How can you create a Pandas DataFrame with examples?

Authored by

stats writer

June 24, 2024

A Pandas DataFrame is a two-dimensional data structure that allows for efficient handling and manipulation of data. It is created using the Pandas library in Python and can be created in various ways. One way is by using a dictionary, where the keys represent the column names and the values represent the data in each column. For example, a DataFrame with student information can be created using a dictionary with keys as 'Name', 'Age', and 'Grade', and values as lists containing the corresponding data. Another way is by importing data from external sources such as CSV files, SQL databases, or Excel spreadsheets. Once the data is imported, it can be converted into a DataFrame using the appropriate Pandas function. Additionally, a DataFrame can also be created by merging multiple smaller DataFrames together. These methods provide flexibility and ease in creating a DataFrame to analyze and manipulate data efficiently.

One simplest way to create a pandas DataFrame is by using its constructor. Besides this, there are many other ways to create a DataFrame in pandas. For example, creating DataFrame from a list, created by reading a CSV file, creating it from a Series, creating empty DataFrame, and many more.

Python pandas is widely used for data science/data analysis and machine learning applications. It is built on top of another popular package named Numpy, which provides scientific computing in Python. pandas DataFrame is a 2-dimensional labeled data structure with rows and columns (columns of potentially different types like integers, strings, float, None, Python objects e.t.c). You can think of it as an excel spreadsheet or SQL table.

1. Create pandas DataFrame

One of the easiest ways to create a pandas DataFrame is by using its constructor. DataFrame constructor takes several optional params that are used to specify the characteristics of the DataFrame.

Below is the **syntax of the DataFrame constructor**.

```
# DataFrame constructor syntax
pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)
```

Now, let's create a DataFrame from a list of lists (with a few rows and columns).

```
# Create pandas DataFrame from List
import pandas as pd
```

```
technologies = ,
,
]
df=pd.DataFrame(technologies)
print(df)
```

Since we have not given index and column labels, DataFrame by default assigns incremental sequence numbers as labels to both rows and columns.

```
# Output:
0 1 2
0 Spark 20000 30days
1 Pandas 25000 40days
```

Column names with sequence numbers don't make sense as it's hard to identify what data holds on each column hence, it is always best practice to provide column names that identify the data it holds. Use `column` param and `index` param to provide column & custom index respectively to the DataFrame.

```
# Add Column & Row Labels to the DataFrame
column_names=
row_label=
df=pd.DataFrame(technologies,columns=column_names,index=row_label)
print(df)
```

Yields below output. Alternatively, you can also add columns labels to the existing DataFrame.

```
# Output:
Courses Fee Duration
a Spark 20000 30days
b Pandas 25000 40days
```

By default, pandas identify the data types from the data and assign's to the DataFrame. `df.dtypes` returns the data type of each column.

```
# Output:
Courses object
```

```
Fee int64
Duration object
dtype: object
```

You can also assign custom data types to columns.

```
# Set custom types to DataFrame
types={'Courses': str,'Fee':float,'Duration':str}
df=df.astype(types)
```

2. Create DataFrame from the Dic (dictionary).

Another most used way to create pandas DataFrame is from the python Dict (dictionary) object. This comes in handy if you wanted to convert the dictionary object into DataFrame. Key from the Dict object becomes column and value convert into rows.

```
# Create DataFrame from Dict
technologies = {
'Courses':,
'Fee' :,
'Duration':
}
df = pd.DataFrame(technologies)
print(df)
```

3. Create DataFrame with Index

By default, DataFrame add's a numeric index starting from zero. It can be changed with a custom index while creating a DataFrame.

```
# Create DataFrame with Index.
technologies = {
'Courses':,
'Fee' :,
'Duration':
}
index_label=
df = pd.DataFrame(technologies, index=index_label)
```

```
print(df)
```

4. Creating Dataframe from list of dicts object

Sometimes we get data in JSON string (similar dict), you can convert it to DataFrame as shown below.

```
# Creates DataFrame from list of dict
technologies =
```

df = pd.DataFrame(technologies)
print(df)

5. Creating DataFrame From Series

By using concat() method you can create Dataframe from multiple Series. This takes several params, for the scenario we use `list` that takes series to combine and `axis=1` to specify merge series as columns instead of rows.

```
# Create pandas Series
courses = pd.Series()
fees = pd.Series()
duration = pd.Series()
```

# Create DataFrame from series objects.
df=pd.concat(,axis=1)
print(df)

Outputs
0 1 2
0 Spark 20000 30days
1 Pandas 25000 40days

6. Add Column Labels

As you see above, by default `concat()` method doesn't add column labels. You can do so as below.

```
# Assign Index to Series
index_labels=
courses.index = index_labels
fees.index = index_labels
duration.index = index_labels
```

```
# Concat Series by Changing Names
df=pd.concat({'Courses': courses,
'Course_Fee': fees,
'Course_Duration': duration},axis=1)
print(df)

# Outputs:
# Courses Course_Fee Course_Duration
# r1 Spark 20000 30days
# r2 Pandas 25000 40days
```

7. Creating DataFrame using zip() function

Multiple lists can be merged using `zip()` method and the output is used to create a DataFrame.

```
# Create Lists
Courses =
Fee =
Duration =

# Merge lists by using zip().
tuples_list = list(zip(Courses, Fee, Duration))
df = pd.DataFrame(tuples_list, columns = )
```

8. Create an empty DataFrame in pandas

Sometimes you would need to create an empty pandas DataFrame with or without columns. This would be required in many cases, below is one example.

While working with files, sometimes we may not receive a file for processing, however, we still need to create a DataFrame manually with the same column names we expect. If we don't create with the same columns, our operations/transformations (like union's) on DataFrame fail as we refer to the columns that may not be present.

Check if pandas DataFrame is empty

To handle situations similar to these, we always need to create a DataFrame with the expected columns, which means the same column names and datatypes regardless of the file exists or empty file processing.

```
# Create Empty DataFrame
df = pd.DataFrame()
print(df)
```

Outputs:
Empty DataFrame
Columns:
Index:

To create an empty DataFrame with just column names but no data.

```
# Create Empty DataFraem with Column Labels
df = pd.DataFrame(columns = )
print(df)
```

Outputs:
Empty DataFrame
Columns:
Index:

9. Create DataFrame From CSV File

In real-time we are often required to read the contents of CSV files and create a DataFrame. In pandas, creating a DataFrame from CSV is done by using `pandas.read_csv()` method. This returns a DataFrame with the contents of a CSV file.

```
# Create DataFrame from CSV file
df = pd.read_csv('data_file.csv')
```

10. Create From Another DataFrame

Finally, you can also copy a DataFrame from another DataFrame using `copy()` method.

```
# Copy DataFrame to another
df2=df.copy()
print(df2)
```

Conclusion

In this article, you have learned different ways to create a pandas DataFrame with examples. It can be created from a constructor, list, dictionary, series, CSV file, and many more.

Happy Learning !!

Related Articles

References