Machine Learning Foundation

Course 6, Part b: Decomposition LAB

Introduction

Here we will discuss decomposition of your time series data into three components:

1. Trend
2. Seasonality
3. Residual

Learning Outcomes

What will we accomplish?

1. First, we'll separately construct each time series component.
2. Then we'll bring the three components together to create a simulated time series dataset.
3. Finally, we'll leverage time series model decomposition in Python to deconstruct the series.

We'll complete this process for the two most common processes: additive and multiplicative.

The whole point of this exercise is to get you familiar with time series data and simple modeling approaches.

You should walk away from this tutorial with:

1. A practical understanding of the three components of time series data
2. A better understanding of additive and multiplicative models
3. A simple way to decompose time series data in Python

Imports

In [1]:

```
import sys
import statsmodels as ss
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import os
import seaborn as sns
os.chdir('C:/Users/julia/OneDrive/Desktop/Coursera/time series Python/')
from colorsetup import colors, palette
sns.set_palette(palette)
```

Section 1: Construct Trend, Seasonality, and Residual Components

In this lesson, we will individually construct fictional trend, seasonality, and residual components. This is

In [11]:

```
time = np.arange(1, 51)
```

Now we need to create a trend. Let's pretend we have a sensor measuring demand for electricity. We'll ignore units to keep things simple.

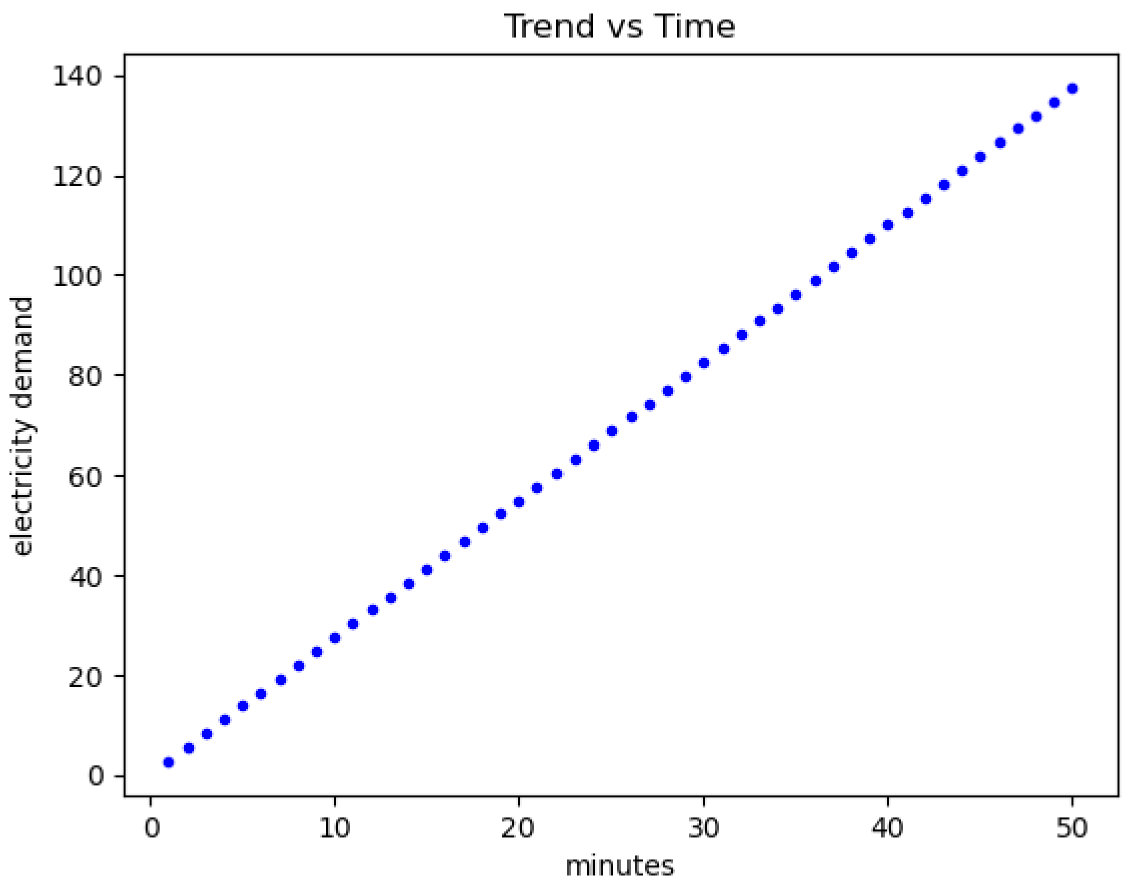
In [12]:

```
trend = time * 2.75
```

What folllows is a plot showing trend as a function of time.

In [13]:

```
plt.plot(time, trend, 'b.')
plt.title("Trend vs Time")
plt.xlabel("minutes")
plt.ylabel("electricity demand");
```



Great! Now let's generate a seasonal component.

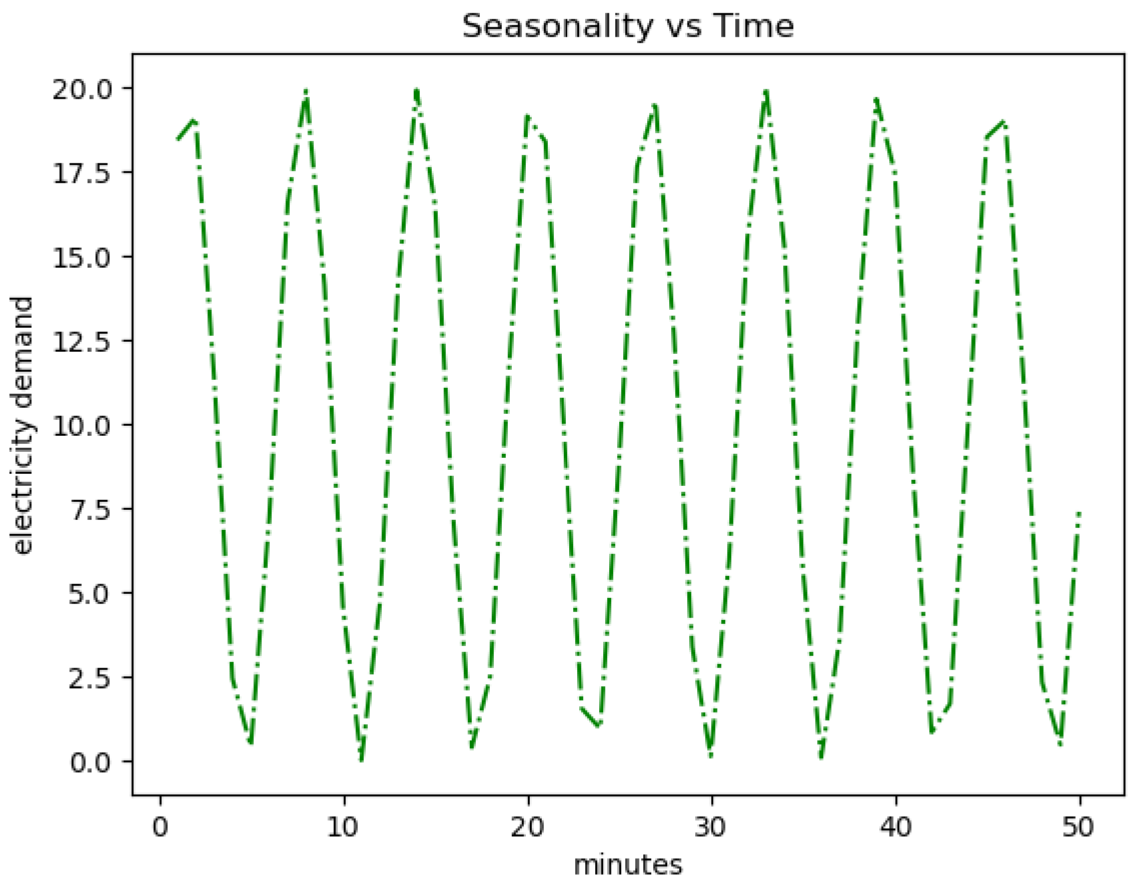
In [14]:

```
seasonal = 10 + np.sin(time) * 10
```

Naturally, we'll want to plot our seasonality.

In [15]:

```
plt.plot(time, seasonal, 'g-.')
plt.title("Seasonality vs Time")
plt.xlabel("minutes")
plt.ylabel("electricity demand");
```



And now to construct the residual component.

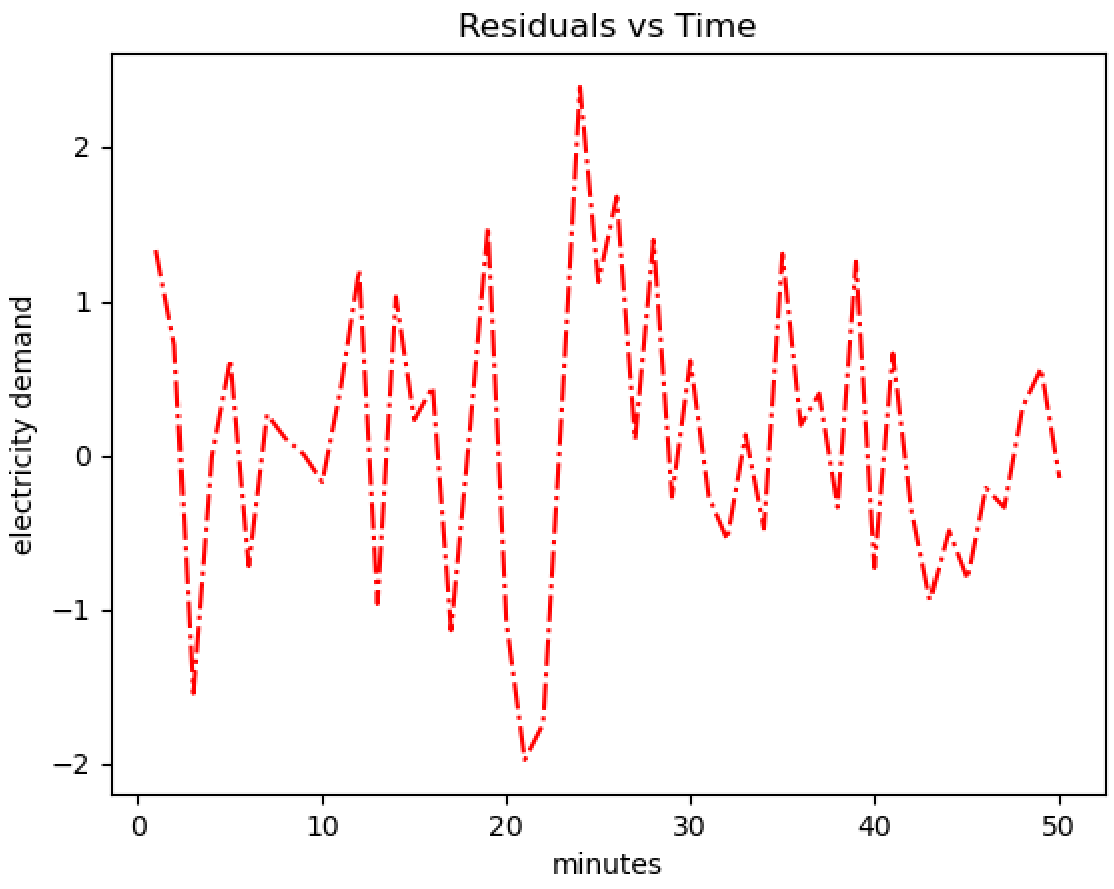
In [16]:

```
np.random.seed(10)  # reproducible results
residual = np.random.normal(loc=0.0, scale=1, size=len(time))
```

A quick plot of residuals...

In [17]:

```
plt.plot(time, residual, 'r-.')
plt.title("Residuals vs Time")
plt.xlabel("minutes")
plt.ylabel("electricity demand");
```



Quick recap: We created data for the fictional demand of electricity as measured by a fictional sensor. We did this by creating the three necessary components of time series data, which include trend, seasonality, and residuals.

Next step: Aggregate the three components to create time series data. We'll do this for two different types of models: additive and multiplicative.

Section 2: Aggregate Trend, Seasonality, and Residual Components

Additive Time Series

Remember the equation for additive time series is simply: $O_t = T_t + S_t + R_t$

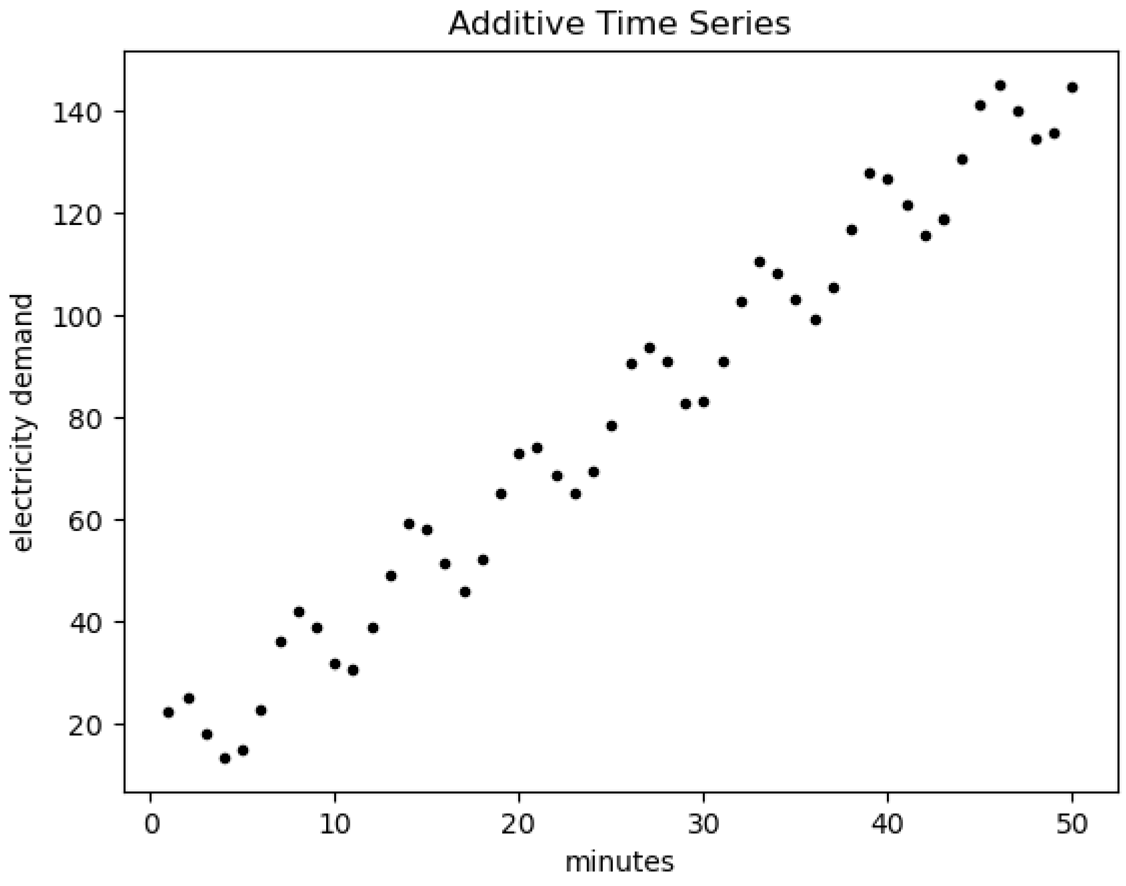
O_t is the output
T_t is the trend
S_t is the seasonality
R_t is the residual
$_t$ is a variable representing a particular point in time

In [18]:

```
additive = trend + seasonal + residual
```

In [19]:

```
plt.plot(time, additive, 'k.')
plt.title("Additive Time Series")
plt.xlabel("minutes")
plt.ylabel("electricity demand");
```



We clearly see trend and seasonality in the plot above, as expected.

Now let's build a multiplicative time series dataset.

Multiplicative Time Series

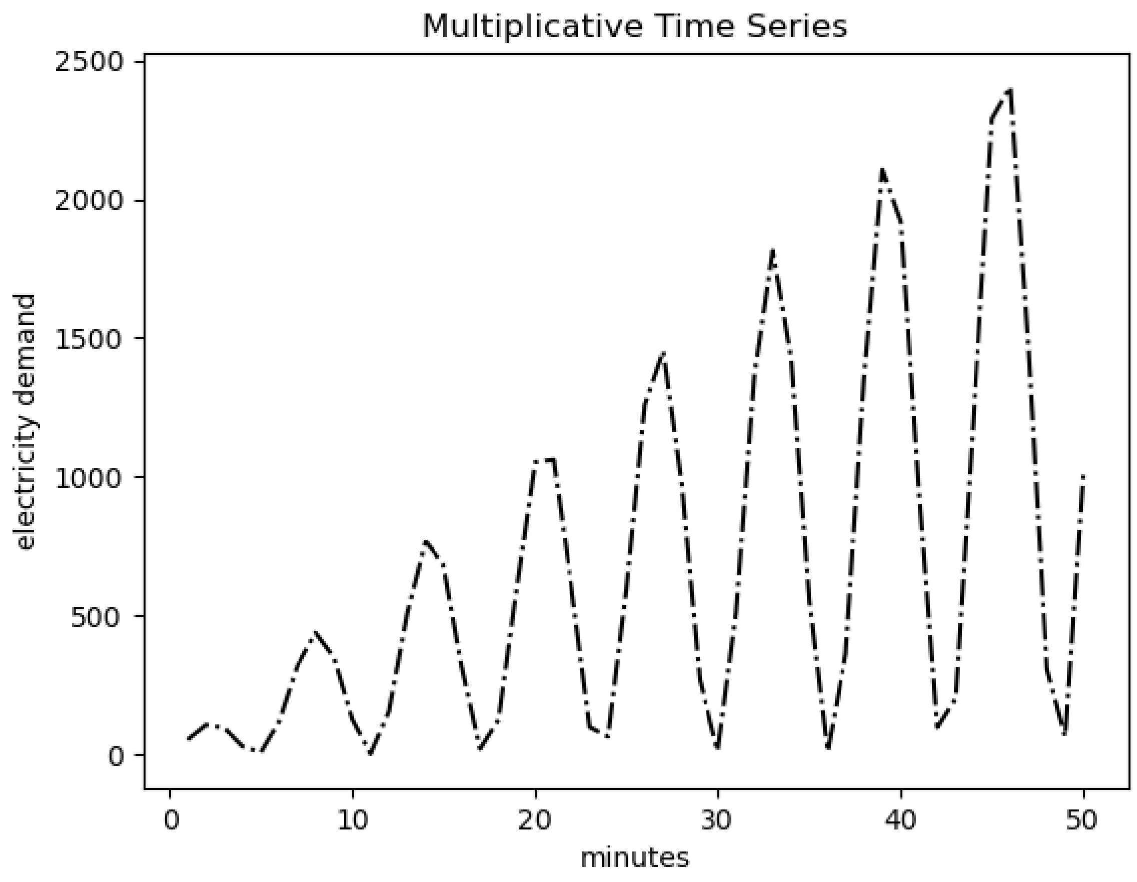
Remember the equation for multiplicative time series is simply: $O_t = T_t * S_t * R_t$

