview

What is NDFrame?

N-dimensional analogue of DataFrame. Store multi-dimensional in a size-mutable, labeled data structure.

What's a DataFrame?

class DataFrame(NDFrame): **Two-dimensional** size-mutable, potentially heterogeneous tabular data structure with labeled axes (**rows and columns**). Arithmetic operations align on both row and column labels. Can be thought of as a **dict-like** container for **Series** objects.

- DataFrame is a subclass (i.e., special case) of NDFrame.
- In Pandas programs generally, DataFrame is used a lot and NDFrame is used rarely.
- In fact, Pandas has Series for 1D, DataFrame for 2D, and for most people that's the end (even though half of Pandas' name is for Panel which Pandas also has, but most people do not use).

FUN FACT: There is/was even a 4D thing in Pandas, (but truly no one uses it (this being the internet, someone will now appear to say they do!)).

For higher dimensions than two or maybe three, some people have shifted their efforts to **xarray**.

That's probably where it's at if your ambitions cannot be contained in 2D.

1.1 Definition

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

	Column Index (df.columns)					
Row of Index	Series of data	Series of data	Series of data	Series of data	Series of data	Series of data

	Column Index (df.columns)					
(df. index)						

1.2 pandas.DataFrame: Attributes

T → Transpose index and columns

at → Fast label-based scalar accessor

axes → Return a list with the row axis labels and column axis labels as the only members.

blocks → Internal property, property synonym for as_blocks()

dtypes → Return the dtypes in this object.

empty → True if NDFrame is entirely empty [no items], meaning any of the axes are of length 0.

ftypes → Return the ftypes (indication of sparse/dense and dtype) in this object.

iat → Fast integer location scalar accessor.

iloc → Purely integer-location based indexing for selection by position.

ix → A primarily label-location based indexer, with integer position fallback.

loc → Purely label-location based indexer for selection by label.

ndim → Number of axes/array dimensions.

shape → Return a tuple representing the dimensionality of the DataFrame.

size → number of elements in the NDFrame

style → Property returning a Styler object containing methods for building a styled HTML representation for the DataFrame.

values → Numpy representation of NDFrame.

2. Create Dataframes

2.1 DataFrame Constructor

`pandas.DataFrame(data, index, columns, dtype, copy)`

Data:

- can be ndarray, series, map, lists, dict, constants, and another DataFrame.

Index:

- For the row labels, the index to be used for the resulting frame is Optional Default np.arange(n), if no index is passed.

Columns:

- For column labels, the Optional Default syntax is - np.arange(n). This is only true if no index is passed.

dtype:

- Data type of each column

Copy:

- This command (or whatever it is) is used for copying of the data.

Default: False

2.2 Create an Empty Dataframe

****Run the following code block:****

In [1]: *# Create an empty dataframe*

```
import pandas as pd

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

Empty DataFrame
Columns: []
Index: []

In [2]: **import** pandas **as** pd

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

Empty DataFrame
Columns: []
Index: []

2.3 Create a Dataframe from lists

****Run the following 2 code blocks:****

In [3]: *# Create a dataframe from a list*

```
import pandas as pd
```

```
# Declare a List
alist = [1,2,3,4,5]

# Create a dataframe from the List
df = pd.DataFrame(alist)

print(df)
```

```
0
0 1
1 2
2 3
3 4
4 5
```

In [4]: `import pandas as pd`

```
prog_Lang = ['Python', 'C#', 'Java', 'C++', 'Javascript']
df = pd.DataFrame(prog_Lang)
print(df)
```

```
0
0 Python
1 C#
2 Java
3 C++
4 Javascript
```

In [5]: `# Create a dataframe from a List of Lists`

```
import pandas as pd

# Declare a List of Lists - each list element has two elements [string, number]
alistOfLists = [['Alex',10],['Bob',12],['Clarke',13]]

# Create a dataframe from this List, naming the columns as 'Name' and 'Age'
df = pd.DataFrame(alistOfLists, columns=['Name', 'Age'])

print(df)
```

```
   Name  Age
0  Alex   10
1   Bob   12
2 Clarke  13
```

In [6]: `import pandas as pd`

```
courses = [['IPAC', 4340],['PSCI', 2606], ['BAAS', 4100]]
df = pd.DataFrame(courses, columns = ['Course Name', 'Course Number'])
print(df)
```

```
   Course Name  Course Number
0        IPAC             4340
1        PSCI             2606
2        BAAS             4100
```

In [17]: `# Create a dataframe from a List of Lists and set the data type`

```
import pandas as pd

# Declare a List of Lists - each list element has two elements [string, number]
aListOfLists = [['Alex',10], ['Bob',12], ['Clarke',13]]
```

```
# Create a dataframe from this list, naming the columns as 'Name' and 'Age'
df = pd.DataFrame(aListOfLists, columns=['Name', 'Age'], dtype=object)

print(df)
```

	Name	Age
0	Alex	10
1	Bob	12
2	Clarke	13

```
In [21]: import pandas as pd

courses = [['IPAC', 4340], ['PSCI', 2606], ['BAAS', 4100]]
df = pd.DataFrame(courses, columns = ['Course Name', 'Course Number'], dtype=object)
print(df)
```

	Course Name	Course Number
0	IPAC	4340
1	PSCI	2606
2	BAAS	4100

2.4 Create dataframes from dictionaries of ndarray/lists

- All the ndarrays must be of same length.
- If index is passed, then the length of the index should equal to the length of the arrays.
- If no index is passed, then by default, index will be range(n), where n is the array length.

****Run the following 3 code blocks:****

```
In [22]: # Create a dataframe from a dictionary without specified indices

import pandas as pd

# Declare a dictionary that has two key-value pairs
# One key is "Name" that has its value = a list of strings
# Another key is "Age" that has its value = a list of integers

aDict = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}

# Create the dataframe from the dictionary
# VIP NOTES: Automatically adding the indices for the rows

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

	Name	Age
0	Tom	28
1	Jack	34
2	Steve	29
3	Ricky	42

```
In [23]: import pandas as pd

avengersDict = {'Name': ['Tony Stark', 'Peter Parker', 'Bruce Banner'], 'Hero Name': [
df = pd.DataFrame(avengersDict)
print(df)
```

	Name	Hero Name
0	Tony Stark	Iron-man
1	Peter Parker	Spider-man
2	Bruce Banner	Hulk

In [24]: *# Create a dataframe from a dictionary with specified indices*

```
import pandas as pd

# Declare a dictionary that has two key-value pairs
# One key is "Name" that has its value = a List of strings
# Another key is "Age" that has its value = a List of integers

aDict = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}

# Create the dataframe from the dictionary
# VIP NOTES: Specifying the indices for the rows

df = pd.DataFrame(aDict, index=['rank1', 'rank2', 'rank3', 'rank4'])
print(df)
```

	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

In [25]: **import** pandas **as** pd

```
acctDict = {'BankType': ['Checking', 'Savings', 'Fun Money'], 'Account Balance': [1248, 2000, 25]}
df = pd.DataFrame(acctDict, index=['High Priority', 'Med Priority', 'Low Priority'])
print(df)
```

	BankType	Account Balance
High Priority	Checking	1248
Med Priority	Savings	2000
Low Priority	Fun Money	25

In [26]: *# Create a dataframe from a dictionary of series*

```
import pandas as pd

# Declare a dictionary of 2 series named 'one' and 'two'

aDictOfSeries = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']),
                  'two': pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

# Create a dataframe from this dictionary
# VIP NOTES: Each column of a dataframe is a series

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

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

In [27]: **import** pandas **as** pd

```

thingSeriesOfDict = {'thing-one': pd.Series([1,1,1], index = ['a1','b1','c1']),
                    'thing-two': pd.Series([2,2,2], index = ['a2','b2','c2'])}

df = pd.DataFrame(thingSeriesOfDict)
print(df)

```

	thing-one	thing-two
a1	1.0	NaN
a2	NaN	2.0
b1	1.0	NaN
b2	NaN	2.0
c1	1.0	NaN
c2	NaN	2.0

2.5 Access Dataframe Columns

****Run the following code block:****

In [28]: *# Access a dataframe columns*

```

import pandas as pd

# Declare a dictionary of 2 series named 'one' and 'two'
aDictOfSeries = {'one': pd.Series([1, 2, 3], index= ['a', 'b', 'c']),
                 'two': pd.Series([1, 2, 3, 4], index= ['a', 'b', 'c', 'd'])}

# Create a dataframe from this dictionary
# VIP NOTES: Each column of a dataframe is a series

df = pd.DataFrame(aDictOfSeries)

# Access the column 'one' and print it out
# HOW TO access a column: Using its label as a column index

print(df['one'])

```

```

a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64

```

In [29]: **import** pandas **as** pd

```

thingSeriesOfDict = {'thing-one': pd.Series([1,1,1], index = ['a1','b1','c1']),
                    'thing-two': pd.Series([2,2,2], index = ['a2','b2','c2'])}

df = pd.DataFrame(thingSeriesOfDict)
print(df['thing-two'])

```

```

a1    NaN
a2    2.0
b1    NaN
b2    2.0
c1    NaN
c2    2.0
Name: thing-two, dtype: float64

```

2.6 Add Columns into a Dataframe

****Run the following 2 code blocks:****

```
In [30]: # Add columns into a dataframe

import pandas as pd

# Declare a dictionary of 2 series named 'one' and 'two'
aDictOfSeries = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']), 'two': pd.Series([10, 20, 30], index=['a', 'b', 'c'])}

# Create a dataframe from this dictionary
# VIP NOTES: Each column of a dataframe is a series

df = pd.DataFrame(aDictOfSeries)

# Adding a new column to an existing DataFrame object with column label by passing new
# Adding a new series into the dataframe as a new column: 'three'
# First, creating a new series
# Then, assign the new series into the new column

df['three'] = pd.Series([10, 20, 30], index=['a', 'b', 'c'])
print(df)
```

	one	two	three
a	1.0	1	10.0
b	2.0	2	20.0
c	3.0	3	30.0
d	NaN	4	NaN

```
In [33]: import pandas as pd

thingSeriesOfDict = {'thing-one': pd.Series([1,1,1], index = ['a1','b1','c1']),
                    'thing-two': pd.Series([2,2,2], index = ['a2','b2','c2'])}

df = pd.DataFrame(thingSeriesOfDict)

df['thing-three'] = pd.Series([3,3,3], index = ['a3','b3','c3'])
print(df)
```

	thing-one	thing-two	thing-three
a1	1.0	NaN	NaN
a2	NaN	2.0	NaN
b1	1.0	NaN	NaN
b2	NaN	2.0	NaN
c1	1.0	NaN	NaN
c2	NaN	2.0	NaN

```
In [36]: # Add columns into a dataframe

import pandas as pd

# Declare a dictionary of 2 series named 'one' and 'two'
aDictOfSeries = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']), 'two': pd.Series([10, 20, 30], index=['a', 'b', 'c'])}

# Create a dataframe from this dictionary
# VIP NOTES: Each column of a dataframe is a series
```



```
df = pd.DataFrame(aDictOfSeries)# Adding a new column using the existing columns in Dc
df['three']=pd.Series([10, 20, 30],index=['a', 'b', 'c'])
df['four'] = df['one'] + df['three']
print('\n')
print(df)
```

	one	two	three	four
a	1.0	1	10.0	11.0
b	2.0	2	20.0	22.0
c	3.0	3	30.0	33.0
d	NaN	4	NaN	NaN

In [37]: `import pandas as pd`

```
thingSeriesOfDict = {'thing-one': pd.Series([1,1,1], index = ['a1','b1','c1']),
                     'thing-two': pd.Series([2,2,2], index = ['a2','b2','c2'])}

df = pd.DataFrame(thingSeriesOfDict)

df['thing-three'] = pd.Series([3,3,3], index = ['a3','b3','c3'])
df['thing-four'] = df['thing-two'] * 2
print(df)
```

	thing-one	thing-two	thing-three	thing-four
a1	1.0	NaN	NaN	NaN
a2	NaN	2.0	NaN	4.0
b1	1.0	NaN	NaN	NaN
b2	NaN	2.0	NaN	4.0
c1	1.0	NaN	NaN	NaN
c2	NaN	2.0	NaN	4.0

2.7 Delete/Pop/Remove a Column from a Dataframe

****Run the following code block:****

```
In [38]: # Delete a column using del function
import pandas as pd
# Declare a dictionary of 2 series named 'one' and 'two'
aDictOfSeries = {'one': pd.Series ([1, 2, 3],
index = ['a', 'b', 'c']), 'two': pd.Series([1, 2, 3, 4],
index = ['a', 'b', 'c', 'd']), 'three': pd.Series([10,20,30],
index = ['a', 'b', 'c'])}
# Create a dataframe from this dictionary
# VIP NOTE: Each column of a dataframe is a series
df = pd.DataFrame(aDictOfSeries)
print(df)
print('\n')
# using del function to delete/remove the first column
del(df['one'])
print(df)
print('\n')
# using pop function to delete another column " 'two'
# Deleting another column using PDP function
df.pop('two')
print(df)
```

	one	two	three
a	1.0	1	10.0
b	2.0	2	20.0
c	3.0	3	30.0
d	NaN	4	NaN

	two	three
a	1	10.0
b	2	20.0
c	3	30.0
d	4	NaN

	three
a	10.0
b	20.0
c	30.0
d	NaN

```
In [41]: import pandas as pd

thingSeriesOfDict = {'thing-one': pd.Series([1,1,1], index = ['a1','b1','c1']),
                     'thing-two': pd.Series([2,2,2], index = ['a2','b2','c2']),
                     'thing-three': pd.Series([3,3,3], index = ['a3','b3','c3'])}

df = pd.DataFrame(thingSeriesOfDict)
print(df)

del(df['thing-one'])
print(df)

df.pop('thing-two')
print(df)
```

	thing-one	thing-two	thing-three
a1	1.0	NaN	NaN
a2	NaN	2.0	NaN
a3	NaN	NaN	3.0
b1	1.0	NaN	NaN
b2	NaN	2.0	NaN
b3	NaN	NaN	3.0
c1	1.0	NaN	NaN
c2	NaN	2.0	NaN
c3	NaN	NaN	3.0

	thing-two	thing-three
a1	NaN	NaN
a2	2.0	NaN
a3	NaN	3.0
b1	NaN	NaN
b2	2.0	NaN
b3	NaN	3.0
c1	NaN	NaN
c2	2.0	NaN
c3	NaN	3.0

	thing-three
a1	NaN
a2	NaN
a3	3.0
b1	NaN
b2	NaN
b3	3.0
c1	NaN
c2	NaN
c3	3.0

2.8 Access Rows of a Dataframe → loc & iloc

****Run the following 3 code blocks:****

```
In [42]: # Access rows of a dataframe using Loc function
# Loc is a row index

import pandas as pd

aDictOfSeries = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']),
                 'two': pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(aDictOfSeries)

# Access the row with index='b' and print the row
print(df.loc['b'])

one    2.0
two    2.0
Name: b, dtype: float64
```

```
In [43]: import pandas as pd

thingSeriesOfDict = {'thing-one': pd.Series([1,1,1], index = ['a1','b1','c1']),
                    'thing-two': pd.Series([2,2,2], index = ['a2','b2','c2']),
                    'thing-three': pd.Series([3,3,3], index = ['a3','b3','c3'])}
```

```
df = pd.DataFrame(thingSeriesOfDict)
print(df.loc['b2'])
```

```
thing-one      NaN
thing-two      2.0
thing-three    NaN
Name: b2, dtype: float64
```

In [44]: *# Access rows of a dataframe using iLoc (integer Location/row index) function*

```
import pandas as pd

aDictOfSeries = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']),
                 'two': pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(aDictOfSeries)

# Access the row with index= '2' and print the row
print(df.iloc[2])
```

```
one    3.0
two    3.0
Name: c, dtype: float64
```

In [47]: **import** pandas **as** pd

```
thingSeriesOfDict = {'thing-one': pd.Series([1,1,1], index = ['a1','b1','c1']),
                    'thing-two': pd.Series([2,2,2], index = ['a2','b2','c2']),
                    'thing-three': pd.Series([3,3,3], index = ['a3','b3','c3'])}

df = pd.DataFrame(thingSeriesOfDict)
print(df.iloc[0])
```

```
thing-one    1.0
thing-two    NaN
thing-three  NaN
Name: a1, dtype: float64
```

In [48]: *# Access a group of rows using the ':' operator*

```
import pandas as pd

aDictOfSeries = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']),
                 'two': pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(aDictOfSeries)

# Access all the rows with indices 2, 3 and print them
print(df[2:4])
```

```
   one  two
c  3.0    3
d  NaN    4
```

In [50]: **import** pandas **as** pd

```
thingSeriesOfDict = {'thing-one': pd.Series([1,1,1], index = ['a1','b1','c1']),
                    'thing-two': pd.Series([2,2,2], index = ['a2','b2','c2']),
                    'thing-three': pd.Series([3,3,3], index = ['a3','b3','c3'])}
```

```
df = pd.DataFrame(thingSeriesOfDict)
print(df[2:-1])
```

	thing-one	thing-two	thing-three
a3	NaN	NaN	3.0
b1	1.0	NaN	NaN
b2	NaN	2.0	NaN
b3	NaN	NaN	3.0
c1	1.0	NaN	NaN
c2	NaN	2.0	NaN

2.9 Delete/Remove Rows from a Dataframe

****Run the following code block:****

In [59]: *# Remove rows from a dataframe using the drop() function*

```
import pandas as pd

df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a', 'b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a', 'b'])

df = pd.concat([df, df2])

# Drop rows with Label 0
df = df.drop(0)

print(df)
```

	a	b
1	3	4
1	7	8

In [60]: **import pandas as pd**

```
df = pd.DataFrame([[11, 12], [13, 14]], columns = ['a', 'b'])
df2 = pd.DataFrame([[15, 16], [17, 18]], columns = ['a', 'b'])

df = pd.concat([df, df2])
df = df.drop(1)
print(df)
```

	a	b
0	11	12
0	15	16

3. Load Data into DataFrame

To read data into a dataframe, we use this command:

pd.read_file_type(file_name)

Where file_type can be csv, excel, etc.

For example, for CSV files, the command to read a csv file: `pd.read_csv()`

Example:

You will have your datasets already loaded but I want you to have an exaple in case you challenge yourself and work on your own.

```
import pandas as pd
```

Reads the flights data set and create the dataframe flights

```
df_flights = pd.read_csv  
( 'C:/DATA/DROPBOX/Dropbox/DATA_APPLS/DATASETS/flights_2 .  
csv' )
```

The command below would print out the 1st 5 rows of the dataframe that was created.

```
df_flights.head(5)
```

