pandas

pandas.DataFrame.to_dict

DataFrame.to_dict(*orient='dict'*, ***, *into=<class 'dict'>*, *index=True*) [source]

Convert the DataFrame to a dictionary.

The type of the key-value pairs can be customized with the parameters (see below).

Parameters:

orient : *str {'dict', 'list', 'series', 'split', 'tight', 'records', 'index'}*

Determines the type of the values of the dictionary.

- 'dict' (default) : dict like {column -> {index -> value}}
- 'list' : dict like {column -> [values]}
- 'series' : dict like {column -> Series(values)}
- 'split' : dict like {'index' -> [index], 'columns' -> [columns], 'data' -> [values]}
- 'tight' : dict like {'index' -> [index], 'columns' -> [columns], 'data' -> [values], 'index_names' -> [index.names], 'column_names' -> [column.names]}
- 'records' : list like [{column -> value}, … , {column -> value}]
- 'index' : dict like {index -> {column -> value}}

New in version 1.4.0: 'tight' as an allowed value for the `orient` argument

into : *class, default dict*

The collections.abc.MutableMapping subclass used for all Mappings in the return value. Can be the actual class or an empty instance of the mapping type you want. If you want a collections.defaultdict, you must pass it initialized.

index : *bool, default True*

Whether to include the index item (and index_names item if *orient* is 'tight') in the returned dictionary. Can only be `False` when *orient* is 'split' or 'tight'.

New in version 2.0.0.

Returns:

dict, list or collections.abc.MutableMapping

Return a collections.abc.MutableMapping object representing the DataFrame. The resulting transformation depends on the *orient* parameter.

See also

`DataFrame.from_dict`

Create a DataFrame from a dictionary.

`DataFrame.to_json`

Convert a DataFrame to JSON format.

Examples

```
>>> df = pd.DataFrame({'col1': [1, 2],
...                    'col2': [0.5, 0.75]},
...                   index=['row1', 'row2'])
>>> df
      col1  col2
row1     1  0.50
row2     2  0.75
>>> df.to_dict()
{'col1': {'row1': 1, 'row2': 2}, 'col2': {'row1': 0.5, 'row2': 0.75}}
```

You can specify the return orientation.

```
>>> df.to_dict('series')
{'col1': row1    1
         row2    2
Name: col1, dtype: int64,
'col2': row1    0.50
        row2    0.75
Name: col2, dtype: float64}
```

```
>>> df.to_dict('split')
{'index': ['row1', 'row2'], 'columns': ['col1', 'col2'],
 'data': [[1, 0.5], [2, 0.75]]}
```

```
>>> df.to_dict('records')
[{'col1': 1, 'col2': 0.5}, {'col1': 2, 'col2': 0.75}]
```

```
>>> df.to_dict('index')
{'row1': {'col1': 1, 'col2': 0.5}, 'row2': {'col1': 2, 'col2': 0.75}}
```

```
>>> df.to_dict('tight')
{'index': ['row1', 'row2'], 'columns': ['col1', 'col2'],
 'data': [[1, 0.5], [2, 0.75]], 'index_names': [None], 'column_names': [No
```

You can also specify the mapping type.

```
>>> from collections import OrderedDict, defaultdict
>>> df.to_dict(into=OrderedDict)
OrderedDict([('col1', OrderedDict([('row1', 1), ('row2', 2)])),
             ('col2', OrderedDict([('row1', 0.5), ('row2', 0.75)]))])
```

If you want a *defaultdict*, you need to initialize it:

```
>>> dd = defaultdict(list)
>>> df.to_dict('records', into=dd)
[defaultdict(<class 'list'>, {'col1': 1, 'col2': 0.5}),
 defaultdict(<class 'list'>, {'col1': 2, 'col2': 0.75})]
```

Previous
pandas.DataFrame.to_hdf

Next
pandas.DataFrame.to_gbq