Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

Pandas

Install pandas

Cmd> py -m pip install --user pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things.

Pandas deals with the following three data structures –

- Series
- DataFrame
- Mutability

All Pandas data structures are value mutable (can be changed) and except Series all are size mutable. Series is size immutable.

Note – DataFrame is widely used and one of the most important data structures. Panel is used much less.

Series

Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56, …

10	23	56	17	52	61	73	90	26	72

Key Points

- Homogeneous data
- Size Immutable
- Values of Data Mutable

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

DataFrame

DataFrame is a two-dimensional array with heterogeneous data. For example,

Name	Age	Gender	Rating
Amol	22	Male	4.2
Aarti	21	Female	2.6
Akshay	25	Male	2.9
Pooja	28	Female	4.78

The table represents the data of a sales team of an organization with their overall performance rating. The data is represented in rows and columns. Each column represents an attribute and each row represents a person.

Data Type of Columns

The data types of the four columns are as follows –

Column	Type
Name	String
Age	Integer
Gender	String
Rating	Float

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.

Create a Series from ndarray

If data is an ndarray, then index passed must be of the same length. If no index is passed, then by default index will be **range(n)** where **n** is array length,

i.e., [0,1,2,3…. **range(len(array))-1].**

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print s
```

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data,index=[100,101,102,103])
print s
```

Its **output** is as follows –

```
100  a
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
101  b
102  c
103  d
dtype: object
```

Create a Series from dict

A **dict** can be passed as input and if no index is specified, then the dictionary keys are taken in a sorted order to construct index. If **index** is passed, the values in data corresponding to the labels in the index will be pulled out.

```
#import the pandas library and aliasing as pd

import pandas as pd

import numpy as np

data = {'a' : 0., 'b' : 1., 'c' : 2.}

s = pd.Series(data)

print s
```

Its **output** is as follows –

```
a 0.0
b 1.0
c 2.0
dtype: float64
```

Observe – Dictionary keys are used to construct index.

```
#import the pandas library and aliasing as pd

import pandas as pd

import numpy as np

data = {'a' : 0., 'b' : 1., 'c' : 2.}

s = pd.Series(data,index=['b','c','d','a'])

print s
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

Its **output** is as follows –

```
b 1.0
c 2.0
d NaN
a 0.0
dtype: float64
```

Observe – Index order is persisted and the missing element is filled with NaN (Not a Number).

Create a Series from Scalar

If data is a scalar value, an index must be provided. The value will be repeated to match the length of **index**

```
#import the pandas library and aliasing as pd

import pandas as pd

import numpy as np

s = pd.Series(5, index=[0, 1, 2, 3])

print s
```

Its **output** is as follows –

```
0  5
1  5
2  5
3  5
dtype: int64
```

Accessing Data from Series with Position

Data in the series can be accessed similar to that in an **ndarray.**

Example 1

Retrieve the first element. As we already know, the counting starts from zero for the array, which means the first element is stored at zeroth position and so on.

```
import pandas as pd
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])


#retrieve the first element

print s[0]
```

Its **output** is as follows –

```
1
```

Example 2

Retrieve the first three elements in the Series. If a : is inserted in front of it, all items from that index onwards will be extracted. If two parameters (with : between them) is used, items between the two indexes (not including the stop index)

```
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])


#retrieve the first three element

print s[:3]
```

Its **output** is as follows –

```
a  1
b  2
c  3
dtype: int64
```

Example 3

Retrieve the last three elements.

```
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])


#retrieve the last three element

print s[-3:]
```

Its **output** is as follows –

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
c  3
d  4
e  5
dtype: int64
```

Retrieve Data Using Label (Index)

A Series is like a fixed-size **dict** in that you can get and set values by index label.

Example 1

Retrieve a single element using index label value.

```
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])


#retrieve a single element

print s['a']
```

Its **output** is as follows –

```
1
```

Example 2

Retrieve multiple elements using a list of index label values.

```
import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])


#retrieve multiple elements

print s[['a','c','d']]
```

Its **output** is as follows –

```
a  1
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
c  3
d  4
dtype: int64
```

Python Pandas - DataFrame

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

DataFrame's key Features

- Potentially columns are of different types
- Size – Mutable
- Labeled axes (rows and columns)
- Can Perform Arithmetic operations on rows and columns

Create an Empty DataFrame

A basic DataFrame, which can be created is an Empty Dataframe.

