In [ ]:

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```python
airline = pd.read_csv('airline_passengers.csv',index_col="Month")
```

In [4]:
```python
airline.dropna(inplace=True)
airline.index = pd.to_datetime(airline.index)
```

In [5]:
```python
airline.head()
```

Out[5]:

Month	Thousands of Passengers
1949-01-01	112.0
1949-02-01	118.0
1949-03-01	132.0
1949-04-01	129.0
1949-05-01	121.0

SMA

Simple Moving Average

We've already shown how to create a simple moving average, for a quick review:

In [6]:
```python
airline['6-month-SMA']=airline['Thousands of Passengers'].rolling(window=6).mean()
airline['12-month-SMA']=airline['Thousands of Passengers'].rolling(window=12).mean()
```

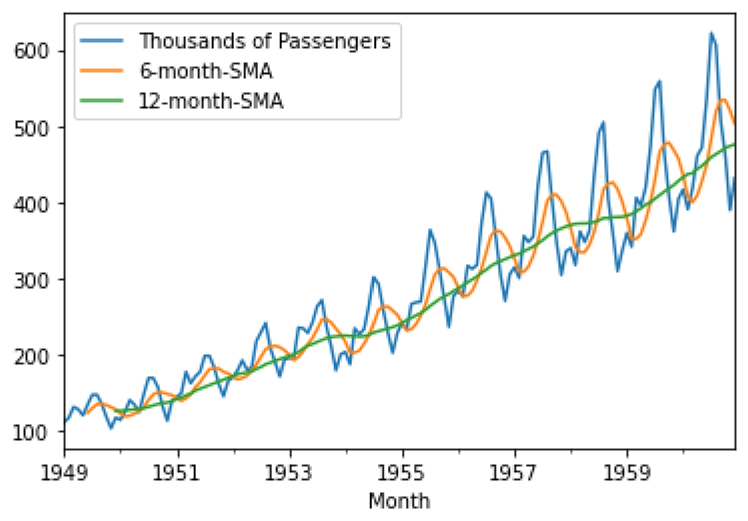
In [7]:
```python
airline.head()
```

Out[7]:

Month	Thousands of Passengers	6-month-SMA	12-month-SMA
1949-01-01	112.0	NaN	NaN
1949-02-01	118.0	NaN	NaN
1949-03-01	132.0	NaN	NaN
1949-04-01	129.0	NaN	NaN
1949-05-01	121.0	NaN	NaN

In [8]:
```python
airline.plot()
```

Out[8]:
```
<AxesSubplot:xlabel='Month'>
```



EWMA

Exponentially-weighted moving average

We just showed how to calculate the SMA based on some window.However, basic SMA has some "weaknesses".

- Smaller windows will lead to more noise, rather than signal
- It will always lag by the size of the window
- It will never reach to full peak or valley of the data due to the averaging.
- Does not really inform you about possible future behaviour, all it really does is describe trends in your data.
- Extreme historical values can skew your SMA significantly

To help fix some of these issues, we can use an EWMA (Exponentially-weighted moving average).

EWMA will allow us to reduce the lag effect from SMA and it will put more weight on values that occured more recently (by applying more weight to the more recent values, thus the name). The amount of weight applied to the most recent values will depend on the actual parameters used in the EWMA and the number of periods given a window size. Full details on Mathematics behind this can be found here Here is the shorter version of the explanation behind EWMA.

The formula for EWMA is:

$$y_t = \frac{\sum_{i=0}^{t} w_i x_{t-i}}{\sum_{i=0}^{t} w_i}$$

Where x_t is the input value, w_i is the applied weight (Note how it can change from i=0 to t), and y_t is the output.

Now the question is, how to we define the weight term w_i ?

This depends on the adjust parameter you provide to the .ewm() method.

When adjust is True (default), weighted averages are calculated using weights:

$y_t = \frac{xt + *(1 - \alpha)*x{t-1} + (1 - \alpha)^2 x_{t-2} + ...

- (1 - \alpha)^t x_{0}}{1 + (1 - \alpha) + (1 - \alpha)^2 + ...
- (1 - \alpha)^t}$

When adjust=False is specified, moving averages are calculated as:

$\begin{split}y_0 &= x_0 \

yt &= (1 - \alpha) y{t-1} + \alpha x_t,\end{split}$

which is equivalent to using weights:

$$w_i = \begin{cases} \alpha(1-\alpha)^i & \text{if } i < t \\ (1-\alpha)^i & \text{if } i = t. \end{cases}$$

When adjust=True we have y0=x0 and from the last representation above we have yt=αxt+(1−α)yt−1, therefore there is an assumption that x0 is not an ordinary value but rather an exponentially weighted moment of the infinite series up to that point.

One must have 0<α≤1, and while since version 0.18.0 it has been possible to pass α directly, it's often easier to think about either the span, center of mass (com) or half-life of an EW moment:

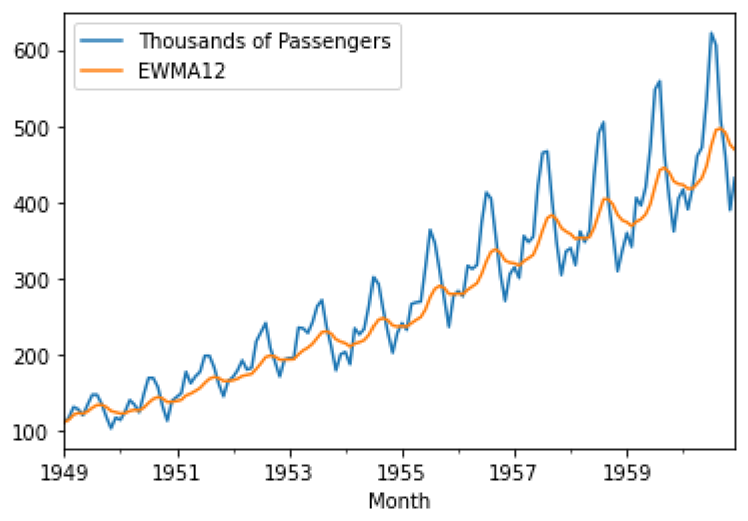
$$\alpha = \begin{cases} \frac{2}{s+1}, & \text{for span } s \geq 1 \\ \frac{1}{1+c}, & \text{for center of mass } c \geq 0 \\ 1 - \exp^{\frac{\log 0.5}{h}}, & \text{for half-life } h > 0 \end{cases}$$

- Span corresponds to what is commonly called an “N-day EW moving average”.
- Center of mass has a more physical interpretation and can be thought of in terms of span: c=(s−1)/2
- Half-life is the period of time for the exponential weight to reduce to one half.
- Alpha specifies the smoothing factor directly.

In [9]:
```python
airline['EWMA12'] = airline['Thousands of Passengers'].ewm(span=12).mean()
```

In [10]:
```python
airline[['Thousands of Passengers','EWMA12']].plot()
```

Out[10]:
```
<AxesSubplot:xlabel='Month'>
```



Great! That is all for now, let's move on to ARIMA modeling!