In [20]:

```
# ignoring residual to make pattern more apparent
ignored_residual = np.ones_like(residual)
multiplicative = trend * seasonal * ignored_residual
```

In [21]:

```
plt.plot(time, multiplicative, 'k-.')
plt.title("Multiplicative Time Series")
plt.xlabel("minutes")
plt.ylabel("electricity demand");
```



Exercise #1

In this exercise, your task is two-fold:

1. Determine if **Dataset_A** and **Dataset_B** are additive or multiplicative time series.
2. Determine the frequency of the seasonal component.

Set Path / Load Datasets

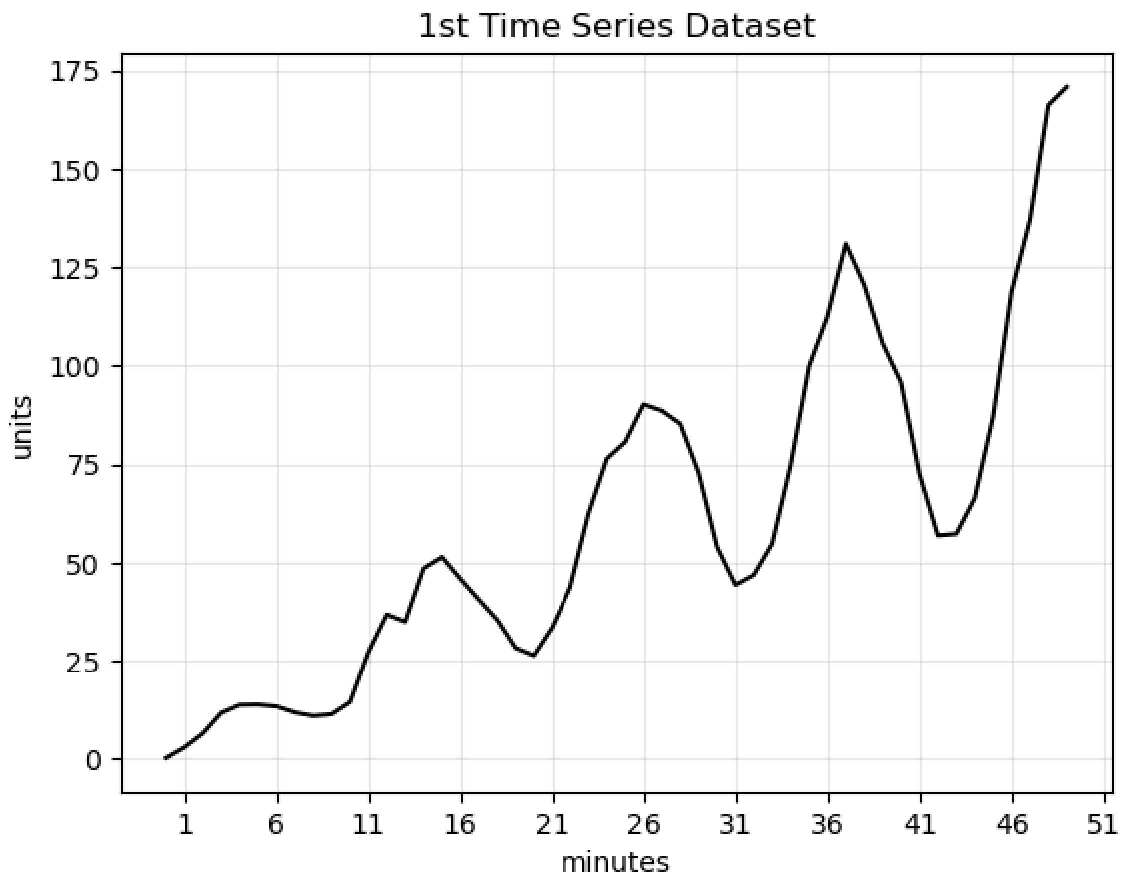
In [22]:

```
### BEGIN SOLUTION
# get data
path_to_file = "./"

time = np.arange(0, 50)
dataset_A = np.load(path_to_file + "dataset_A.npy")
dataset_B = np.load(path_to_file + "dataset_B.npy")
```

Plot Dataset_A

In [23]:

```
plt.plot(time, dataset_A, 'k-')
plt.title("1st Time Series Dataset")
plt.xlabel("minutes")
plt.xticks(np.linspace(1,51,11, dtype='int'))
plt.ylabel("units")
plt.grid(alpha=0.3);
```



Additive or Multiplicative?

Dataset A is multiplicative. We know this because magnitude of the seasonality component in the 1st plot above is dependent on the trend.

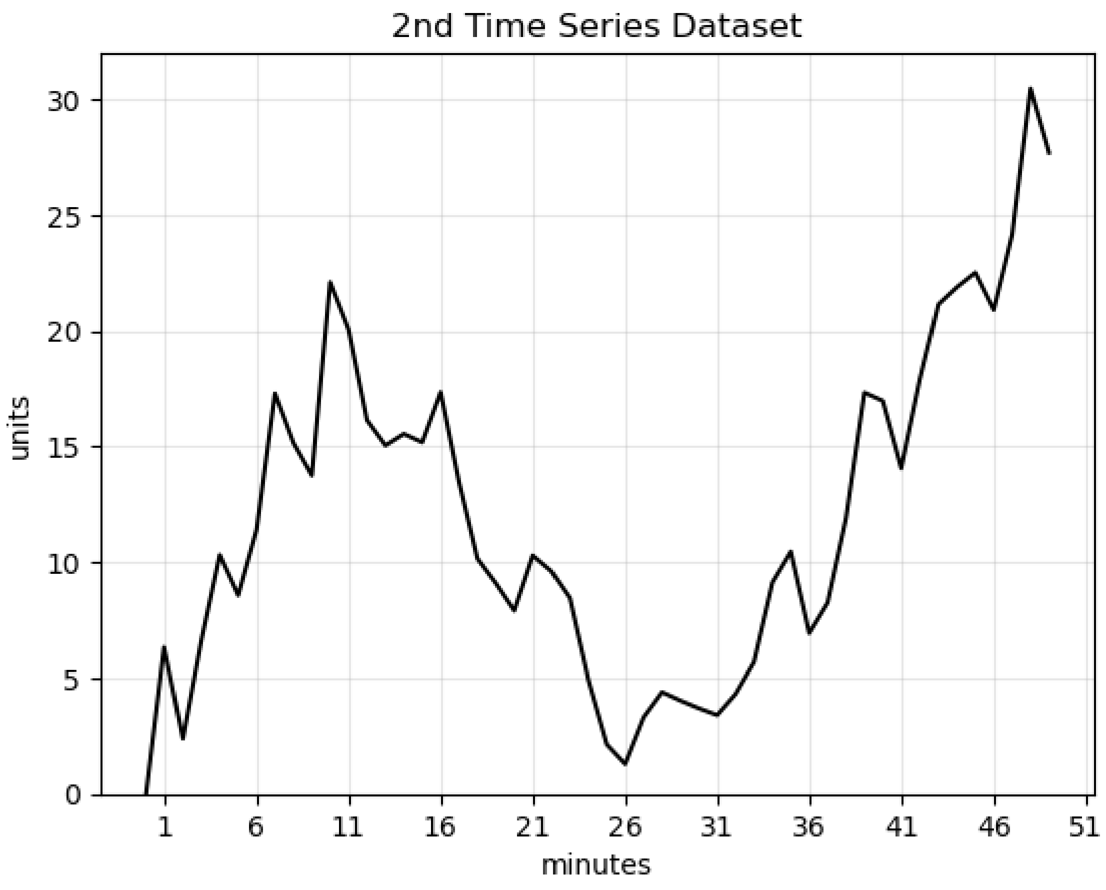
Frequency of Seasonal Component?

The frequency appears to be ~6 for Dataset_A.

Plot Dataset_B

In [24]:

```
plt.plot(time, dataset_B, 'k-')
plt.title("2nd Time Series Dataset")
plt.xlabel("minutes")
plt.xticks(np.linspace(1,51,11, dtype='int'))
plt.ylabel("units")
plt.ylim(0, 32)
plt.grid(alpha=0.3);
```



Additive or Multiplicative?

Dataset B is additive. We know this because the magnitude of the seasonality component in the 2nd plot above is independent of the trend.

Frequency of Seasonal Component?

The frequency appears to be ~3 for Dataset_B.

END SOLUTION

Section 3: Time Series Decomposition

Now let's reverse directions. We have additive and multiplicative data. Let's decompose them into their three components.

Please note that the goal here is NOT to understand HOW the decomposition works. Rather, the goal is simply to see that the time series data that we previously constructed can be decomposed into its three constituent parts: trend, seasonality, and residual.

> *Optional: look [here (http://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html)](http://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html) for details on how the following method works.*

Let's take a look.

Additive Decomposition

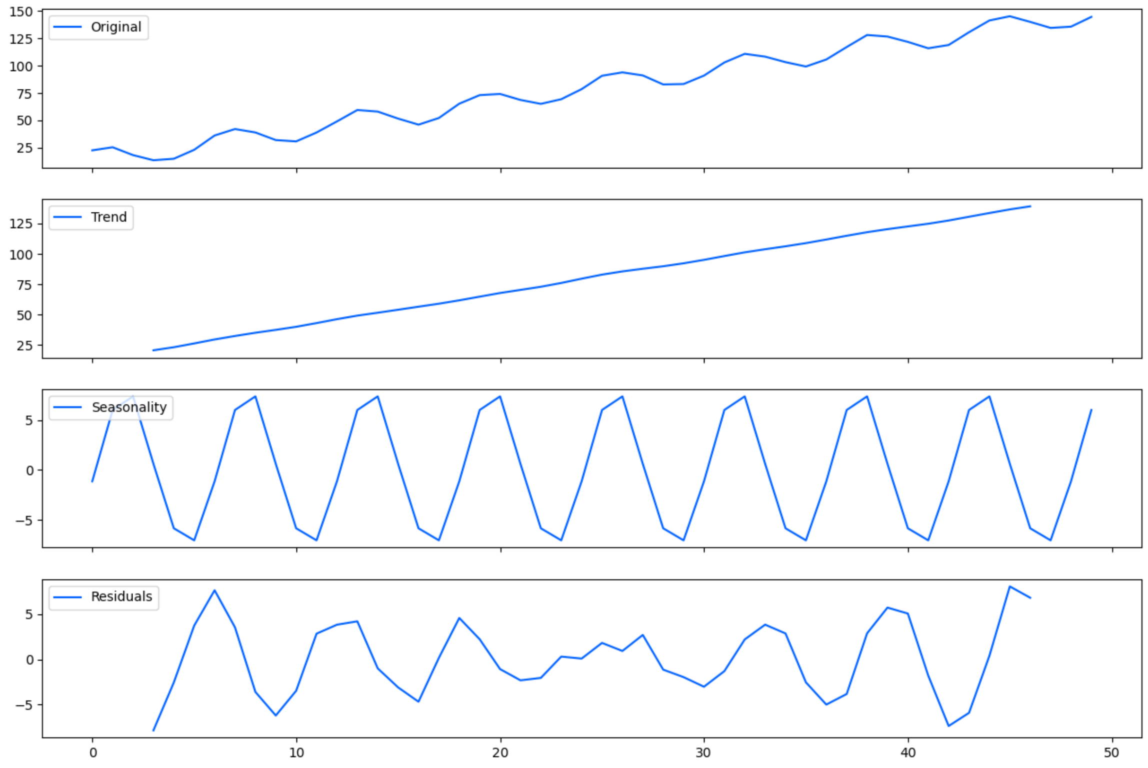
In [25]:

```
from statsmodels.tsa.seasonal import seasonal_decompose

ss_decomposition = seasonal_decompose(x=additive, model='additive', period=6)
estimated_trend = ss_decomposition.trend
estimated_seasonal = ss_decomposition.seasonal
estimated_residual = ss_decomposition.resid
```

Additive Plots

In [26]:

```
fig, axes = plt.subplots(4, 1, sharex=True, sharey=False)
fig.set_figheight(10)
fig.set_figwidth(15)

axes[0].plot(additive, label='Original')
axes[0].legend(loc='upper left');

axes[1].plot(estimated_trend, label='Trend')
axes[1].legend(loc='upper left');

axes[2].plot(estimated_seasonal, label='Seasonality')
axes[2].legend(loc='upper left');

axes[3].plot(estimated_residual, label='Residuals')
axes[3].legend(loc='upper left');
```



The blue line in each plot representes the decomposition. There is a legend in the upper left corner of each plot to let you know what each plot represents.

