How (and why) should I make a copy of a Pandas DataFrame?

Authored by

stats writer

June 27, 2024

A Pandas DataFrame is a widely used data structure in Python that allows for efficient manipulation and analysis of data. Making a copy of a DataFrame can be beneficial in various scenarios, such as preserving the original data, avoiding unintended changes, and performing multiple operations on the same data. Copying a DataFrame can be easily done using the .copy() method, which creates a deep copy of the data. This ensures that any changes made to the copied DataFrame do not affect the original one. It is important to make a copy of a DataFrame when working with large datasets or performing complex operations to avoid any potential errors. Therefore, it is recommended to make a copy of a Pandas DataFrame to ensure the integrity and accuracy of the data being analyzed.

How (And Why) to Make Copy of Pandas DataFrame

Whenever you create a subset of a pandas DataFrame and then modify the subset, the original DataFrame will also be modified.

For this reason, it's always a good idea to use .copy() when subsetting so that any modifications you make to the subset won't also be made to the original DataFrame.

The following examples demonstrate how (and why) to make a copy of a pandas DataFrame when subsetting.

Example 1: Subsetting a DataFrame Without Copying

Suppose we have the following pandas DataFrame:

import pandas as pd

```
#create DataFrame
df = pd.DataFrame({'team': ,
'points': ,
'assists': })

#view DataFrame
print(df)

team points assists
0 A 18 5
1 B 22 7
2 C 19 7
3 D 14 9
4 E 14 12
5 F 11 9
6 G 20 9
7 H 28 4
```

Now suppose we create a subset that contains only the first four rows of the original DataFrame:

```
#define subsetted DataFrame
df_subset = df

#view subsetted DataFrame
```

```
print(df_subset)

team points assists rebounds
0 A 18 5 11
1 B 22 7 8
2 C 19 7 10
3 D 14 9 6
```

If we modify one of the values in the subset, the value in the original DataFrame will also be modified:

```
#change first value in team column
df_subset.team = 'X'

#view subsetted DataFrame
print(df_subset)

team points assists
0 X 18 5
1 B 22 7
2 C 19 7
3 D 14 9

#view original DataFrame
print(df)
```

team points assists
0 X 18 5
1 B 22 7
2 C 19 7
3 D 14 9
4 E 14 12
5 F 11 9
6 G 20 9
7 H 28 4

Notice that the first value in the team column has been changed from 'A' to 'X' in both the subsetted DataFrame and the original DataFrame.

This is because we didn't make a copy of the original DataFrame.

Example 2: Subsetting a DataFrame With Copying

Once again suppose we have the following pandas DataFrame:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'team': ,
```

'points': ,
'assists': })

#view DataFrame
print(df)

team points assists
0 A 18 5
1 B 22 7
2 C 19 7
3 D 14 9
4 E 14 12
5 F 11 9
6 G 20 9
7 H 28 4

Once again suppose we create a subset that contains only the first four rows of the original DataFrame, but this time we use .copy() to make a copy of the original DataFrame:

#define subsetted DataFrame
df_subset = df.copy()

#change first value in team column

df_subset.team = 'X'

#view subsetted DataFrame

print(df_subset)

team points assists

0 X 18 5

1 B 22 7

2 C 19 7

3 D 14 9

#view original DataFrame

print(df)

team points assists

0 A 18 5

1 B 22 7

2 C 19 7

3 D 14 9

4 E 14 12

5 F 11 9

6 G 20 9

7 H 28 4

Notice that the first value in the team column has been

changed from 'A' to 'X' only in the subsetted DataFrame.

The original DataFrame remains untouched since we used .copy() to make a copy of it when creating the subset.

Additional Resources

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