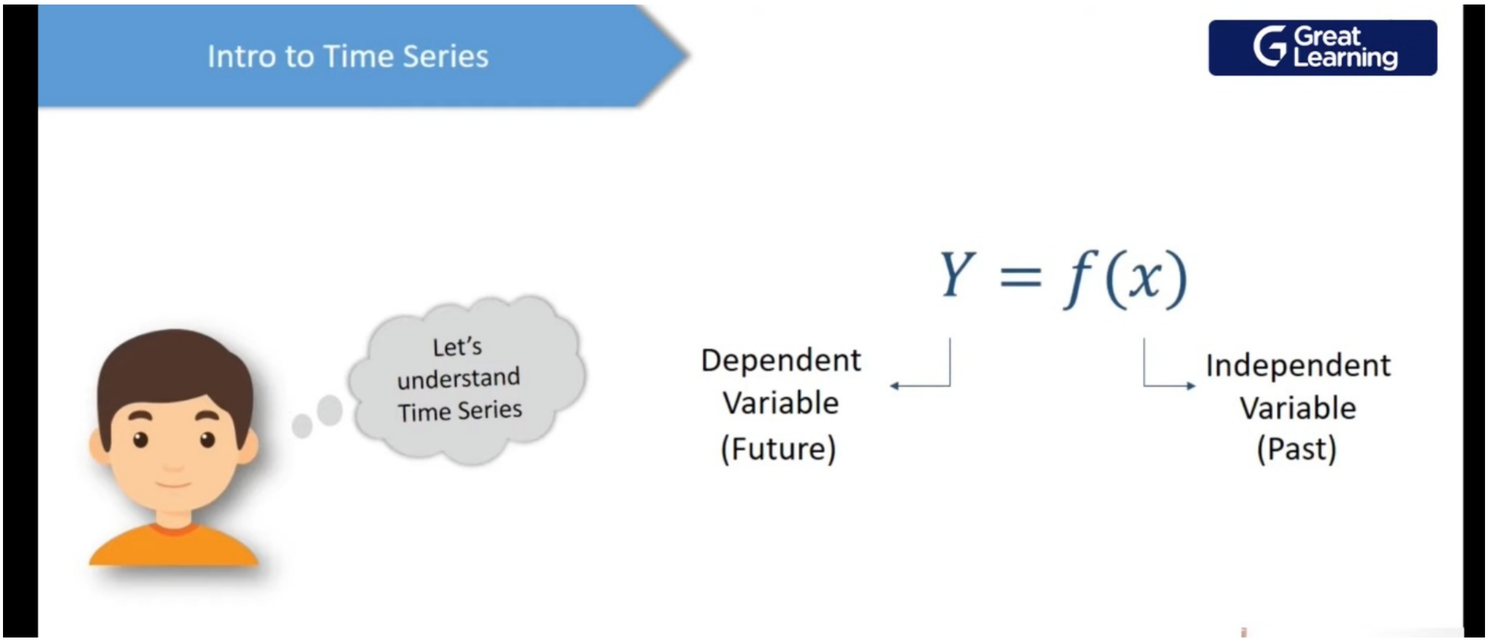


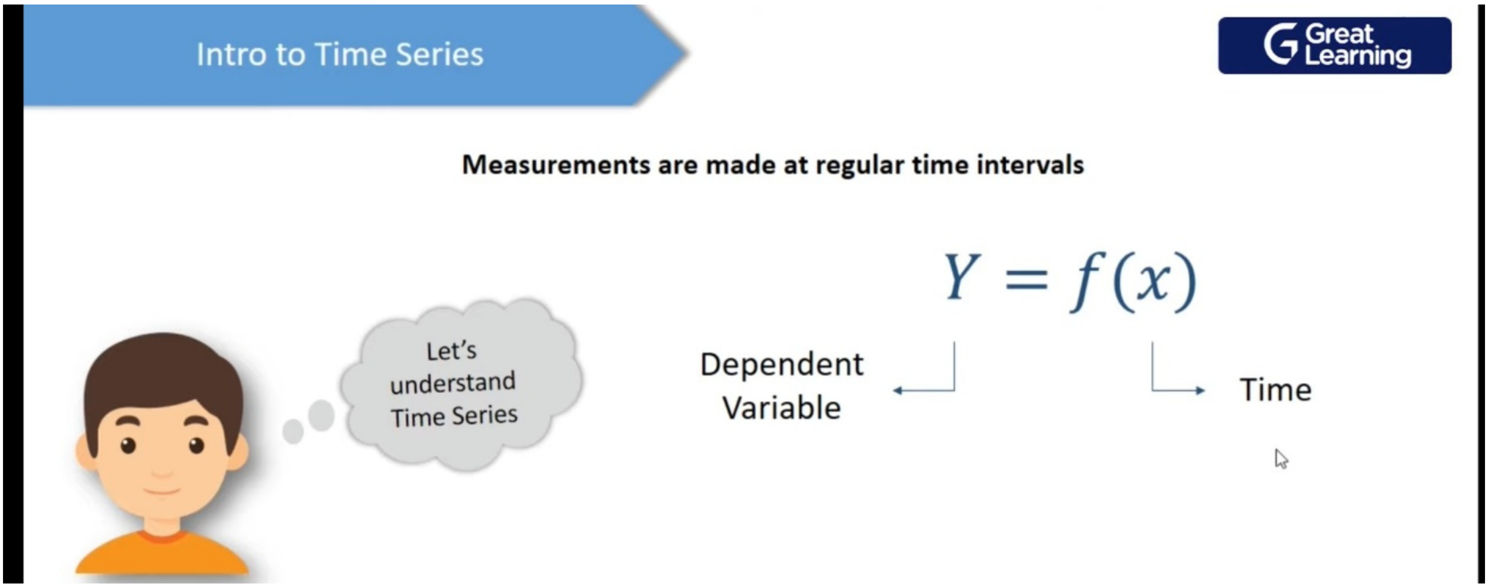
Intro to Time Series

Let's understand Time Series

$$Y = f(x)$$

Dependent Variable (Future)

Independent Variable (Past)



