How can the assign() method in Pandas be used?

Authored by

stats writer

June 27, 2024

The assign() method in Pandas can be used to create new columns in a data frame or modify existing ones. This method takes in a dictionary as input, where the keys represent the column names and the values represent the values to be assigned to those columns. It allows for easy manipulation and transformation of data without modifying the original data frame, making it a useful tool for data analysis and preprocessing. Additionally, the assign() method can be chained with other methods, providing a convenient way to perform multiple operations on a data frame in a single line of code.

Use the assign() Method in Pandas (With Examples)

The assign() method can be used to add new columns to a pandas DataFrame.

This method uses the following basic syntax:

df.assign(new_column = values)

It's important to note that this method will only output the new DataFrame to the console, but it won't actually modify the original DataFrame.

To modify the original DataFrame, you would need to store the results of the assign() method in a new variable.

The following examples show how to use the assign() method in different ways with the following pandas DataFrame:

```
import pandas as pd

#define DataFrame
df = pd.DataFrame({'points': ,
'assists': ,
'rebounds': })

#view DataFrameprint(df)

points assists rebounds
0 25 5 11
1 12 7 8
2 15 7 10
3 14 9 6
4 19 12 6
5 23 9 5
6 25 9 9
7 29 4 12
```


Example 1: Assign New Variable to DataFrame

The following code shows how to use the assign() method to add a new variable to the DataFrame called points2 whose values are equal to the values in the points column multiplied by two:

#add new variable called points2

df.assign(points2 = df.points * 2)pointsassists reboundspoints2

0 25 5 11 50

1 12 7 8 24

2 15 7 10 30

3 14 9 6 28

4 19 12 6 38

5 23 9 5 46

6 25 9 9 50

7 29 4 12 58

Note that this assign() method doesn't change the original DataFrame.

If we print the original DataFrame, we'll see that it remains unchanged:

#print original DataFrameprint(df)

points assists rebounds

0 25 5 11

1 12 7 8

2 15 7 10

3 14 9 6

4 19 12 6

5 23 9 5

6 25 9 9

7 29 4 12

To save the results of the assign() method, we can store the results in a new DataFrame:

#add new variable called points2 and save results in new DataFrame
df.assign(points2 = df.points * 2)

#view new DataFrameprint(df_new)

points assists rebounds points2
0 25 5 11 50
1 12 7 8 24
2 15 7 10 30
3 14 9 6 28
4 19 12 6 38
5 23 9 5 46
6 25 9 9 50
7 29 4 12 58

The new DataFrame called df_new now contains the

points2 column that we created.

Example 2: Assign Multiple New Variables to DataFrame

```
#add three new variables to DataFrame and store results in new DataFrame
df_new = df.assign(points2 = df.points * 2,
assists_rebs = df.assists + df.rebounds,
conference = 'Western')

#view new DataFrame
print(df_new)
```

```
points assists rebounds points2 assists_rebs conference
0 25 5 11 50 16 Western
1 12 7 8 24 15 Western
2 15 7 10 30 17 Western
3 14 9 6 28 15 Western
4 19 12 6 38 18 Western
5 23 9 5 46 14 Western
6 25 9 9 50 18 Western
7 29 4 12 58 16 Western
```

Notice that three new columns have been added to the

DataFrame.

Note: You can find the complete documentation for the pandas assign() method .

The following tutorials explain how to use other common functions in pandas: