How can I save a Pandas DataFrame for later use, and can you provide an example?

Authored by

stats writer

June 28, 2024

Saving a Pandas DataFrame allows for the data to be stored and accessed at a later time, without the need to recreate the DataFrame. This can be achieved by using the "to_csv" function, which converts the DataFrame into a comma-separated values (CSV) file. An example of this would be:

df.to_csv('filename.csv')

This will create a CSV file named "filename" which can be accessed and used in the future by using the "read_csv" function.

Save Pandas DataFrame for Later Use (With Example)

Often you may want to save a pandas DataFrame for later use without the hassle of importing the data again from a CSV file.

The easiest way to do this is by using to save the DataFrame as a pickle file:

df.to_pickle("my_data.pkl")

This will save the DataFrame in your current working environment.

You can then use to quickly read the DataFrame from the pickle file:

df = pd.read_pickle("my_data.pkl")

The following example shows how to use these

functions in practice.

Example: Save and Load Pandas DataFrame

Suppose we create the following pandas DataFrame that contains information about various basketball teams:

```
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

We can use df.info() to view the data type of each variable in the DataFrame:

```
#view DataFrame info
print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 4 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 team 8 non-null object
1 points 8 non-null int64
2 assists 8 non-null int64
3 rebounds 8 non-null int64
dtypes: int64(3), object(1)
memory usage: 292.0+ bytes
None
```

We can use the to_pickle() function to save this DataFrame to a pickle file with a .pkl extension:

#save DataFrame to pickle file
df.to_pickle("my_data.pkl")

Our DataFrame is now saved as a pickle file in our current working environment.

We can then use the read_pickle() function to quickly read the DataFrame:

#read DataFrame from pickle file
df= pd.read_pickle("my_data.pkl")

#view DataFrameprint(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

#view DataFrame info

```
print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 4 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 team 8 non-null object
1 points 8 non-null int64
2 assists 8 non-null int64
3 rebounds 8 non-null int64
dtypes: int64(3), object(1)
memory usage: 292.0+ bytes
None
```

The benefit of using pickle files is that the data type of each column is retained when we save and load the DataFrame.

This provides an advantage over saving and loading CSV files because we don't have to perform any transformations on the DataFrame since the pickle file preserves the original state of the DataFrame.

Additional Resources

The following tutorials explain how to fix other common errors in Python: