



# Introduction to Pandas dataframes

## Contents

- [5.1. Import the pandas library](#)
- [5.2. What is a dataframe](#)
- [5.3. Attributes](#)

Modelflow is built on top of the Pandas library. Pandas is the Swiss knife of data science and can perform an impressive array of tasks.

This tutorial is a very short introduction to how pandas dataframes are used with Modelflow.

For more info on pandas:

[Pandas homepage](#)

[Pandas community tutorials](#)

## 5.1. Import the pandas library

The convention is, that pandas is imported as pd

```
import pandas as pd
```

## 5.2. What is a dataframe

A dataframe is a two-dimensional data structure with named rows and columns

Creating a dataframe can be done in many ways. Here we are creating a Dataframe from a dictionary and assigning a list of years as the index.

```
df = pd.DataFrame({'B': [1,1,1,1], 'C': [1,2,3,6], 'E': [4,4,4,4]}, index=[2018,2019,2020,2021])
df
```

|      | B | C | E |
|------|---|---|---|
| 2018 | 1 | 1 | 4 |
| 2019 | 1 | 2 | 4 |
| 2020 | 1 | 3 | 4 |
| 2021 | 1 | 6 | 4 |

In modelflow each column is a time serie for an economic variable. So in this dataframe A, B and C are economic time series.

In this manual all variables will be timeseries scalars (numbers).

However for more advanced use a variable can also be a timeserie of matrices or vectors.

## 5.3. Attributes

Lets take a look at some of the most important attributes of a dataframe.

To learn more check out the [official pandas website](#).

### 5.3.1. .columns, column names

.columns gives you the names of the columns in the dataframe.

```
df.columns
```

```
Index(['B', 'C', 'E'], dtype='object')
```

#### Python object properties

A python object such as `df` has properties and methods. They are accessed by the `.` operator

The dataframe `df` has 3 columns.

### 5.3.2. Column names in Modelflow

The columns have names.

In A dataframe column names can be many different python objects. So for instance 123.4 is a legal column name. That would make construction of models very impossible.

To facilitate construction of models modelflow a column name has to be a string and start with a letter or `_` and followed by letters, underscore or digits.

**All letters in column names should be upper case.**

#### Modelflow variable names

Many python Objects are legal as column names in a dataframe, but not as variable names (column names) in modelflow

- `IB` is legal
- `ib` is illegal
- `42ANSWER` is illegal
- `_HORSE1` is legal
- `VERY_LONG_VARIABLE_NAME_WHICH_IS_LEGAL` is legal

### 5.3.3. .index and time dimensions in Modelflow

The row names are called `.index`. For yearly models a list of integers like in `df` is fine.

Pandas has a number of data types for different date and time objects. They can all be used for models with other frequencies.

However be aware that plots will not necessary be working well with all index names.

When modelflow uses a lagged variable like `A(-1)` It will take the value from the row above the current row. No matter if the index is an integer, a year, quarter or a millisecond. The same goes for leads `A(+1)` That will be the value in the next row.

```
df.index
```

```
Int64Index([2018, 2019, 2020, 2021], dtype='int64')
```

### 5.3.4. .eval() evaluation expressions

With this method expressions can be evaluated and new columns created.

```
df.eval('''X = B*C  
          THE_ANSWER = 42''')
```

|      | B | C | E | X | THE_ANSWER |
|------|---|---|---|---|------------|
| 2018 | 1 | 1 | 4 | 1 | 42         |
| 2019 | 1 | 2 | 4 | 2 | 42         |
| 2020 | 1 | 3 | 4 | 3 | 42         |
| 2021 | 1 | 6 | 4 | 6 | 42         |

However, .eval cannot handle lagged variables and no systems of equations. Modelflow can!

### 5.3.5. .loc[] when slicing a dataframe

#### 5.3.5.1. .loc[row,column] A single element

```
df.loc[2019, 'C']
```

```
2
```

#### 5.3.5.2. .loc[:,column] A single column

```
df.loc[:, 'C']
```

```
2018    1  
2019    2  
2020    3  
2021    6  
Name: C, dtype: int64
```

#### 5.3.5.3. .loc[row,:] A single row

```
df.loc[2019, :]
```

```
B      1  
C      2  
E      4  
Name: 2019, dtype: int64
```

#### 5.3.5.4. .loc[:,[names...]] Several columns

```
df.loc[:, ['B', 'C']]
```


|             | <b>B</b> | <b>C</b> |
|-------------|----------|----------|
| <b>2018</b> | 1        | 1        |
| <b>2019</b> | 1        | 2        |
| <b>2020</b> | 1        | 3        |
| <b>2021</b> | 1        | 6        |

5.3.5.5. .loc[] Slices of the dataframe can also be assigned values

This can be very handy when updating scenarios.

```
df.loc[2019, 'C'] = 42
df
```

|             | <b>B</b> | <b>C</b> | <b>E</b> |
|-------------|----------|----------|----------|
| <b>2018</b> | 1        | 1        | 4        |
| <b>2019</b> | 1        | 42       | 4        |
| <b>2020</b> | 1        | 3        | 4        |
| <b>2021</b> | 1        | 6        | 4        |

 **Warning**

The dimensions on the right hand side of = and the left hand side should match. That is: either the dimensions should be the same, or the right hand side should be broadcasted into the left hand slice. A link [here](#)