Example

```
#import the pandas library and aliasing as pd
import pandas as pd
df = pd.DataFrame()
print df
```

Its **output** is as follows –

```
Empty DataFrame
Columns: []
Index: []
```

Create a DataFrame from Lists

The DataFrame can be created using a single list or a list of lists.

Example 1

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
import pandas as pd

data = [1,2,3,4,5]

df = pd.DataFrame(data)

print df
```

Its **output** is as follows –

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

```
import pandas as pd

data = [['Ram',40],['Chetan',32],['Mayur',23]]

df = pd.DataFrame(data,columns=['Name','Age'])

print df
```

out put:

```
import pandas as pd

data = [['Amol',20],['Manish',32],['Rahul',33]]

df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)

print df
```

```
import pandas as pd

data = [['Alex',10,'mumbai'],

        ['Bob',12,'pune'],

        ['Clarke',13,'Nashik']]
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
df = pd.DataFrame(data,columns=['Name','Age','city'])

print (df)
```

Create a DataFrame from Dict of ndarrays / 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.

```
import pandas as pd

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

df = pd.DataFrame(data)

print df
```

Its **output** is as follows –

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

Note – Observe the values 0,1,2,3. They are the default index assigned to each using the function range(n).

Example 2

Let us now create an indexed DataFrame using arrays.

```
import pandas as pd

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

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

print df
```

Its **output** is as follows –

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

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

Note – Observe, the **index** parameter assigns an index to each row.

Create a DataFrame from List of Dicts

List of Dictionaries can be passed as input data to create a DataFrame. The dictionary keys are by default taken as column names.

Example 1

The following example shows how to create a DataFrame by passing a list of dictionaries.

```
import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]

df = pd.DataFrame(data)

print df
```

Its **output** is as follows –

```
   a   b     c
0  1   2   NaN
1  5   10  20.0
```

Note – Observe, NaN (Not a Number) is appended in missing areas.

Example 2

The following example shows how to create a DataFrame by passing a list of dictionaries and the row indices.

```
import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]

df = pd.DataFrame(data, index=['first', 'second'])

print df
```

Its **output** is as follows –

```
        a   b     c
first   1   2   NaN
second  5   10  20.0
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

Example 3

The following example shows how to create a DataFrame with a list of dictionaries, row indices, and column indices.

```
import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]


#With two column indices, values same as dictionary keys

df1 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])


#With two column indices with one index with other name

df2 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])

print df1

print df2
```

Its **output** is as follows –

```
#df1 output
        a  b
first   1  2
second  5  10

#df2 output
        a  b1
first   1  NaN
second  5  NaN
```

Note – Observe, df2 DataFrame is created with a column index other than the dictionary key; thus, appended the NaN's in place. Whereas, df1 is created with column indices same as dictionary keys, so NaN's appended.

Column Addition

We will understand this by adding a new column to an existing data frame.

Example

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
import pandas as pd


d = {'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(d)


# Adding a new column to an existing DataFrame object with column label by passing new
series


print ("Adding a new column by passing as Series:")

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

print df


print ("Adding a new column using the existing columns in DataFrame:")

df['four']=df['one']+df['three']


print df
```

Its **output** is as follows –

```
Adding a new column by passing as Series:
    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

Adding a new column using the existing columns in DataFrame:
     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
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

Column Deletion

Columns can be deleted or popped; let us take an example to understand how.

Example

```
# Using the previous DataFrame, we will delete a column
# using del function
import pandas as pd

d = {'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'])}

df = pd.DataFrame(d)
print ("Our dataframe is:")
print df

# using del function
print ("Deleting the first column using DEL function:")
del df['one']
print df

# using pop function
print ("Deleting another column using POP function:")
df.pop('two')
print df
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

Its **output** is as follows –

```
Our dataframe is:
     one   three  two
a     1.0    10.0   1
b     2.0    20.0   2
c     3.0    30.0   3
d     NaN     NaN   4

Deleting the first column using DEL function:
     three    two
a     10.0     1
b     20.0     2
c     30.0     3
d     NaN      4

Deleting another column using POP function:
   three
a  10.0
b  20.0
c  30.0
d  NaN
```

Python Pandas - Operation with Text Data

```
import pandas as pd


s = pd.Series([‘Martin', 'Turner', 'John', 'Alber@t',’Smith'])


print s
```

lower()

