How can I use cbind in Python to combine columns from multiple arrays, similar to the cbind function in R?

Authored by

stats writer

July 2, 2024

The cbind function in R allows users to combine columns from multiple arrays into a single array. Similarly, in Python, the cbind function can be used to achieve the same result. This function takes in multiple arrays as arguments and combines their columns to create a new array. It is a useful tool for data manipulation and analysis, allowing users to easily merge data from different sources. By utilizing the cbind function in Python, users can efficiently organize and analyze their data in a concise and effective manner.

Use cbind in Python (Equivalent to R)

The cbind function in R, short for *column-bind*, can be used to combine data frames together by their columns.

We can use the function from pandas to perform the equivalent function in Python:

df3 = pd.concat(, axis=1)

The following examples shows how to use this function in practice.

Example 1: Use cbind in Python with Equal Index Values

Suppose we have the following two pandas DataFrames:

import pandas as pd

#define DataFrames
df1 = pd.DataFrame({'team': ,

'points': })

print(df1)

team points
0 A 99
1 B 91
2 C 104
3 D 88
4 E 108

df2 = pd.DataFrame({'assists': ,
'rebounds': })

print(df2)

assists rebounds
0 A 22
1 B 19
2 C 25
3 D 33
4 E 29

We can use the concat() function to quickly bind these two DataFrames together by their columns:

```
#column-bind two DataFrames into new DataFrame
df3 = pd.concat(, axis=1)

#view resulting DataFrame
df3

team points assists rebounds
0 A 99 A 22
1 B 91 B 19
2 C 104 C 25
3 D 88 D 33
4 E 108 E 29
```

Example 2: Use cbind in Python with Unequal Index Values

Suppose we have the following two pandas DataFrames:

```
import pandas as pd

#define DataFrames
df1 = pd.DataFrame({'team': ,
'points': })

print(df1)
```

team points
0 A 99
1 B 91
2 C 104
3 D 88
4 E 108

df2 = pd.DataFrame({'assists': ,
'rebounds': })

df2.index =

print(df2)

assists rebounds
6 A 22
7 B 19
8 C 25
9 D 33
10 E 29

Notice that the two DataFrames do not have the same index values.

If we attempt to use the concat() function to cbind them

together, we'll get the following result:

```
#attempt to column-bind two DataFrames
df3 = pd.concat(, axis=1)

#view resulting DataFrame
df3

team points assists rebounds
0 A 99.0 NaN NaN
1 B 91.0 NaN NaN
2 C 104.0 NaN NaN
3 D 88.0 NaN NaN
4 E 108.0 NaN NaN
6 NaN NaN A 22.0
7 NaN NaN B 19.0
8 NaN NaN C 25.0
9 NaN NaN D 33.0
10 NaN NaN E 29.0
```

This is not the result we wanted.

To fix this, we need to first reset the index of each DataFrame before concatenating them together:

```
import pandas as pd

#define DataFrames
df1 = pd.DataFrame({'team': ,
'points': })

df2 = pd.DataFrame({'assists': ,
'rebounds': })

df2.index =

#reset index of each DataFrame
df1.reset_index(drop=True, inplace=True)
df2.reset_index(drop=True, inplace=True)

#column-bind two DataFrames
df3 = pd.concat(, axis=1)

#view resulting DataFrame
df3

team points assists rebounds
0 A 99 A 22
1 B 91 B 19
2 C 104 C 25
3 D 88 D 33
```

4 E 108 E 29

Notice that this DataFrame matches the one we got in the previous example.

Additional Resources

The following tutorials explain how to perform other common operations in Python: