Let's set dayfirst to True. Observe that the first input in the string is treated as a day in the output.

```
# set dayfirst to True
pd.to_datetime('5-11-2018', dayfirst = True)
'Output':
Timestamp('2018-11-05 00:00:00')
```

The shift() Method

A typical step in a timeseries use case is to convert the timeseries dataset into a supervised learning framework for predicting the outcome for a given time instant. The **shift()** method is used to adjust a Pandas DataFrame column by shifting the observations forward or backward. If the observations are pulled backward (or lagged), **NaNs** are attached at the tail of the column. But if the values are pushed forward, the head of the column will contain **NaNs**. This step is important for adjusting the **target** variable of a dataset to predict outcomes *n*-days or steps or instances into the future. Let's see some examples.

Subset columns for the observations related to Bitcoin Cash.

```
# subset a few columns
data_subset_BCH = data.loc[data.symbol == 'BCH',
['open','high','low','close']]
data_subset_BCH.head()
'Output':
              open    high     low   close
date
2017-07-23  555.89  578.97  411.78  413.06
2017-07-24  412.58  578.89  409.21  440.70
2017-07-25  441.35  541.66  338.09  406.90
2017-07-26  407.08  486.16  321.79  365.82
2017-07-27  417.10  460.97  367.78  385.48
```

Now let's create a target variable that contains the closing rates 3 days into the future.

```
data_subset_BCH['close_4_ahead'] = data_subset_BCH['close'].shift(-4)
data_subset_BCH.head()
```

```
'Output':
              open    high     low   close  close_4_ahead
date
2017-07-23  555.89  578.97  411.78  413.06         385.48
2017-07-24  412.58  578.89  409.21  440.70         406.05
2017-07-25  441.35  541.66  338.09  406.90         384.77
2017-07-26  407.08  486.16  321.79  365.82         345.66
2017-07-27  417.10  460.97  367.78  385.48         294.46
```

Observe that the tail of the column **close_4_head** contains **NaNs**.

```
data_subset_BCH.tail()
'Output':
              open     high      low    close  close_4_ahead
date
2018-01-06  2583.71  2829.69  2481.36  2786.65        2895.38
2018-01-07  2784.68  3071.16  2730.31  2786.88            NaN
2018-01-08  2786.60  2810.32  2275.07  2421.47            NaN
2018-01-09  2412.36  2502.87  2346.68  2391.56            NaN
2018-01-10  2390.02  2961.20  2332.48  2895.38            NaN
```

Rolling Windows

Pandas provides a function called **rolling()** to find the rolling or moving statistics of values in a column over a specified window. The window is the "number of observations used in calculating the statistic." So we can find the rolling sums or rolling means of a variable. These statistics are vital when working with timeseries datasets. Let's see some examples.

Let's find the rolling means for the closing variable over a 30-day window.

```
# find the rolling means for Bitcoin cash
rolling_means = data_subset_BCH['close'].rolling(window=30).mean()
```

The first few values of the **rolling_means** variable contain **NaNs** because the method computes the rolling statistic from the earliest time to the latest time in the dataset. Let's print out the first five values using the **head** method.