```
import pandas as pd



s = pd.Series(['Tom', 'Rick', 'John',’Smith'])


print s.str.lower()
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

upper()

```
import pandas as pd


s = pd.Series(['Tom', ' Rick', 'John', ])


print s.str.upper()
```

len()

```
import pandas as pd
import numpy as np


s = pd.Series(['Akshay', 'Ram','Rahul'])
print s.str.len()
```

strip()

```
import pandas as pd
import numpy as np
s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])
print s
print ("After Stripping:")
print s.str.strip()
```

Its **output** is as follows –

```
0           Tom
1   William Rick
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
2          John
3       Alber@t
dtype: object

After Stripping:
0           Tom
1   William Rick
2          John
3       Alber@t
dtype: object
```

split(pattern)

```
import pandas as pd

import numpy as np

s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])

print s

print ("Split Pattern:")

print s.str.split(' ')
```

Its **output** is as follows –

```
0           Tom
1   William Rick
2          John
3       Alber@t
dtype: object

Split Pattern:
0   [Tom, , , , , , , , , , ]
1   [, , , , , William, Rick]
2   [John]
3   [Alber@t]
dtype: object
```

cat(sep=pattern)

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
import pandas as pd
import numpy as np

s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])

print s.str.cat(sep='_')
```

Its **output** is as follows –

```
Tom _ William Rick_John_Alber@t
```

contains ()

```
import pandas as pd

s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])

print s.str.contains(' ')
```

Its **output** is as follows –

```
0   True
1   True
2   False
3   False
dtype: bool
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

replace(a,b)

```
import pandas as pd

s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])

print s

print ("After replacing @ with $:")

print s.str.replace('@','$')
```

Its **output** is as follows –

```
0   Tom
1   William Rick
2   John
3   Alber@t
dtype: object

After replacing @ with $:
0   Tom
1   William Rick
2   John
3   Alber$t
dtype: object
```

count(pattern)

```
import pandas as pd


s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])


print ("The number of 'm's in each string:")

print s.str.count('m')
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

Its **output** is as follows –

```
The number of 'm's in each string:
0    1
1    1
2    0
3    0
```

startswith(pattern)

```
import pandas as pd

s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])

print ("Strings that start with 'T':")

print s.str. startswith ('T')
```

Its **output** is as follows –

```
0  True
1  False
2  False
3  False
dtype: bool
```

endswith(pattern)

```
import pandas as pd

s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])

print ("Strings that end with 't':")

print s.str.endswith('t')
```

Its **output** is as follows –

```
Strings that end with 't':
0  False
1  False
2  False
3  True
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
dtype: bool
```

find(pattern)

```
import pandas as pd

s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])

print s.str.find('e')
```

Its **output** is as follows –

```
0 -1
1 -1
2 -1
3  3
dtype: int64
```

"-1" indicates that there no such pattern available in the element.

findall(pattern)

```
import pandas as pd

s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
print s.str.findall('e')
```

Its **output** is as follows –

```
0 []
1 []
2 []
3 [e]
dtype: object
```

Null list([ ]) indicates that there is no such pattern available in the element.

swapcase()

```
import pandas as pd


s = pd.Series(['Tom', 'William Rick', 'John', 'Alber@t'])

print s.str.swapcase()
```

Its **output** is as follows –

```
0  tOM
1  wILLIAM rICK
2  jOHN
3  aLBER@T
dtype: object
```

islower()

```
import pandas as pd


s = pd.Series(['Tom', 'William Rick', 'John', 'Alber@t'])

print s.str.islower()
```

Its **output** is as follows –

```
0  False
```

Amit Khojare
Mobile: +91 9975 61 2580
Email : amit@anvistar.in
www.anvistar.in PUNE, INDIA.

```
1  False
2  False
3  False
dtype: bool
```

isupper()

```
import pandas as pd


s = pd.Series(['Tom', 'William Rick', 'John', 'Alber@t'])


print s.str.isupper()
```

Its **output** is as follows –

```
0  False
1  False
2  False
3  False
dtype: bool
```

isnumeric()

```
import pandas as pd


s = pd.Series(['Tom', 'William Rick', 'John', 'Alber@t'])


print s.str.isnumeric()
```

Its **output** is as follows –

```
0  False
1  False
2  False
3  False
dtype: bool
```

**********The End **********