Intro to Time Series

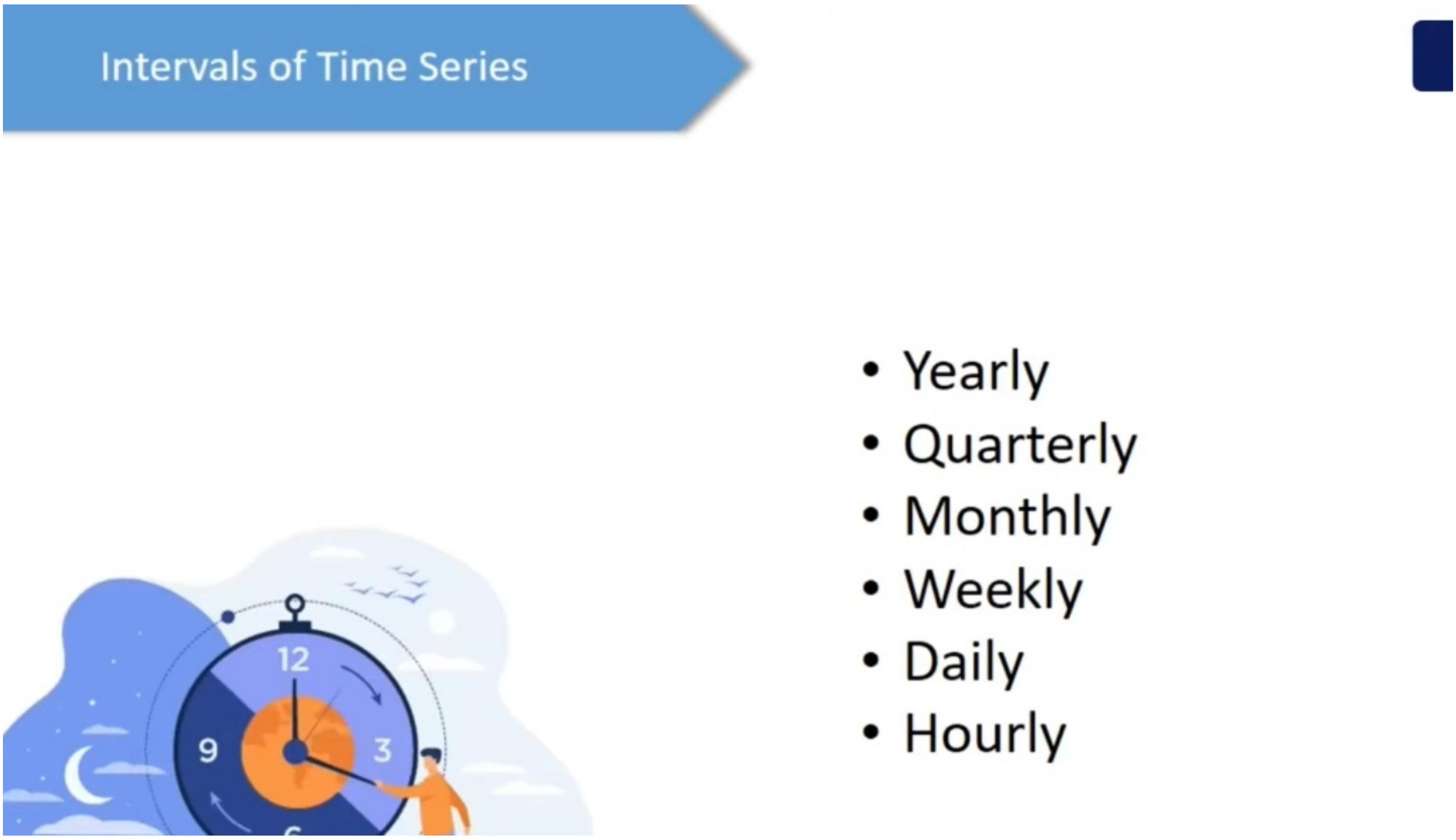
Great Learning

Measurements are made at regular time intervals

$$Y = f(x)$$

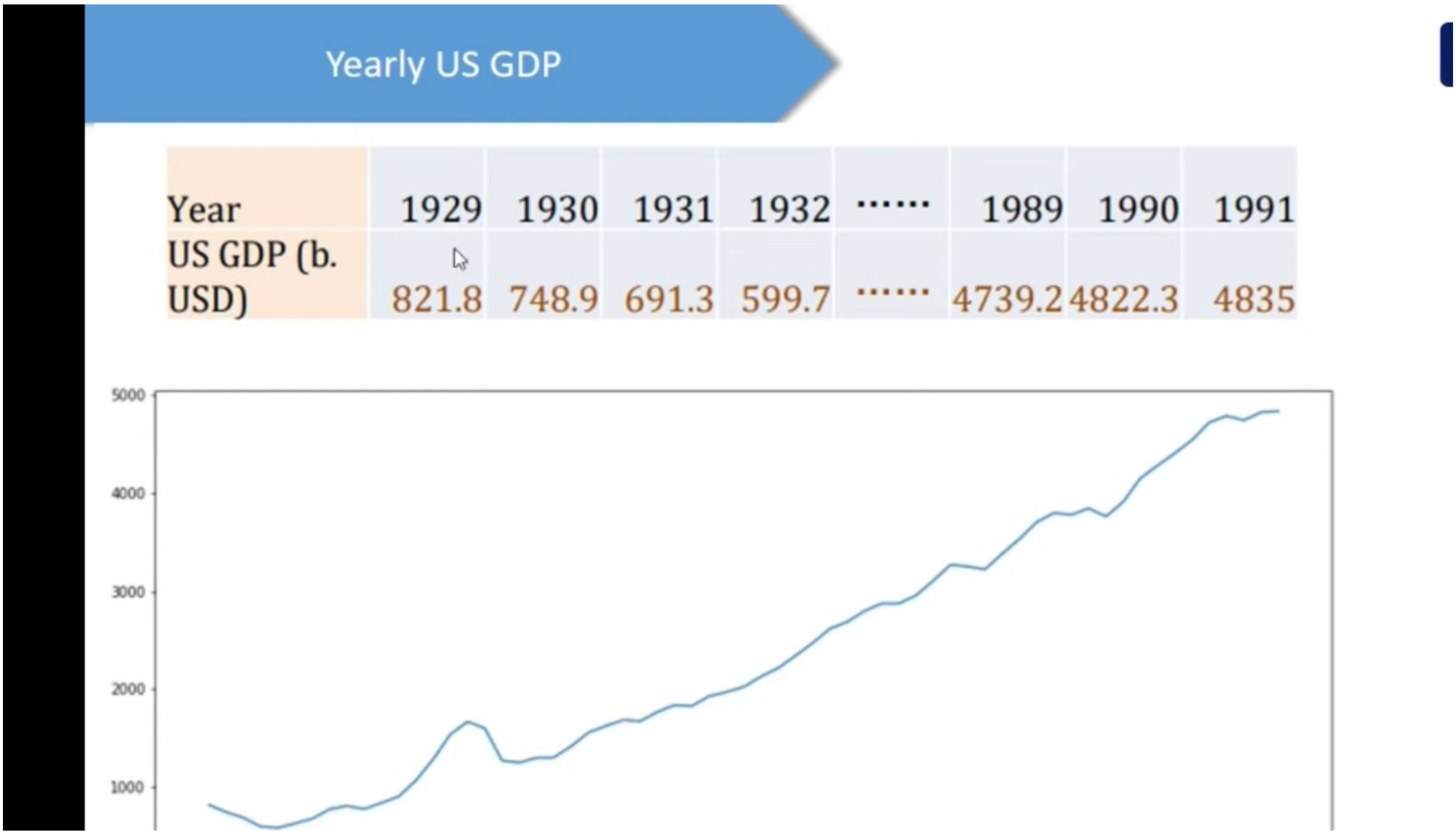
Dependent Variable

Time



Intervals of Time Series

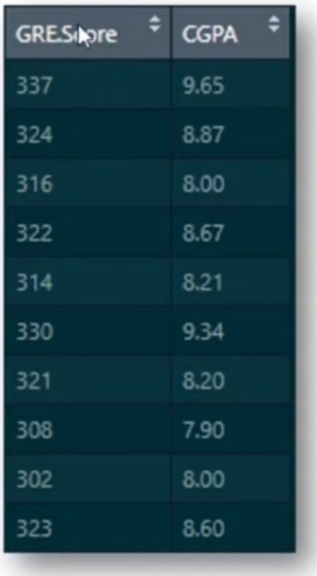
- Yearly
- Quarterly
- Monthly
- Weekly
- Daily
- Hourly



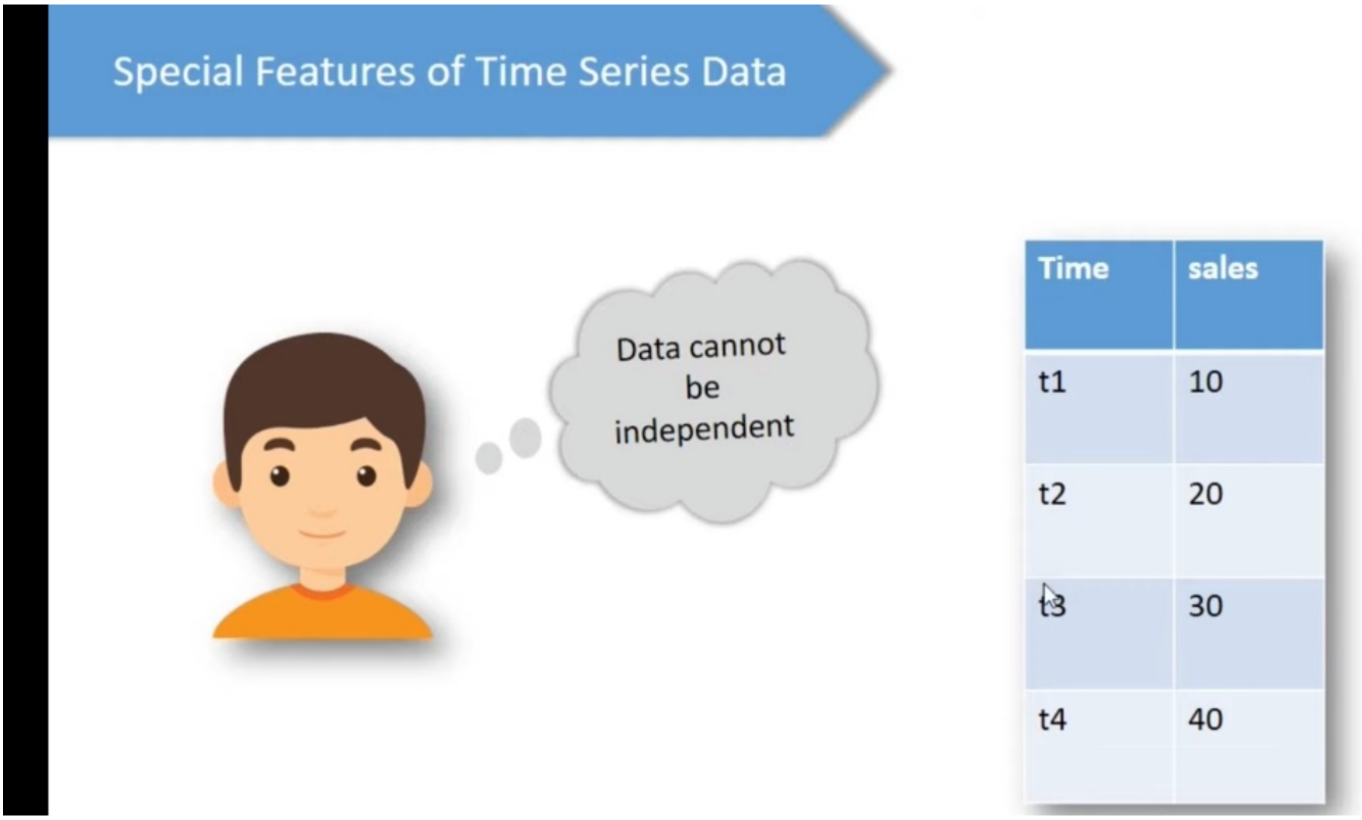
Yearly US GDP

Year	1929	1930	1931	1932	……	1989	1990	1991
US GDP (b. USD)	821.8	748.9	691.3	599.7	……	4739.2	4822.3	4835

Special Features of Time Series Data



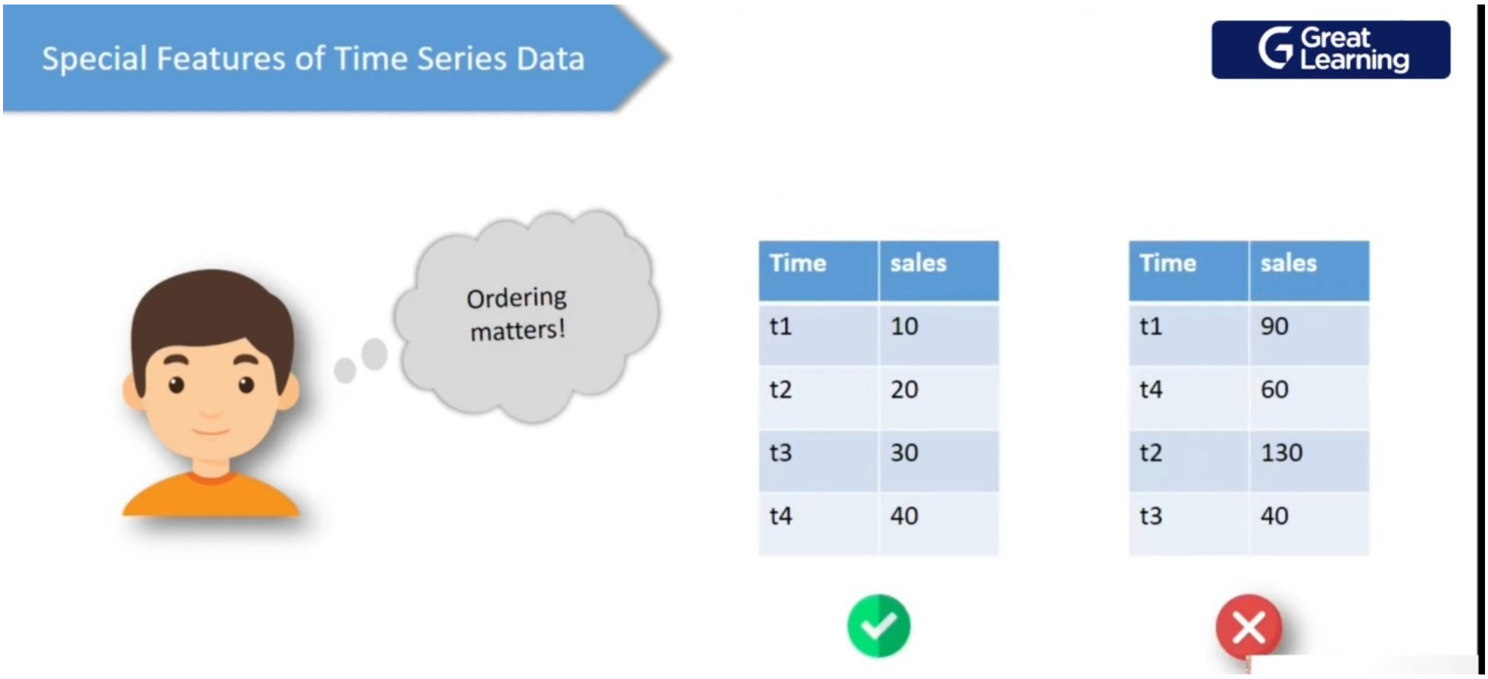
GRE.Score	CGPA
337	9.65
324	8.87
316	8.00
322	8.67
314	8.21
330	9.34
321	8.20
308	7.90
302	8.00
323	8.60



Special Features of Time Series Data

Data cannot be independent

Time	sales
t1	10
t2	20
t3	30
t4	40



Special Features of Time Series Data

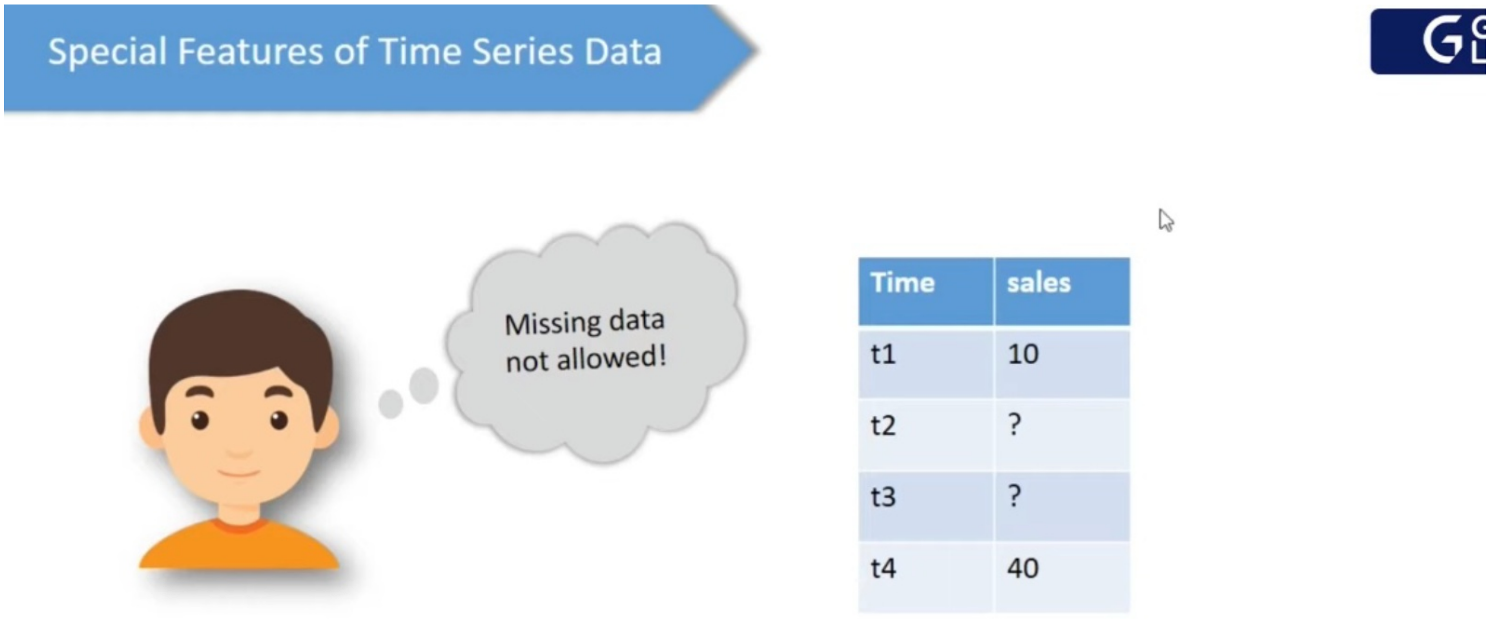
Great Learning

Time	sales
t1	10
t2	20
t3	30
t4	40

✓

Time	sales
t1	90
t4	60
t2	130
t3	40

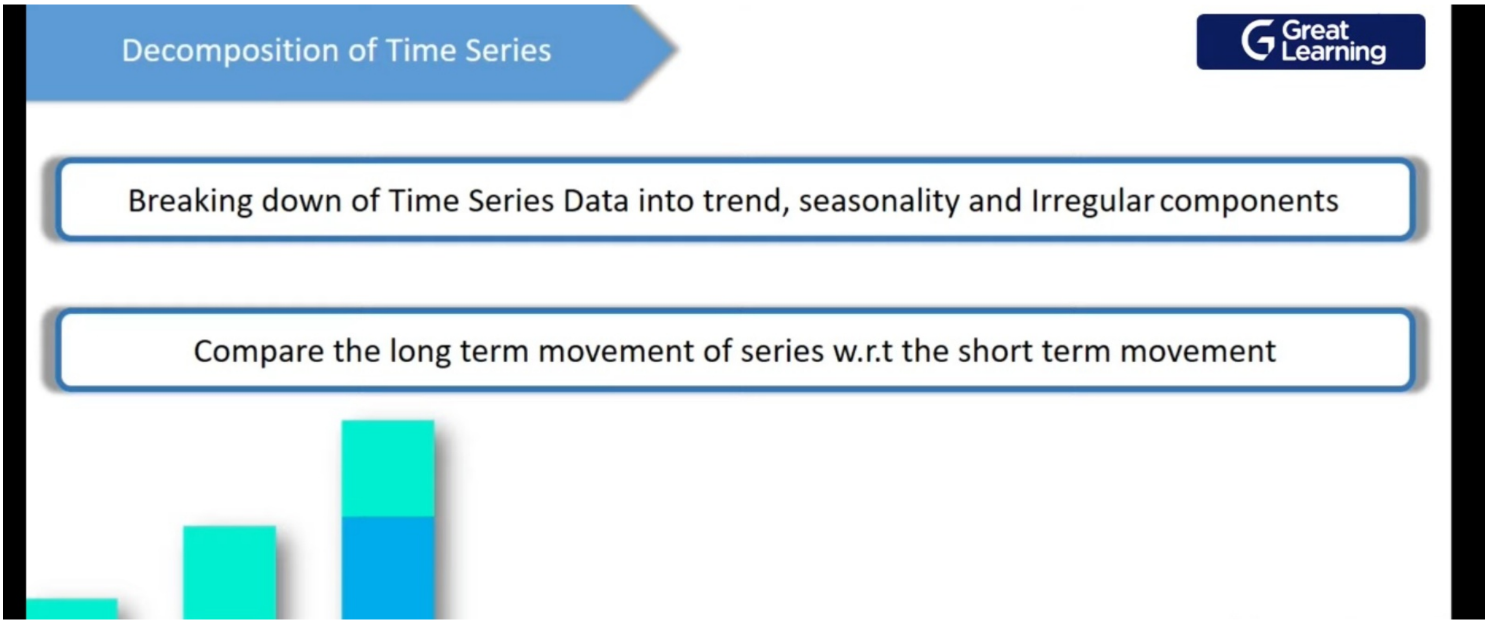
✗



Special Features of Time Series Data

Missing data not allowed!

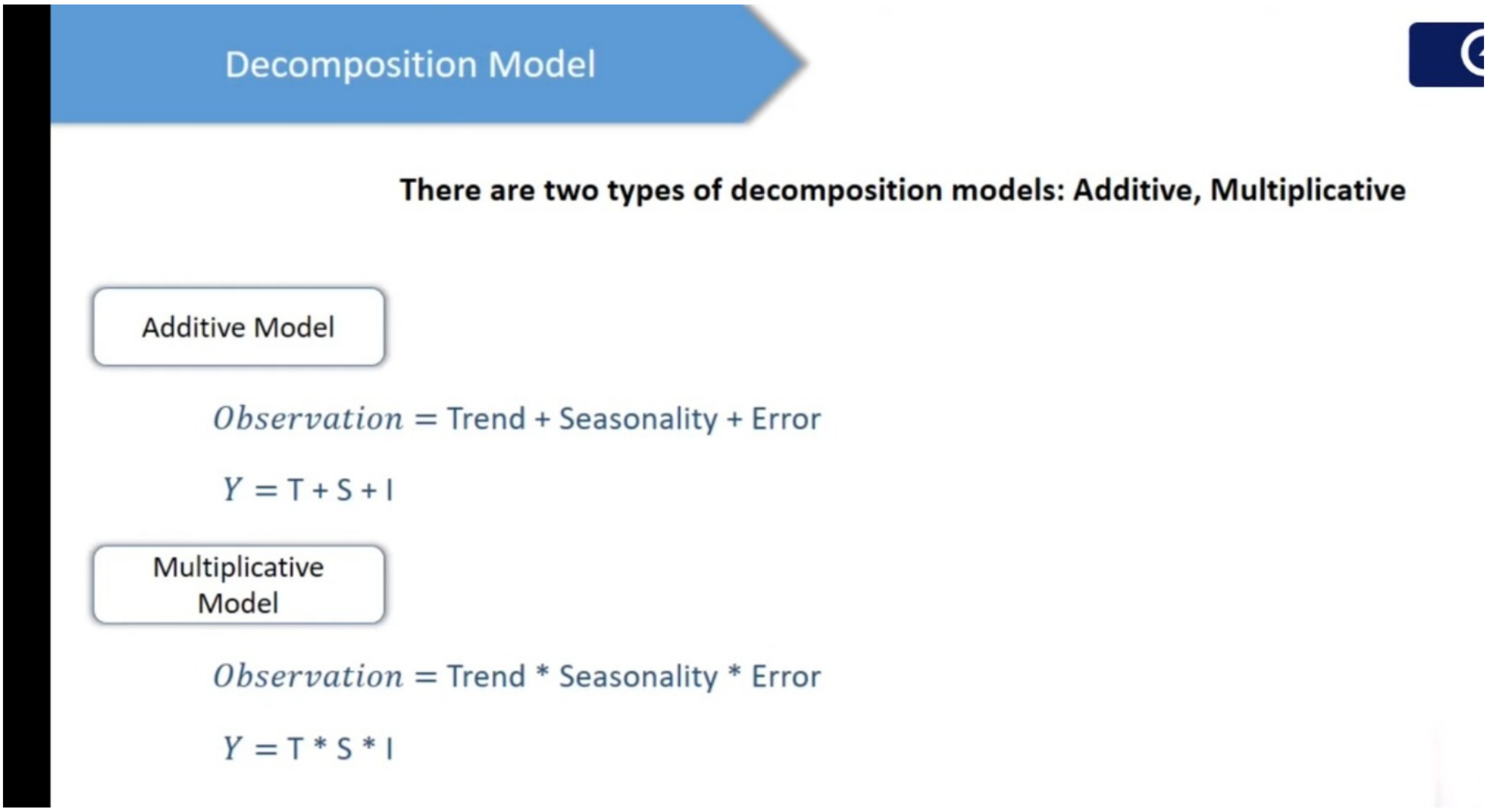
Time	sales
t1	10
t2	?
t3	?
t4	40



Decomposition of Time Series

Breaking down of Time Series Data into trend, seasonality and Irregular components

Compare the long term movement of series w.r.t the short term movement



Decomposition Model

There are two types of decomposition models: Additive, Multiplicative

Additive Model

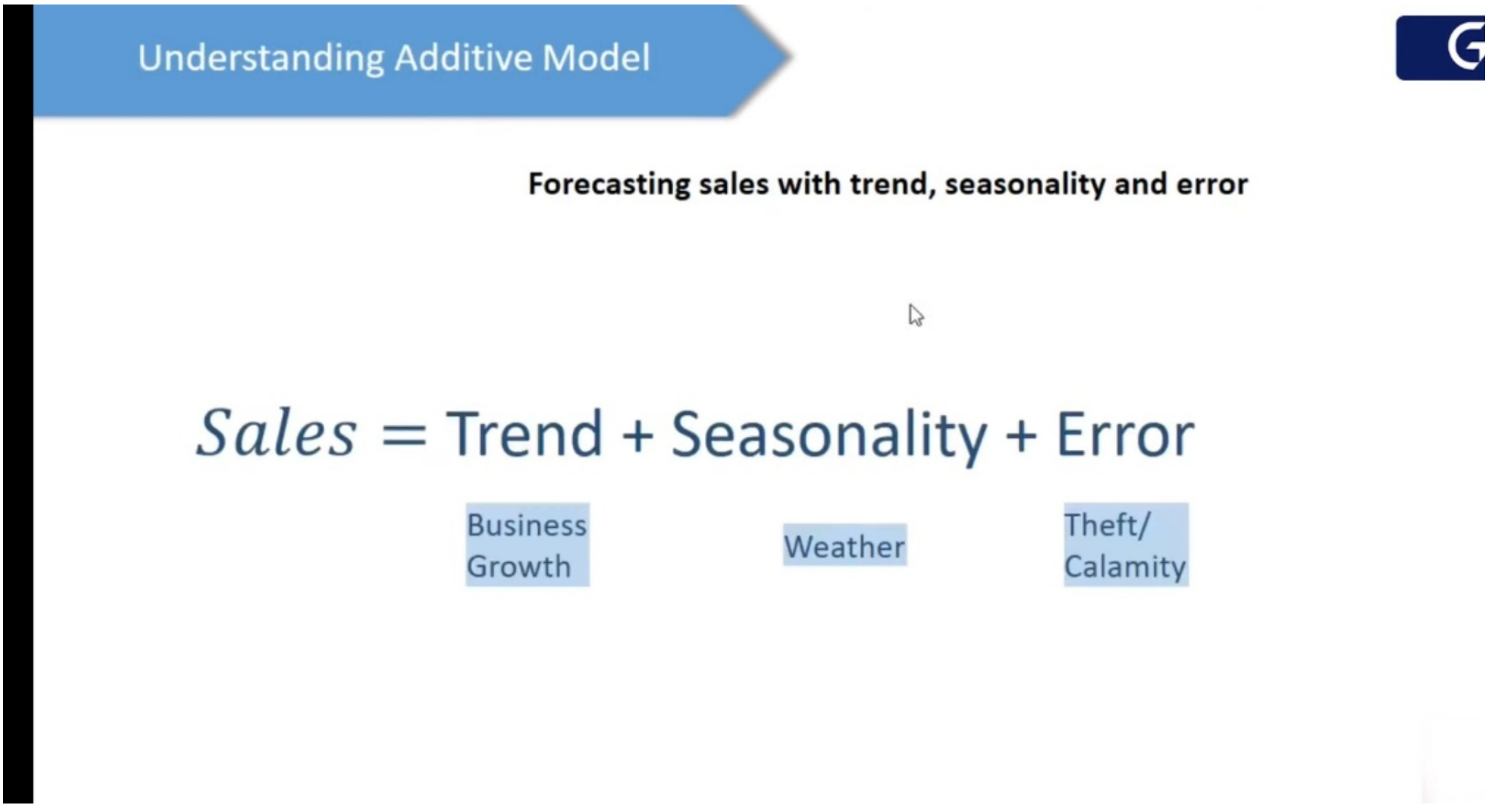
$$Observation = \text{Trend} + \text{Seasonality} + \text{Error}$$

$$Y = \text{T} + \text{S} + \text{I}$$

Multiplicative Model

$$Observation = \text{Trend} * \text{Seasonality} * \text{Error}$$

$$Y = \text{T} * \text{S} * \text{I}$$



Understanding Additive Model

Forecasting sales with trend, seasonality and error

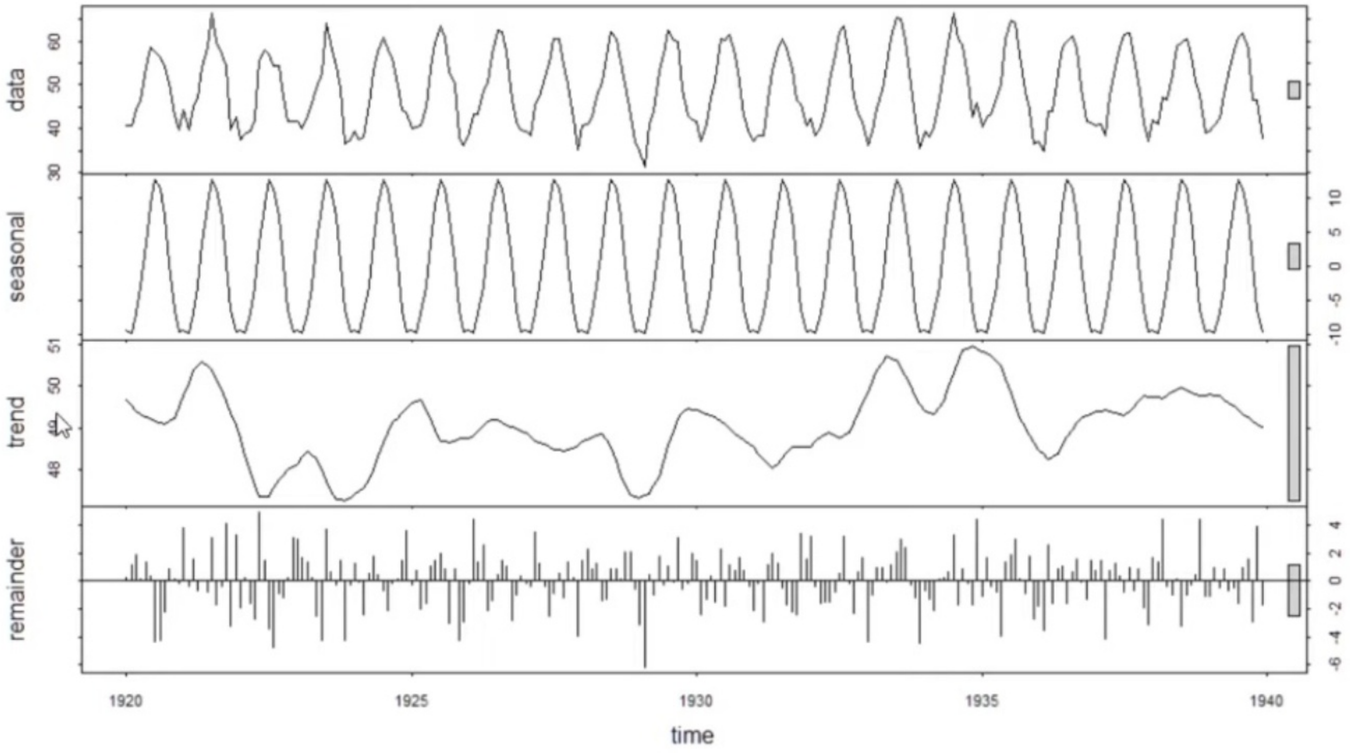
$$Sales = \text{Trend} + \text{Seasonality} + \text{Error}$$

Business Growth

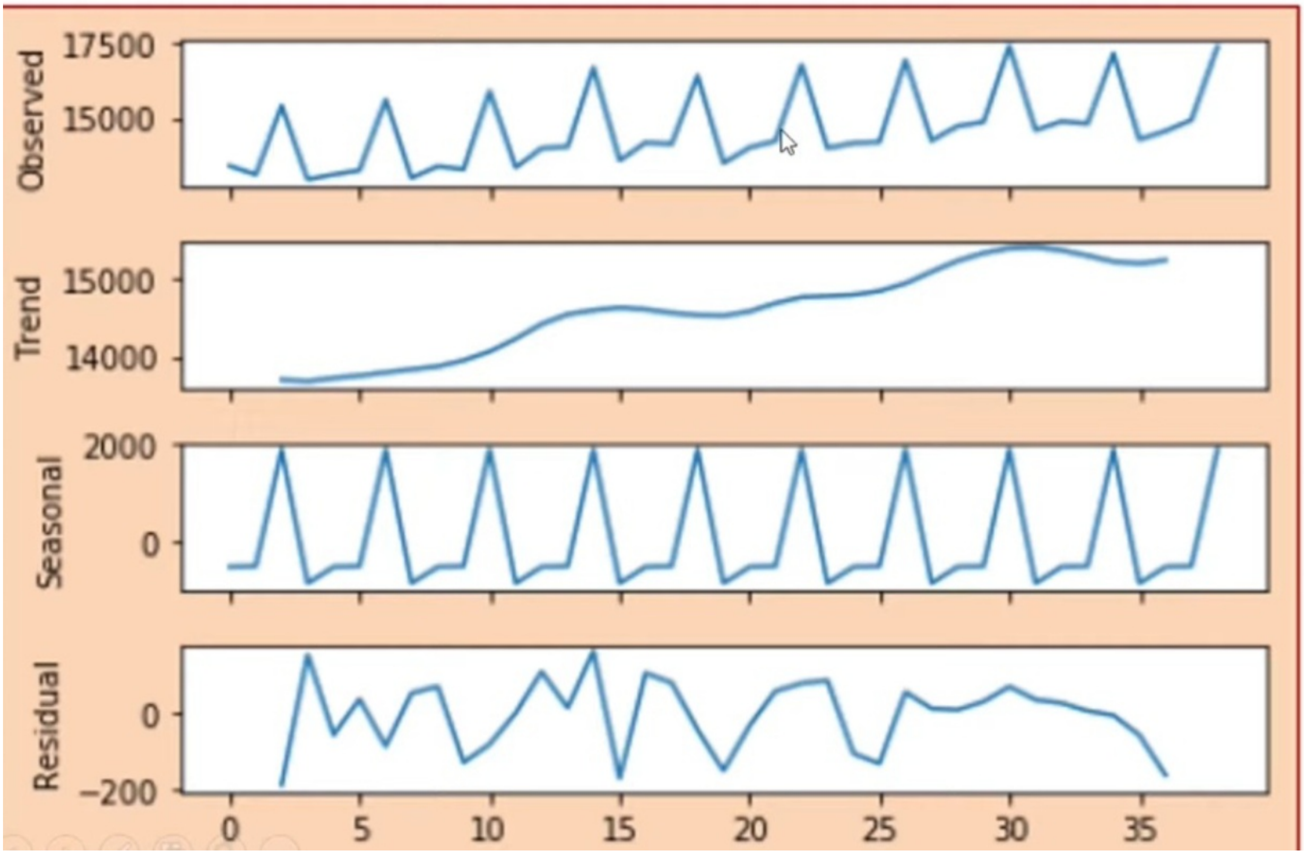
Weather

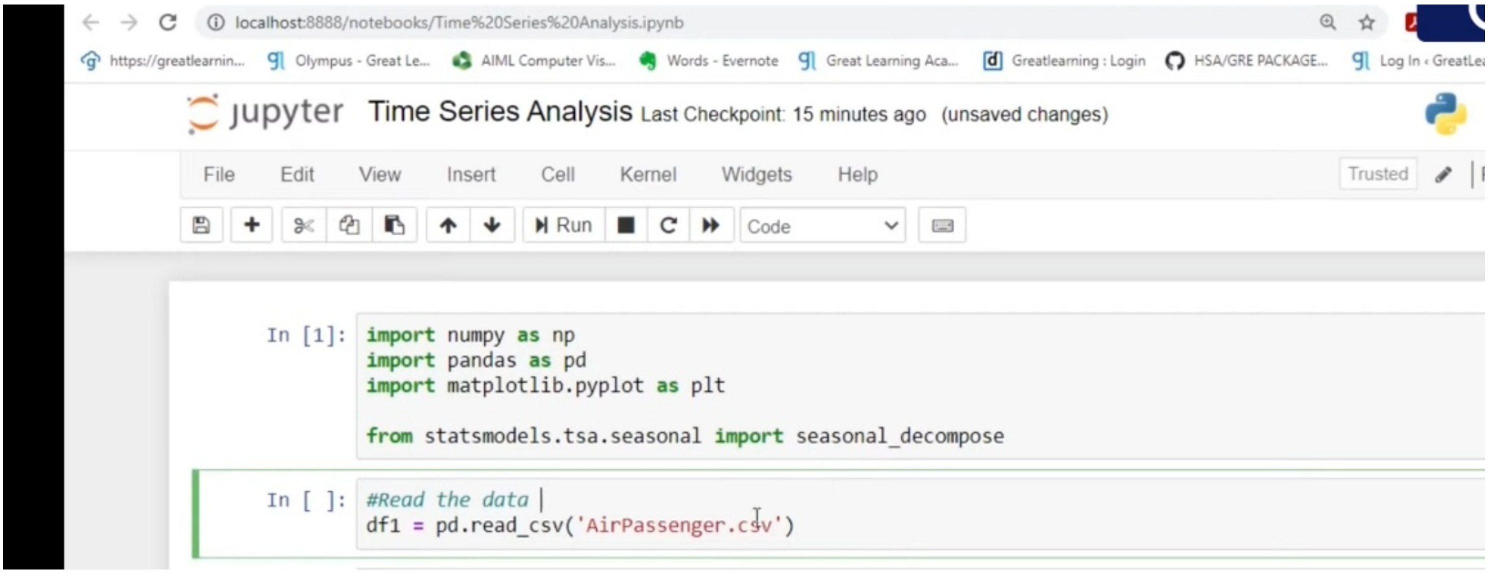
Theft/ Calamity

Decomposition Visualization



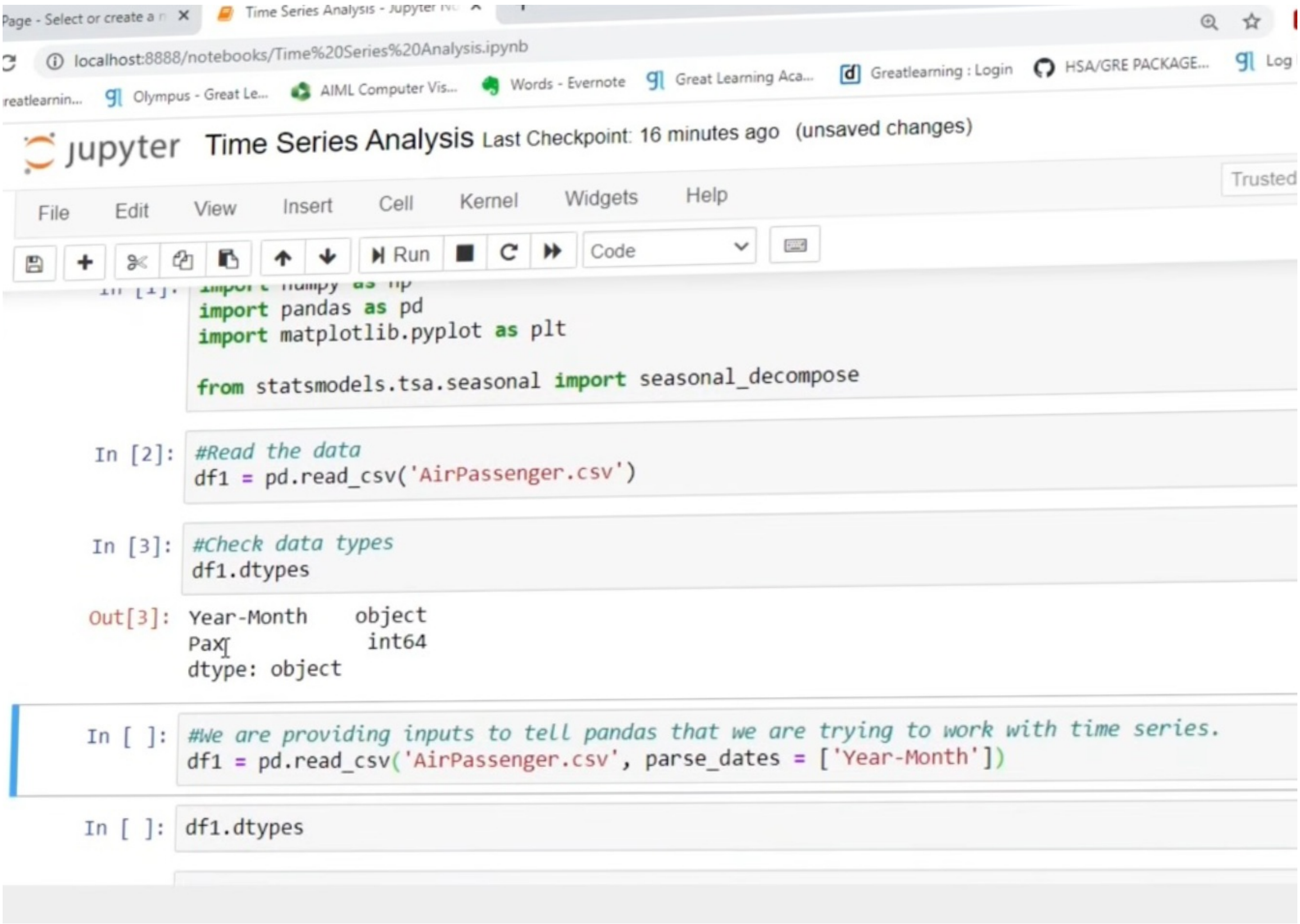
Decomposition Visualization





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

from statsmodels.tsa.seasonal import seasonal_decompose
```

In [ ]:
```
#Read the data
df1 = pd.read_csv('AirPassenger.csv')
```



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

        from statsmodels.tsa.seasonal import seasonal_decompose
```

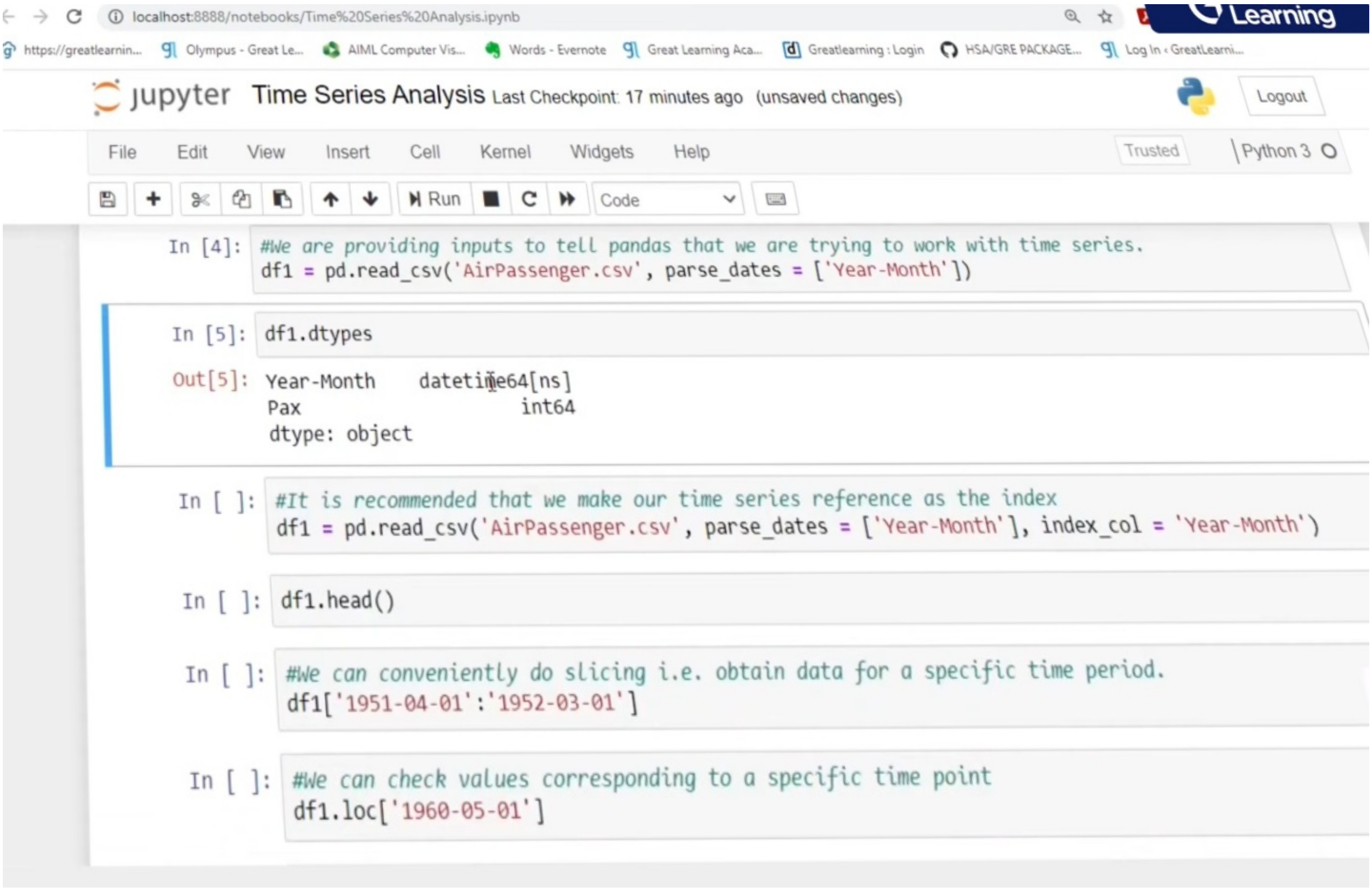
```
In [2]: #Read the data
        df1 = pd.read_csv('AirPassenger.csv')
```

```
In [3]: #Check data types
        df1.dtypes
```

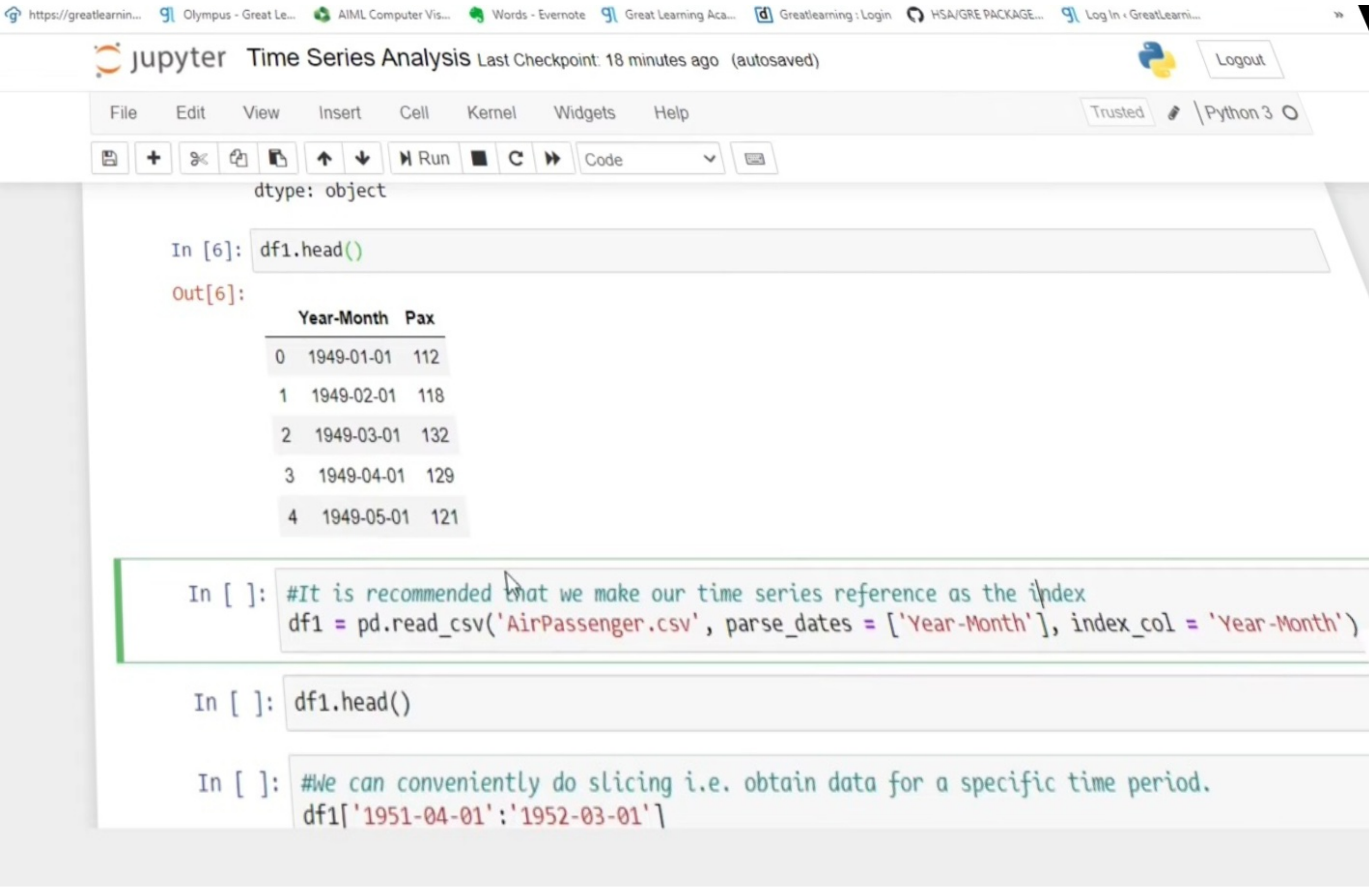
```
Out[3]: Year-Month    object
        Pax            int64
        dtype: object
```

```
In [ ]: #We are providing inputs to tell pandas that we are trying to work with time series.
        df1 = pd.read_csv('AirPassenger.csv', parse_dates = ['Year-Month'])
```

```
In [ ]: df1.dtypes
```



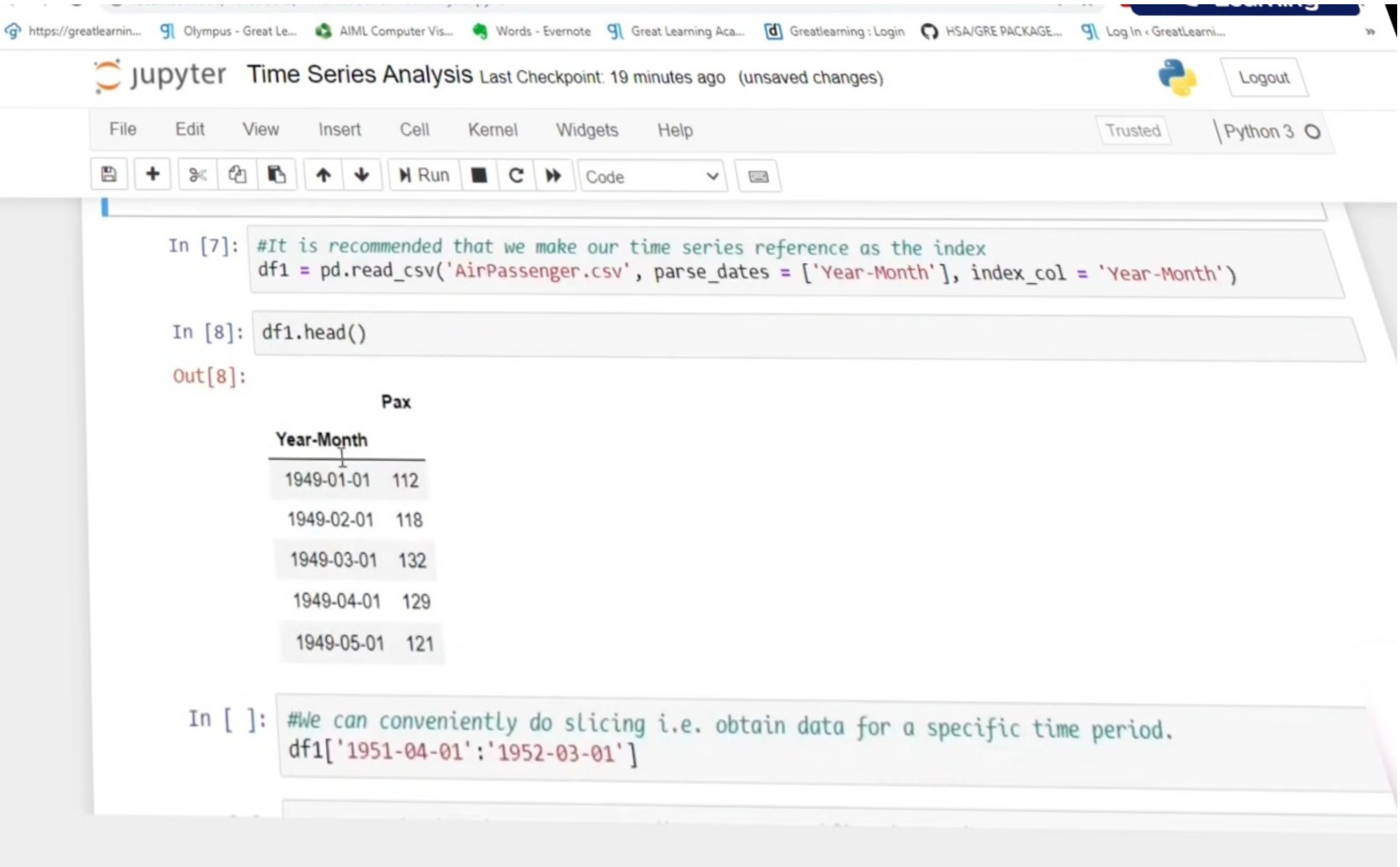
```
In [4]: #We are providing inputs to tell pandas that we are trying to work with time series.
        df1 = pd.read_csv('AirPassenger.csv', parse_dates = ['Year-Month'])

In [5]: df1.dtypes

Out[5]: Year-Month    datetime64[ns]
        Pax                    int64
        dtype: object

In [ ]: #It is recommended that we make our time series reference as the index
        df1 = pd.read_csv('AirPassenger.csv', parse_dates = ['Year-Month'], index_col = 'Year-Month')

In [ ]: df1.head()

In [ ]: #We can conveniently do slicing i.e. obtain data for a specific time period.
        df1['1951-04-01':'1952-03-01']

In [ ]: #We can check values corresponding to a specific time point
        df1.loc['1960-05-01']
```



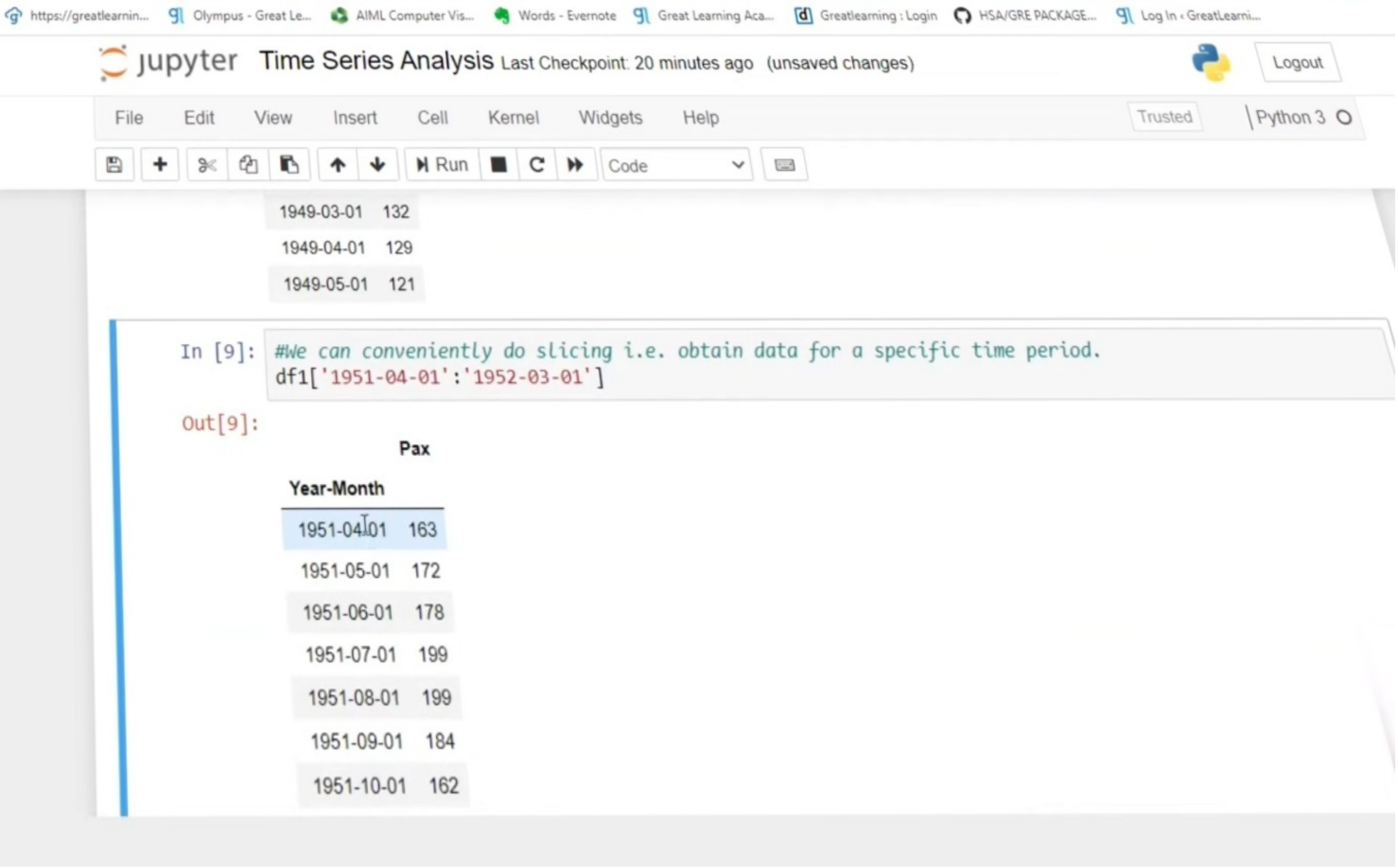
```
dtype: object

In [6]: df1.head()
Out[6]:
```

	Year-Month	Pax
0	1949-01-01	112
1	1949-02-01	118
2	1949-03-01	132
3	1949-04-01	129
4	1949-05-01	121

```
In [ ]: #It is recommended that we make our time series reference as the index
        df1 = pd.read_csv('AirPassenger.csv', parse_dates = ['Year-Month'], index_col = 'Year-Month')

In [ ]: df1.head()

In [ ]: #We can conveniently do slicing i.e. obtain data for a specific time period.
        df1['1951-04-01':'1952-03-01']
```



```
In [7]: #It is recommended that we make our time series reference as the index
        df1 = pd.read_csv('AirPassenger.csv', parse_dates = ['Year-Month'], index_col = 'Year-Month')

In [8]: df1.head()

Out[8]:
                Pax
Year-Month
1949-01-01      112
1949-02-01      118
1949-03-01      132
1949-04-01      129
1949-05-01      121

In [ ]: #We can conveniently do slicing i.e. obtain data for a specific time period.
        df1['1951-04-01':'1952-03-01']
```



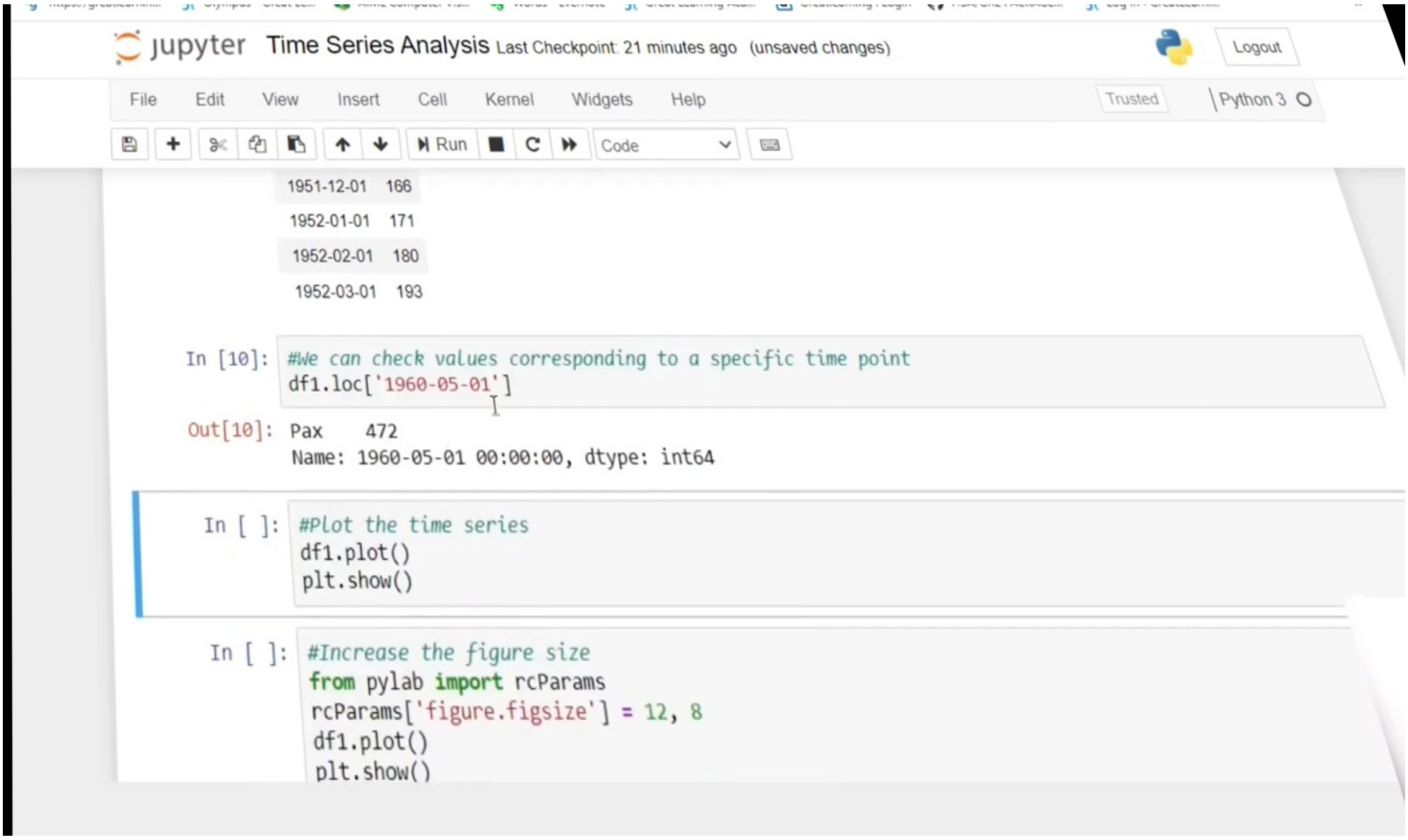
Time Series Analysis Last Checkpoint: 20 minutes ago (unsaved changes)

1949-03-01	132
1949-04-01	129
1949-05-01	121

```
In [9]: #We can conveniently do slicing i.e. obtain data for a specific time period.
        df1['1951-04-01':'1952-03-01']
```

Out[9]:

Year-Month	Pax
1951-04-01	163
1951-05-01	172
1951-06-01	178
1951-07-01	199
1951-08-01	199
1951-09-01	184
1951-10-01	162

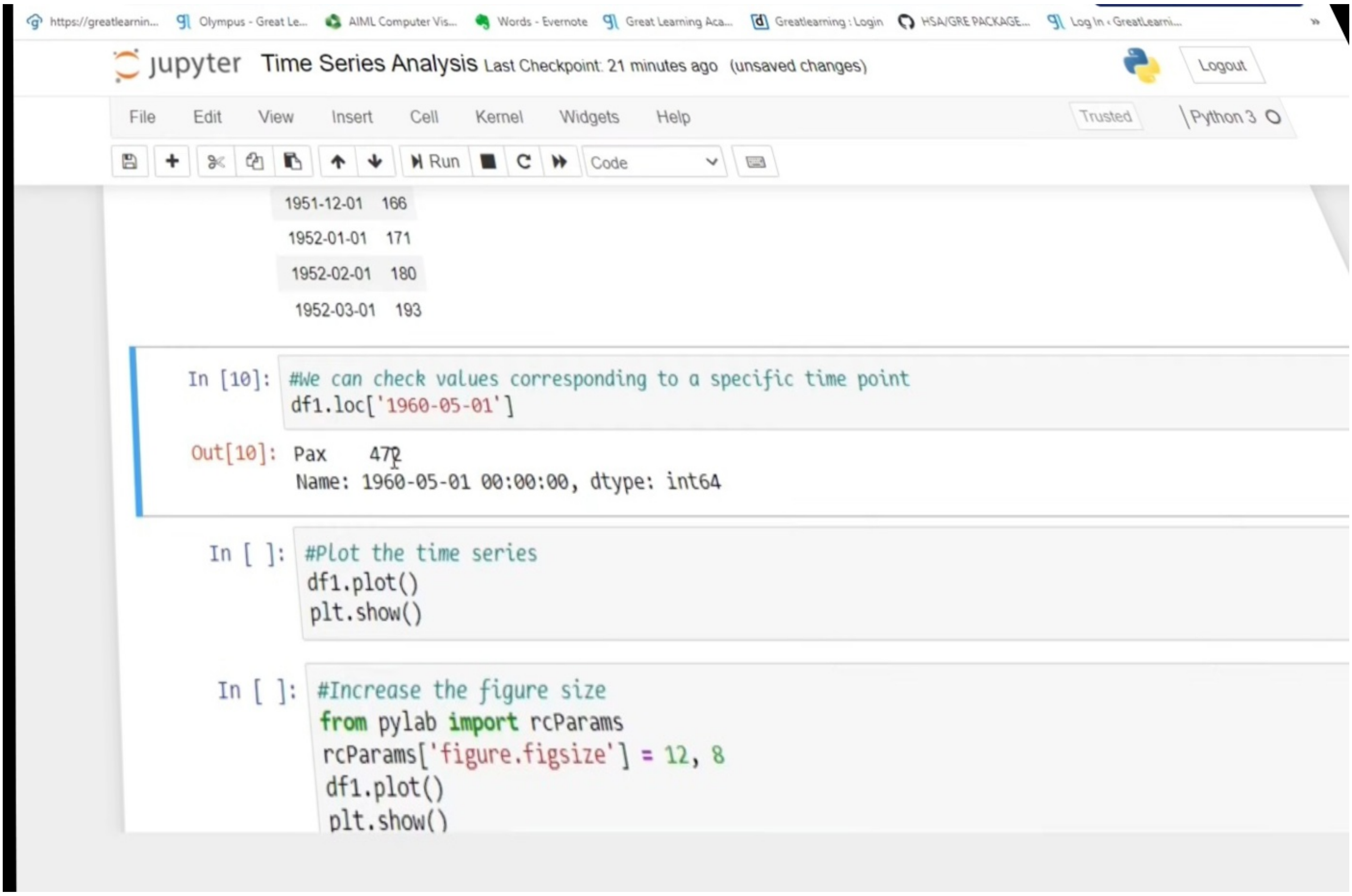


```
1951-12-01  166
1952-01-01  171
1952-02-01  180
1952-03-01  193
```

```
In [10]: #We can check values corresponding to a specific time point
         df1.loc['1960-05-01']

Out[10]: Pax     472
         Name: 1960-05-01 00:00:00, dtype: int64
```

```
In [ ]: #Plot the time series
        df1.plot()
        plt.show()
```

```
In [ ]: #Increase the figure size
        from pylab import rcParams
        rcParams['figure.figsize'] = 12, 8
        df1.plot()
        plt.show()
```



