How can I learn about Pandas Series with examples?

Authored by

stats writer

June 24, 2024

Pandas Series is a powerful data structure in Python, designed to handle labeled one-dimensional data. To learn about Pandas Series with practical examples, one can start by accessing online resources such as documentation, tutorials, and video tutorials. These resources provide a comprehensive overview of the functions and methods available in Pandas Series, along with real-world examples to demonstrate their usage. Additionally, one can also practice coding with Pandas Series by working on small projects or exercises to gain hands-on experience. This approach will help in understanding the various features of Pandas Series and how to effectively use them for data manipulation and analysis. With dedication and practice, one can easily master the use of Pandas Series for data analysis tasks.

Pandas Series Introduction

This is a beginner's guide of Python pandas Series Tutorial where you will learn what is pandas Series? its features, advantages, and how to use panda Series with sample examples.

Advertisements

Every sample example explained in this tutorial is tested in our development environment and is available for reference.

All pandas Series examples provided in this tutorial are basic, simple, and easy to practice for beginners who are enthusiastic to learn pandas and advance their career in Data Science, analytics, and Machine Learning.

Note: In case you can't find the pandas Series examples you are looking for on this tutorial page, I would recommend using the Search option from the menu bar to find your tutorial and sample example code, there are hundreds of tutorials in pandas on this website you can learn from.

What is the Pandas Series

Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, etc.). It's similar to a one-dimensional array or a list in Python, but with additional functionalities. Each element in a Pandas Series has a label associated with it, called an index. This index allows for fast and efficient data access and manipulation. Pandas Series can be created from various data structures like lists, dictionaries, NumPy arrays, etc.

Pandas Series vs DataFrame?

Syntax of pandas.series()

Following is the syntax of the pandas.series(), which is used to create Pandas Series objects.

```
# Pandas Series Constructor Syntax
Pandas.series(data,index,dtype,copy)
```

Create pandas Series

pandas Series can be created in multiple ways, From array, list, dict, and from existing DataFrame.

Create Series using array

Before creating a Series, first, we have to import the NumPy module and use `array()` function in the program. If the data is ndarray, then the passed index should be in the same length, if the index is not passed the default value is **range(n)**.

```
# Create Series from array
import pandas as pd
import numpy as np
data = np.array()
series = pd.Series(data)
print (series)
```

Output:
0 python
1 php
2 java
dtype: object

Notice that the column doesn't have a name. And Series also adds an incremental sequence number as Index (first column) by default.

To customize the index of a Pandas Series, you can provide the index parameter when creating the Series using the `pd.Series()` constructor.

```
# Create pandas DataFrame with custom index
s2=pd.Series(data=data, index=)
print(s2)
```

Output:
r1 python
r2 php

r3 java
dtype: object

Create Series using Dict

A **Dict** can be used as input. Keys from Dict are used as Index and values are used as a column.

```
# Create a Dict from a input
data = {'Courses' :"pandas", 'Fees' : 20000, 'Duration' : "30days"}
s2 = pd.Series(data)
print (s2)
```

Output:
Courses pandas
Fees 20000
Duration 30days
dtype: object

Now let's see how to ignore Index from Dict and add the Index while creating a Series with Dict.

```
# To See index from Dict and add index while creating a Series.
data = {'Courses' :"pandas", 'Fees' : 20000, 'Duration' : "30days"}
s2 = pd.Series(data, index=)
print (s2)
```

Create Series using List

Below is an example of creating DataFrame from List.

```
# Creating DataFrame from List
data =
s2 = pd.Series(data, index=)
print(s2)
```

Output:
r1 python
r2 php
r3 java

dtype:object

Create Empty Series

Sometimes you would require to create an empty Series. you can do so by using its empty constructor.

```
# Create empty Series
import pandas as pd
s = pd.Series()
print(s)
```

This shows an empty series.

Convert a Series into a DataFrame

To convert Series into DataFrame, you can use `pandas.concat()`, `pandas.merge()`, `DataFrame.join()`. Below I have explained using `concat()` function. For others, please refer to pandas combine two Series to DataFrame

```
# Convert series to dataframe
courses = pd.Series(, name='courses')
fees = pd.Series(, name='fees')
df=pd.concat(,axis=1)
print(df)
```

Output:
courses fees
0 Spark 22000
1 PySpark 25000
2 Hadoop 23000

Convert pandas DataFrame to Series

In this section of the pandas Series Tutorial, I will explain different ways to convert DataFrame to Series. As I explained in the beginning. Given that each column in a DataFrame is essentially a Series, it follows that we can easily extract single or multiple columns from a DataFrame and convert them into Series objects

You can convert a single-column DataFrame into a Series by extracting that single column.To obtain a Series from a specific column in a multi-column DataFrame, simply access that column using its name.To convert a single row of a DataFrame into a Series, you can utilize indexing to select the row and obtain it as a Series

Convert a single DataFrame column into a series

To run some examples of converting a single DataFrame column into a series, let's create a DataFrame. By using **DataFrame.squeeze()** to convert the DataFrame into a Series:

```
# Create DataFrame with single column
data =
df = pd.DataFrame(data, columns = )
my_series = df.squeeze()
print(my_series)
print (type(my_series))
```

The DataFrame will now get converted into a Series:

```
# Output:
0 Python
1 PHP
2 Java
Name: Courses, dtype: object
<class 'pandas.core.series.Series'>
```

Convert the DataFrame column into a series

You can use the `.squeeze()` method to convert a DataFrame column into a Series.

For example, if we a multiple-column DataFrame

```
# Create DataFrame with multiple columns
import pandas as pd
data = {'Courses': ,
'Duration':,
'Fee':
}
df = pd.DataFrame(data, columns = )
```

```
print(df)
print (type(df))
```

This will convert the `Fee` column of your DataFrame `df` into a Series named `my_series`. If the column contains only one level of data (i.e., it's not a DataFrame itself), `.squeeze()` will return it as a Series.

```
# Pandas DataFrame column to series
my_series= df.squeeze()
```

Convert DataFrame Row into a Series

You can use `.iloc` to access a row by its integer position and then use `.squeeze()` to convert it into a Series if it has only one element.

```
# Convert dataframe row to series
my_series = df.iloc.squeeze()
print(my_series)
print (type(my_series))
```

Then, we can get the following series:

```
# Output:
Courses Python
Duration 50 days
Fee NaN
Name: 2, dtype: object
<class 'pandas.core.series.Series'>
```

Merge DataFrame and Series?

Construct a dataframe from the series.After that merge with the dataframe.Specify the data as the values, multiply them by the length, set the columns to the index and set params for left_index and set the right_index to True.

```
# Syntax for merge with the DataFrame.
```

```
df.merge(pd.DataFrame(data = * len(s), columns = s.index), left_index=True, right_index=True)
```

Pandas Series Attributes:

T	Return the transpose, which is by definition self.
array	The ExtensionArray of the data backing this Series or Index.
at	Access a single value for a row/column label pair.
attrs	Dictionary of global attributes of this dataset.
axes	Return a list of the row axis labels.
dtype	Return the dtype object of the underlying data.
dtypes	Return the dtype object of the underlying data.
flags	Get the properties associated with this pandas object.
hasnas	Return if I have any nans; enables various perf speedups.
iat	Access a single value for a row/column pair by integer position.
iloc	Purely integer-location based indexing for selection by position.
index	The index (axis labels) of the Series.
is_monotonic	Return boolean if values in the object are monotonic_increasing.
is_monotonic_decreasing	Return boolean if values in the object are monotonic_decreasing.
is_monotonic_increasing	Alias for is_monotonic.
is_unique	Return boolean if values in the object are unique.
loc	Access a group of rows and columns by label(s) or a boolean array.
name	Return the name of the Series.
nbytes	Return the number of bytes in the underlying data.
ndim	Number of dimensions of the underlying data, by definition 1.
shape	Return a tuple of the shape of the underlying data.
size	Return the number of elements in the underlying data.
values	Return Series as ndarray or ndarray-like depending on the dtype.

Pandas Series Methods:

abs()	Return a Series/DataFrame with absolute numeric value of each element.
add(other)	Return Addition of series and other, element-wise (binary operator add).
add_prefix(prefix)	Prefix labels with string prefix.
add_suffix(suffix)	Suffix labels with string suffix.
agg()	Aggregate using one or more operations over the specified axis.
aggregate()	Aggregate using one or more operations over the specified axis.
align(other)	Align two objects on their axes with the specified join method.
all()	Return whether all elements are True, potentially over an axis.
any()	Return whether any element is True, potentially over an axis.
append(to_append)	Concatenate two or more Series.
apply(func)	Invoke function on values of Series.
argmax()	Return int position of the largest value in the Series.
argmin()	Return int position of the smallest value in the Series.
argsort()	Return the integer indices that would sort the Series values.
asfreq(freq)	Convert time series to specified frequency.
asof(where)	Return the last row(s) without any NaNs before where.
astype(dtype)	Cast a pandas object to a specified dtype .
at_time(time)	Select values at particular time of day (e.g., 9:30AM).
autocorr()	Compute the lag-N autocorrelation.
backfill()	Synonym for DataFrame.fillna() with method="bfill".
between(left, right)	Return boolean Series equivalent to left <= series <= right.
between_(start_time, end_time)	Select values between particular times of the day (e.g., 9:00-9:30 AM).

Continue..

bfill()	Synonym for DataFrame.fillna() with method="bfill" .
bool()	Return the bool of a single element Series or DataFrame.
cat	alias of pandas.core.arrays.categorical.categoricalAccessor
clip()	Trim values at input threshold(s).

combine(other, func)	Combine the Series with a Series or scalar according to func.
combine_first(other)	Update null elements with value in the same location in 'other'.
compare(other)	Compare to another Series and show the differences.
convert()	Convert columns to best possible dtypes using dtypes supporting pd.NA.
copy()	Make a copy of this object's indices and data.
corr(other)	Compute correlation with other Series, excluding missing values.
count()	Return number of non-NA/null observations in the Series.
cov(other)	Compute covariance with Series, excluding missing values.
cummax()	Return cumulative maximum over a DataFrame or Series axis.
cummin()	Return cumulative minimum over a DataFrame or Series axis.
cumprod()	Return cumulative product over a DataFrame or Series axis.
cumsum()	Return cumulative sum over a DataFrame or Series axis.
describe()	Generate descriptive statistics.
diff()	First discrete difference of element.
div(other)	Return Floating division of series and other, element-wise (binary operator truediv).
divide(other)	Return Floating division of series and other, element-wise (binary operator truediv).
divmod(other)	Return Integer division and modulo of series and other, element-wise (binary operator divmod).
dot(other)	Compute the dot product between the Series and the columns of other.
drop()	Return Series with specified index labels removed.
drop_duplicate()	Return Series with duplicate values removed.
droplevel(level)	Return Series/DataFrame with requested index / column level(s) removed.
dropna()	Return a new Series with missing values removed.
dt	alias of pandas.core.indexes.accessors.CombinedDatetimelikeproperties.
duplicated()	Indicate duplicate Series values.
eq(other)	Return Equal to of series and other, element-wise (binary operator eq).
equals(other)	Test whether two objects contain the same elements.
ewm()	Provide exponential weighted (EW) functions.
expanding()	Provide expanding transformations.

explode()	Transform each element of a list-like to a row.
factorize()	Encode the object as an enumerated type or categorical variable.
ffill()	Synonym for DataFrame.fillna()with method=ffill().
fillna()	Fill NA/NaN values using the specified method.
filter()	Subset the dataframe rows or columns according to the specified index labels.
first(offset)	Select initial periods of time series data based on a date offset.
first_valid_other()	Return index for first non-NA value or None, if no NA value is found.

Conclusion

In this pandas Series tutorial, we have learned about what is panda series? how to create a Panda Series with different types of inputs, convert Pandas Series to DataFrame, and vice versa with working examples.

References

Related Articles