What is the difference between using “at” and “loc” in Pandas?

Authored by

stats writer

June 28, 2024

Pandas is a popular data analysis library in Python, often used for working with large datasets. Two commonly used methods in Pandas are "at" and "loc" for accessing specific data elements within a DataFrame. The main difference between the two is their functionality. "At" is used for accessing a single value at a specific row and column index, while "loc" is used for accessing multiple values at once, using labels or boolean arrays to specify the rows and columns. Therefore, "at" is more efficient for accessing a single data point, while "loc" is more useful for selecting multiple data points simultaneously. Understanding the difference between these two methods is crucial for effectively manipulating and analyzing data in Pandas.

Pandas at vs. loc: What's the Difference?

When it comes to selecting rows and columns of a pandas DataFrame, .loc and .at are two commonly used functions.

Here is the subtle difference between the two functions:

.loc can take multiple rows and columns as input arguments.at can only take one row and one column as input arguments

The following examples show how to use each function in practice with the following pandas DataFrame:

import pandas as pd

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

'assists': ,
'rebounds': })

#view DataFrame
print(df)

team points assists rebounds
0 A 18 5 11
1 B 22 7 8
2 C 19 7 10
3 D 14 9 6
4 E 14 12 6
5 F 11 9 5
6 G 20 9 9
7 H 28 4 12

Example 1: How to Use loc in Pandas

The following code shows how to use .loc to access the value in the DataFrame located at index position 0 of the points column:

#select value located at index position 0 of the points column
df.loc

18

This returns a value of 18.

And the following code shows how to use .loc to access rows between index values 0 and 4 along with the columns points and assists:

#select rows between index values 0 and 4 and columns 'points' and 'assists'
df.loc]

points assists
0 18 5
1 22 7
2 19 7
3 14 9
4 14 12

Whether we'd like to access one single value or a group of rows and columns, the .loc function can do both.

Example 2: How to Use at in Pandas

The following code shows how to use .at to access the

value in the DataFrame located at index position 0 of the points column:

#select value located at index position 0 of the points column
df.at

18

This returns a value of 18.

However, suppose we try to use at to access rows between index values 0 and 4 along with the columns points and assists:

#try to select rows between index values 0 and 4 and columns 'points' and 'assists'
df.at] TypeError: unhashable type: 'list'

We receive an error because the at function is unable to take multiple rows or multiple columns as input arguments.

Conclusion

When you'd like to access just one value in a pandas

DataFrame, both the loc and at functions will work fine.

However, when you'd like to access a group of rows and columns, only the loc function is able to do so.

Related:

Additional Resources

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

How to Select Rows Based on Column Values in Pandas