You can see the decomposition is not perfect with regards to seasonality and residuals, but it's pretty close!

NOTE: You may notice both *trend* and *residuals* are missing data towards the beginning and end. This has to do with how trend is calculated (beyond the scope of this lesson). The residuals are missing simply because $R_t = O_t - T_t - S_t$, so missing trend values mean missing residual values as well. In other words, there is nothing wrong with these graphs.

Multiplicative Decomposition

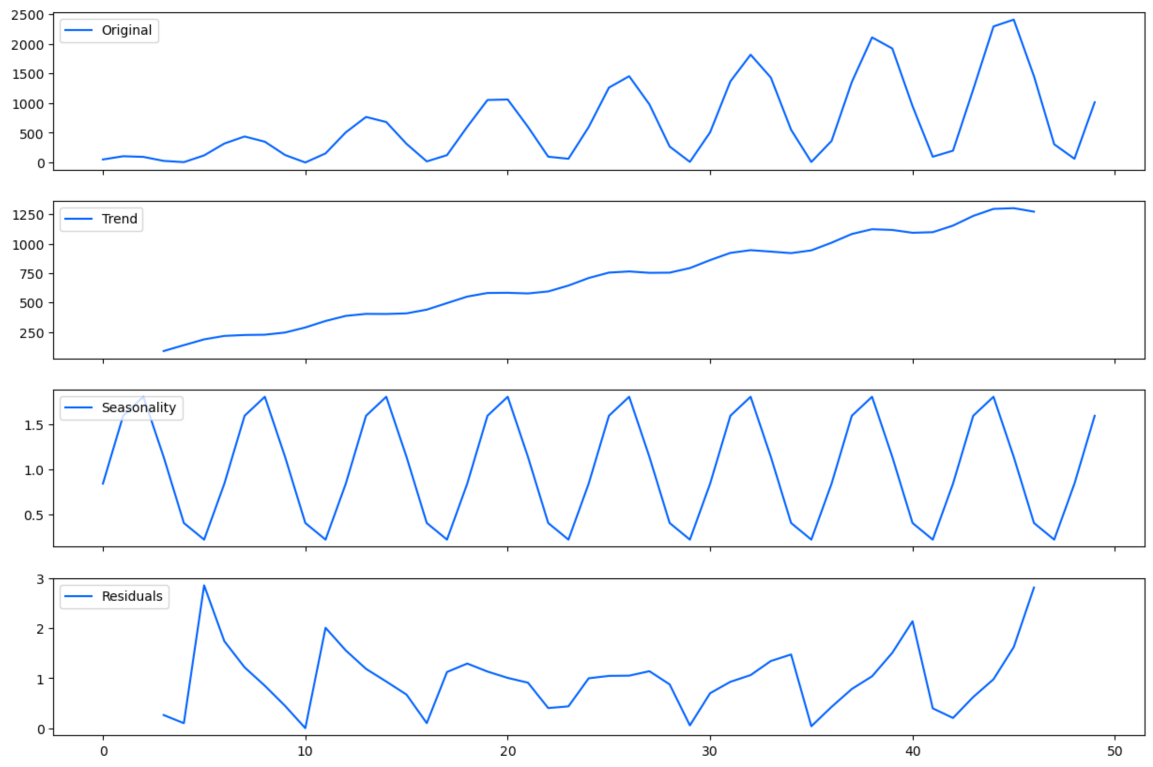
In [27]:

```
ss_decomposition = seasonal_decompose(x=multiplicative, model='multiplicative', period=6)
estimated_trend = ss_decomposition.trend
estimated_seasonal = ss_decomposition.seasonal
estimated_residual = ss_decomposition.resid
```

Multiplicative Plots

In [28]:

```
fig, axes = plt.subplots(4, 1, sharex=True, sharey=False)
fig.set_figheight(10)
fig.set_figwidth(15)

axes[0].plot(multiplicative, label='Original')
axes[0].legend(loc='upper left');

axes[1].plot(estimated_trend, label='Trend')
axes[1].legend(loc='upper left');

axes[2].plot(estimated_seasonal, label='Seasonality')
axes[2].legend(loc='upper left');

axes[3].plot(estimated_residual, label='Residuals')
axes[3].legend(loc='upper left');
```



Again, the decomposition does a relatively good job picking up the overall trend and seasonality. We can see the shapes follow the patterns we expect.

Exercise #2

In this exercise, your task is decompose **Dataset_A** and **Dataset_B**. You should first create a decomposition model in Python. Then you should plot the original series, the trend, seasonality, and residuals, in that order.

Decomposition Models

In [32]:

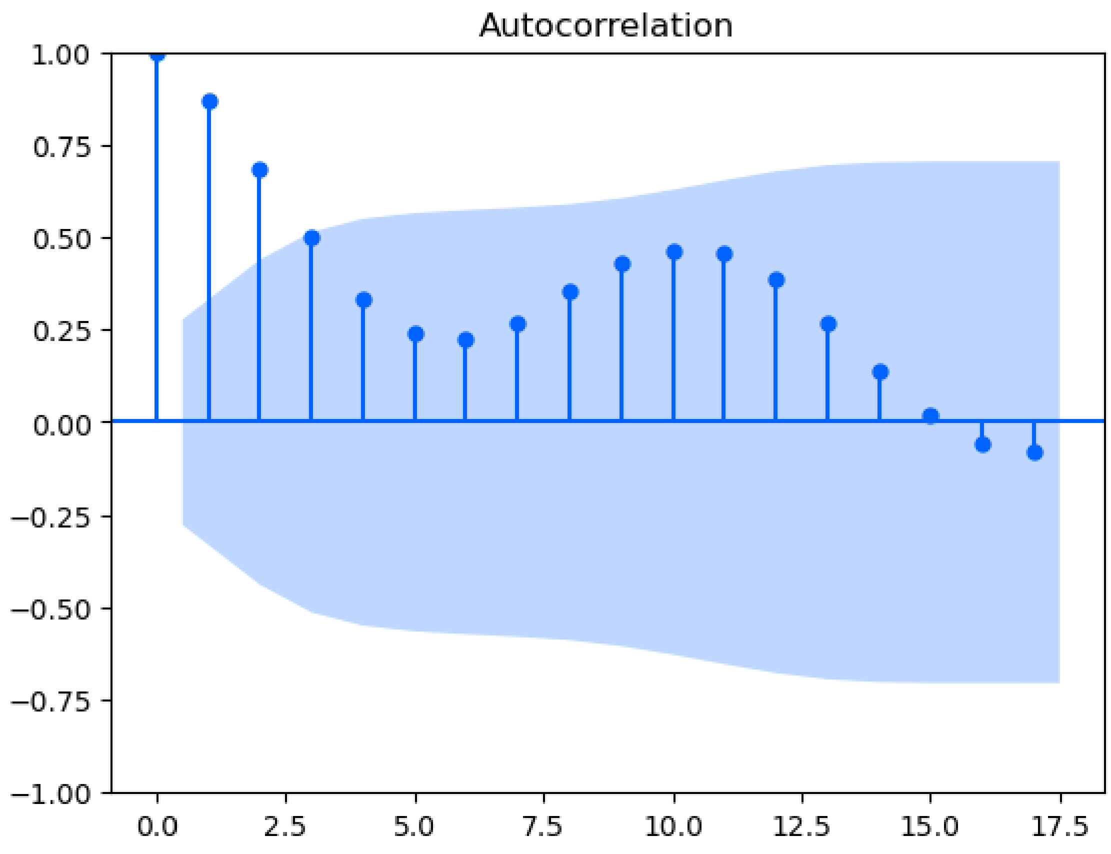
```
from statsmodels.tsa.seasonal import seasonal_decompose

### BEGIN SOLUTION
ss_decomposition_multi = seasonal_decompose(x=dataset_A,
                                            model='multiplicative',
                                            period=11)

estimated_trend_multi = ss_decomposition_multi.trend
estimated_seasonal_multi = ss_decomposition_multi.seasonal
estimated_residual_multi = ss_decomposition_multi.resid


ss_decomposition_add = seasonal_decompose(x=dataset_B,
                                          model='additive',
                                          period=3)

estimated_trend_add = ss_decomposition_add.trend
estimated_seasonal_add = ss_decomposition_add.seasonal
estimated_residual_add = ss_decomposition_add.resid
```

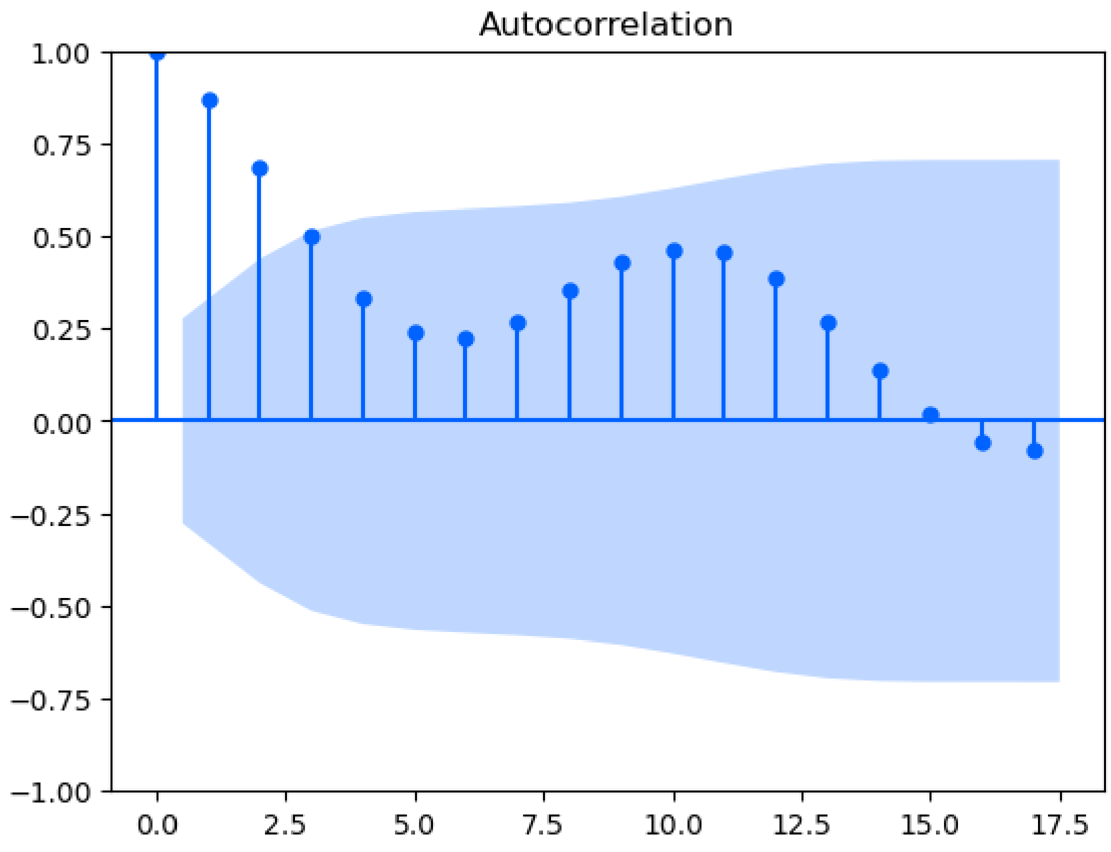
Dataset_A Plot

In [33]:

```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(dataset_A)
```