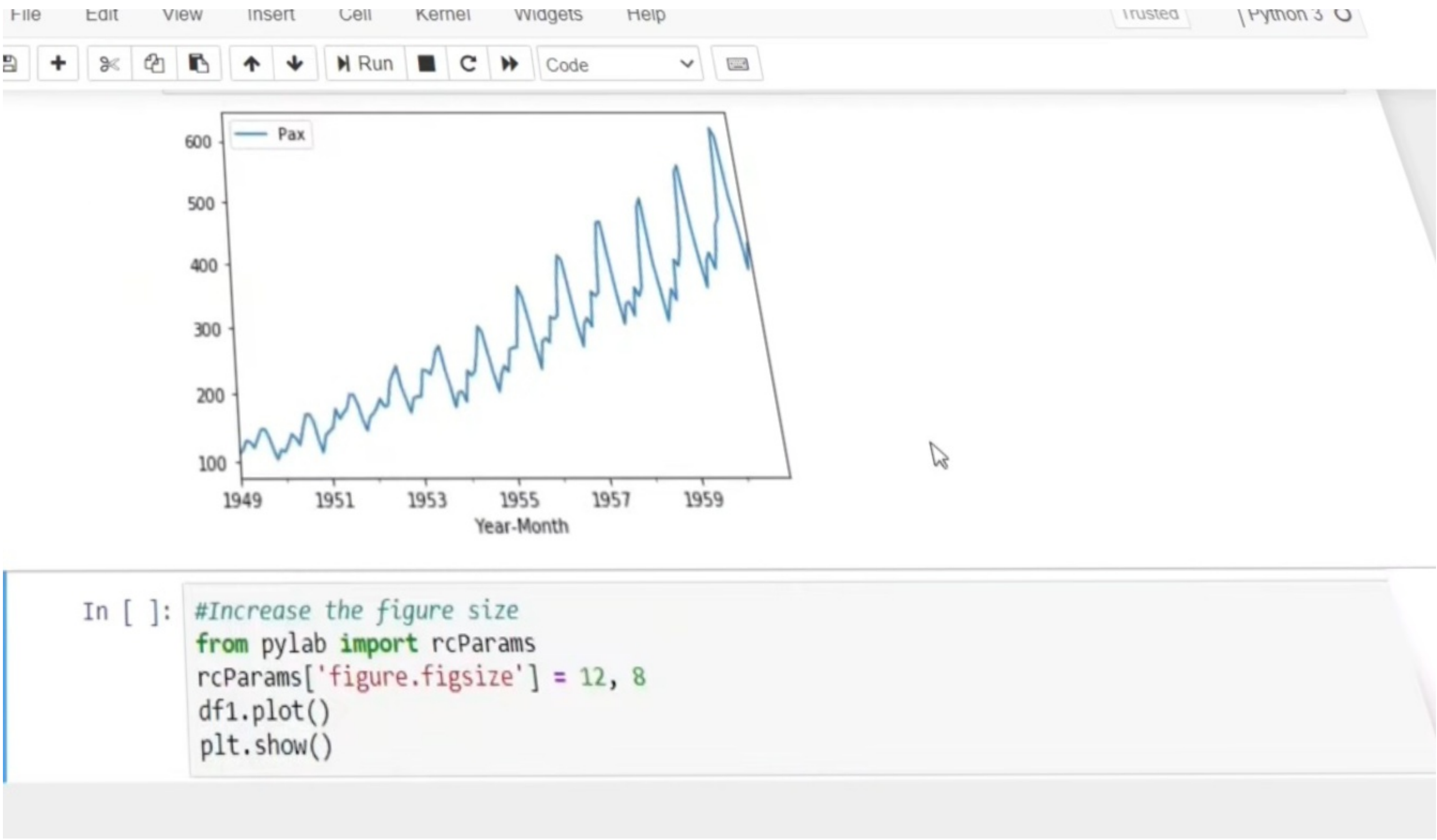
Time Series Analysis Last Checkpoint: 21 minutes ago (unsaved changes)

```
1951-12-01  166
1952-01-01  171
1952-02-01  180
1952-03-01  193
```

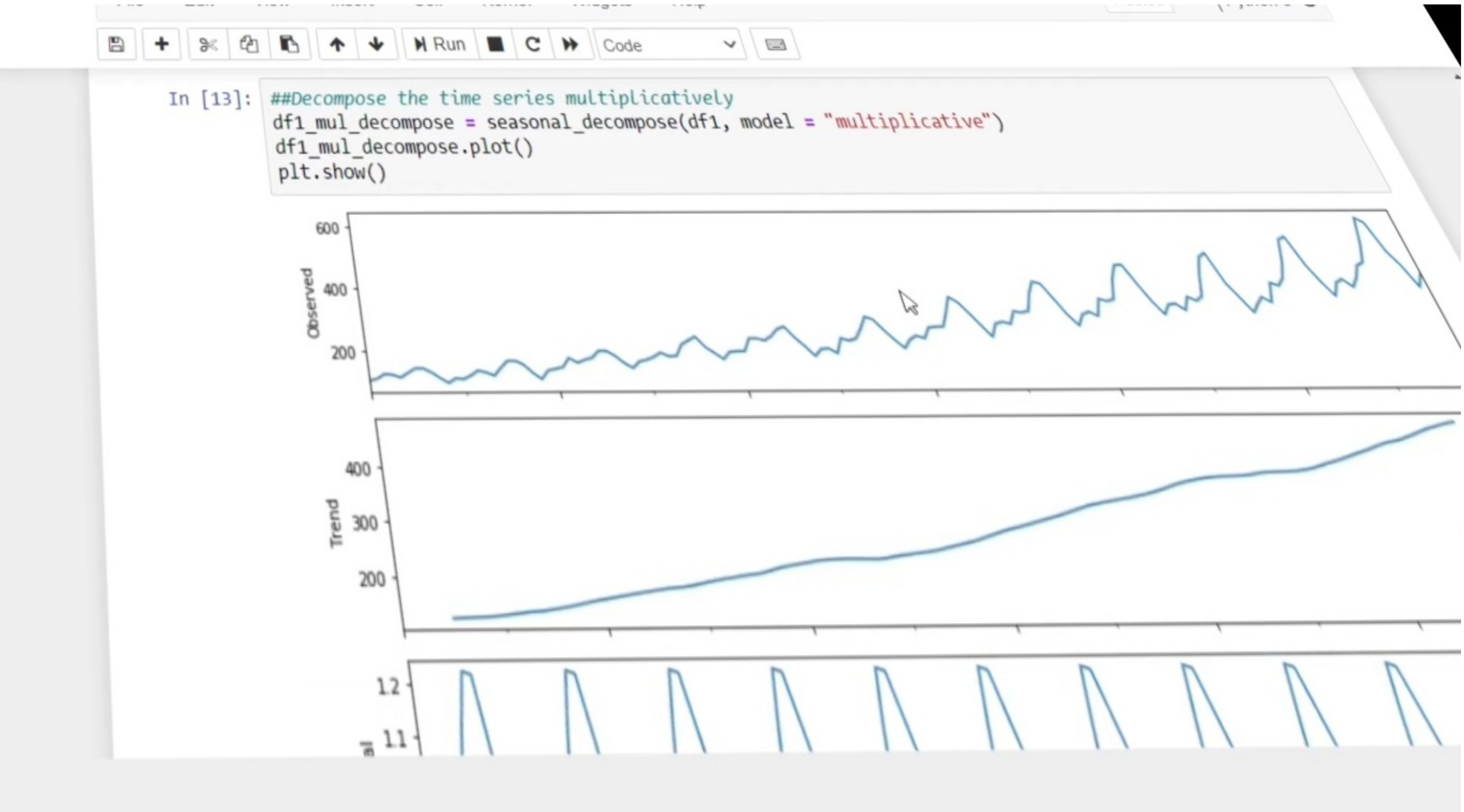
```
In [10]: #We can check values corresponding to a specific time point
         df1.loc['1960-05-01']

Out[10]: Pax    472
         Name: 1960-05-01 00:00:00, dtype: int64
```

```
In [ ]: #Plot the time series
        df1.plot()
        plt.show()
```

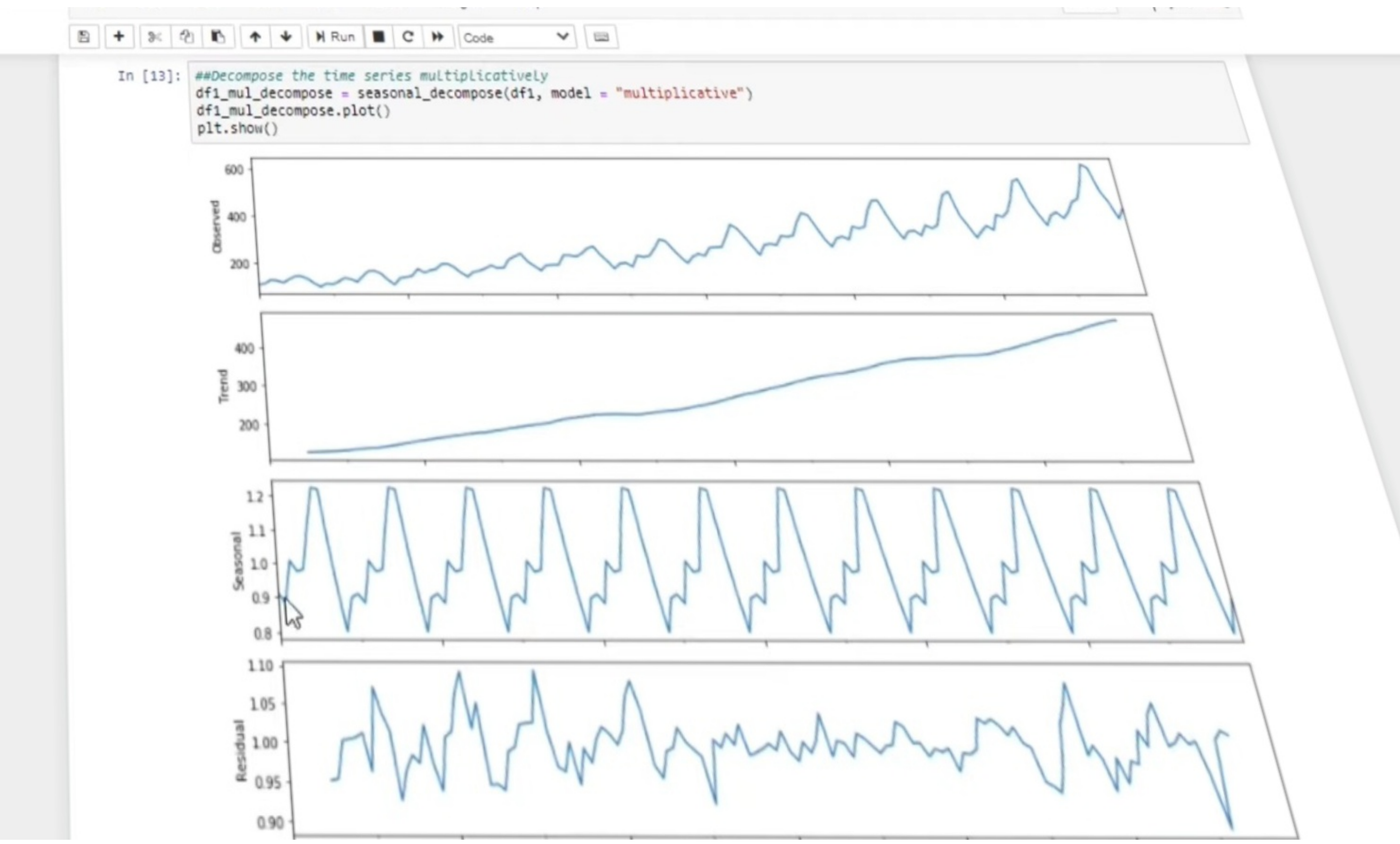
```
In [ ]: #Increase the figure size
        from pylab import rcParams
        rcParams['figure.figsize'] = 12, 8
        df1.plot()
        plt.show()
```



```
In [ ]: #Increase the figure size
        from pylab import rcParams
        rcParams['figure.figsize'] = 12, 8
        df1.plot()
        plt.show()
```



```
##Decompose the time series multiplicatively
df1_mul_decompose = seasonal_decompose(df1, model = "multiplicative")
df1_mul_decompose.plot()
plt.show()
```



```
##Decompose the time series multiplicatively
df1_mul_decompose = seasonal_decompose(df1, model = "multiplicative")
df1_mul_decompose.plot()
plt.show()
```