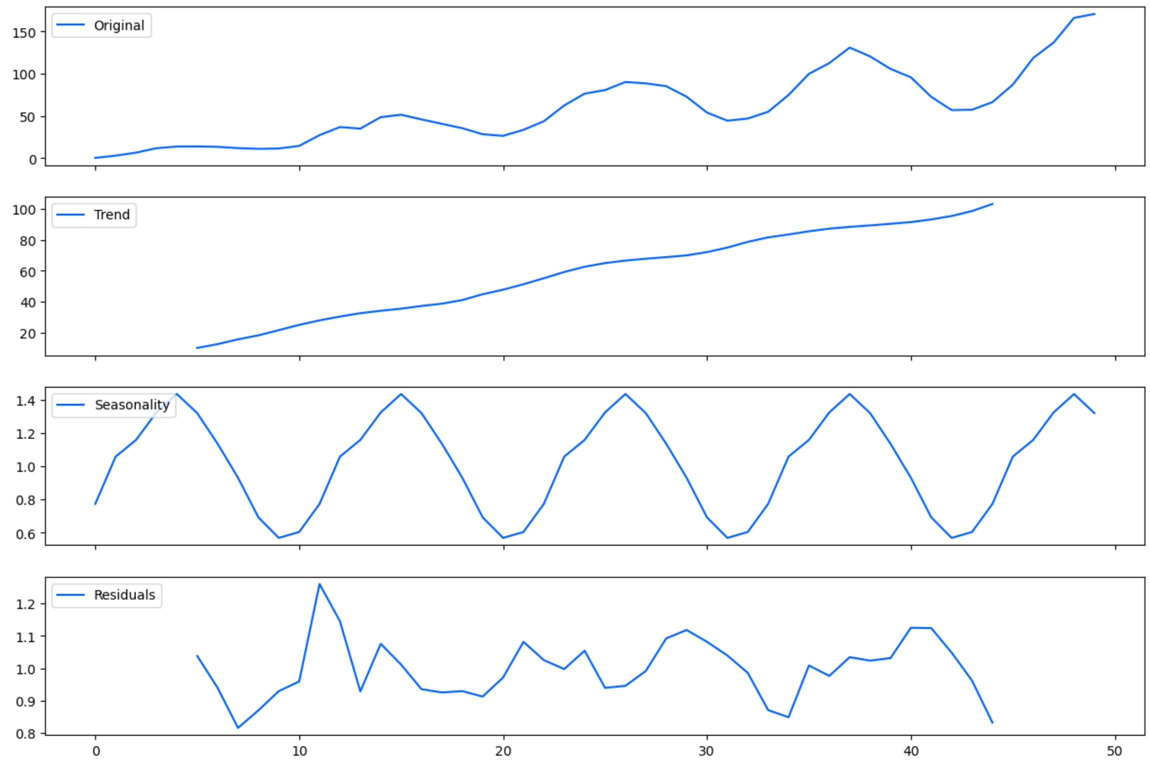
Out[33]:





In [34]:

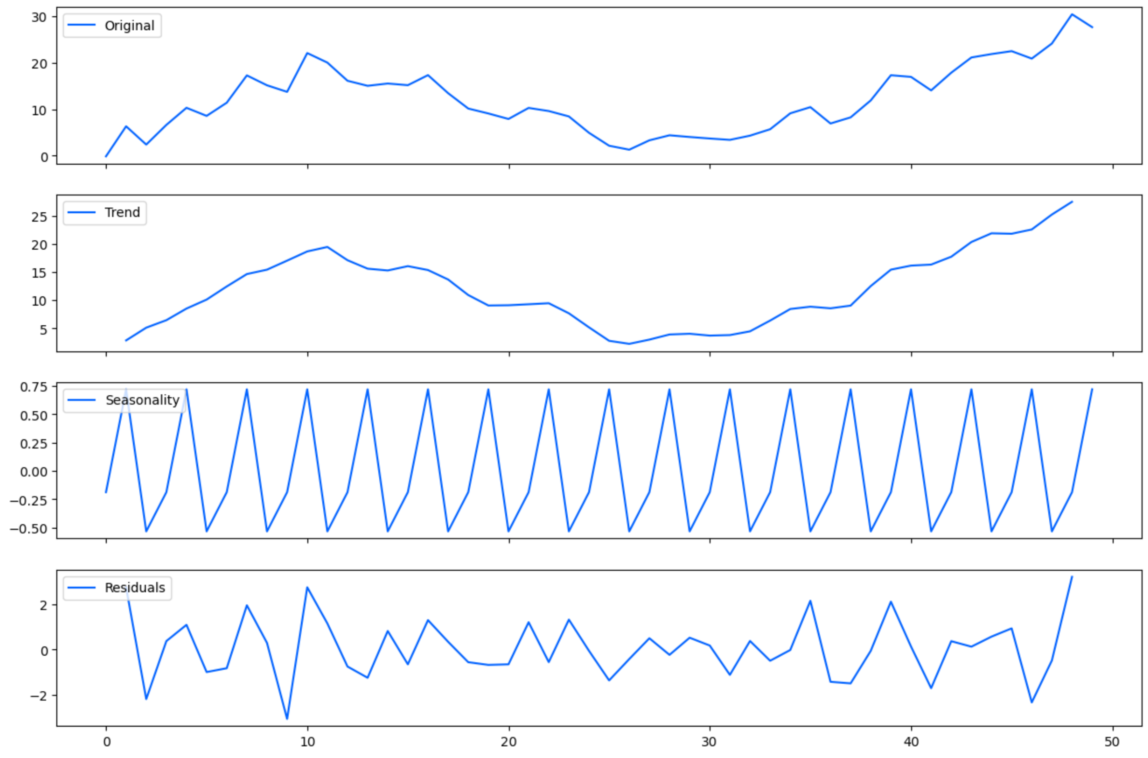
```
fig, axes = plt.subplots(4, 1, sharex=True, sharey=False)
fig.set_figheight(10)
fig.set_figwidth(15)

axes[0].plot(dataset_A, label='Original')
axes[0].legend(loc='upper left');

axes[1].plot(estimated_trend_multi, label='Trend')
axes[1].legend(loc='upper left');

axes[2].plot(estimated_seasonal_multi, label='Seasonality')
axes[2].legend(loc='upper left');

axes[3].plot(estimated_residual_multi, label='Residuals')
axes[3].legend(loc='upper left');
```



Dataset_B Plot

In [35]:

```
fig, axes = plt.subplots(4, 1, sharex=True, sharey=False)
fig.set_figheight(10)
fig.set_figwidth(15)

axes[0].plot(dataset_B, label='Original')
axes[0].legend(loc='upper left');

axes[1].plot(estimated_trend_add, label='Trend')
axes[1].legend(loc='upper left');

axes[2].plot(estimated_seasonal_add, label='Seasonality')
axes[2].legend(loc='upper left');

axes[3].plot(estimated_residual_add, label='Residuals')
axes[3].legend(loc='upper left');
### END BEGIN SOLUTION
```



Summary

In this lesson you should have gained:

1. A practical understanding of the three components of time series data
2. A better understanding of additive and multiplicative models
3. A simple way to decompose time series data

Machine Learning Foundation (C) 2020 IBM Corporation