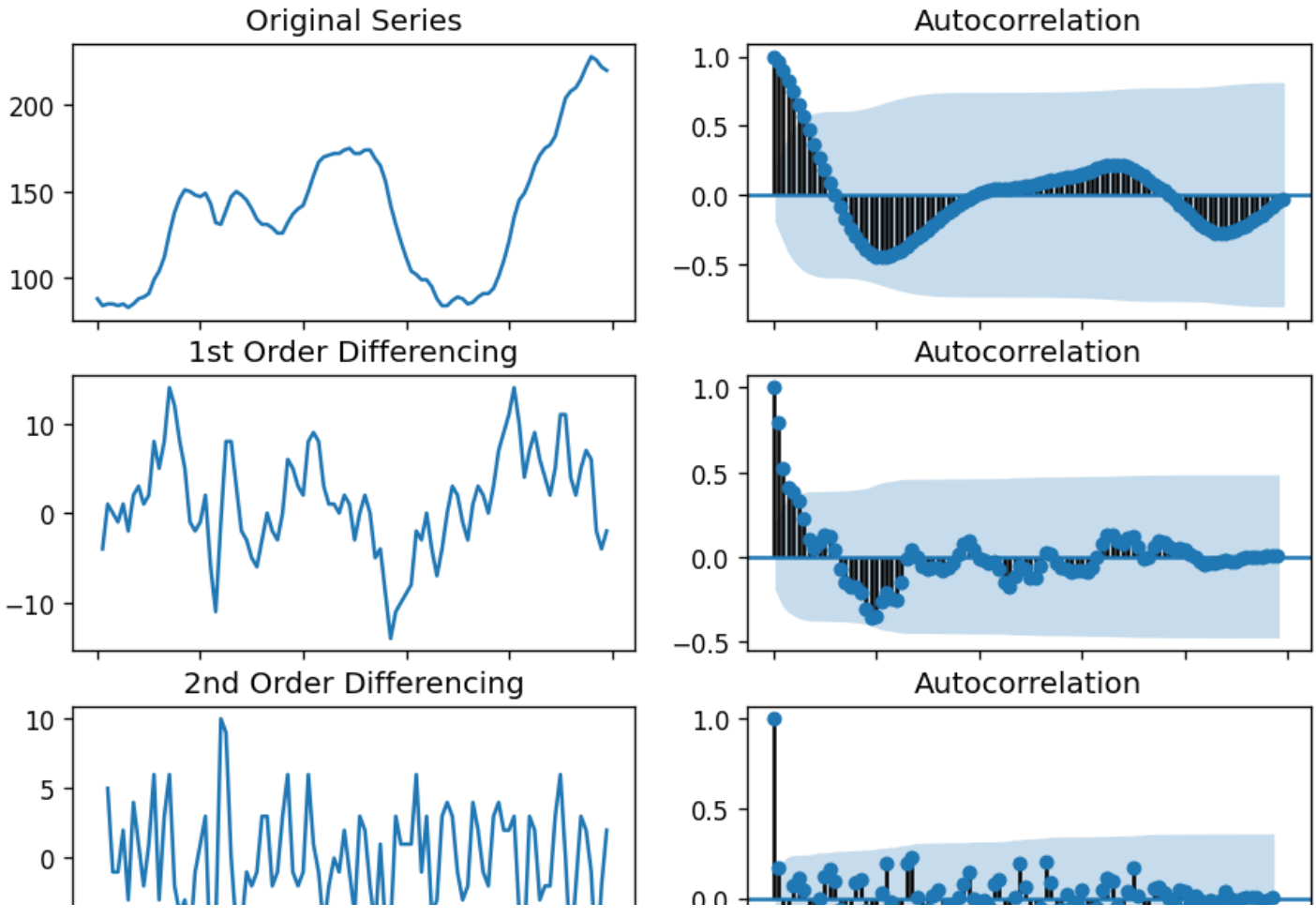
Arima model

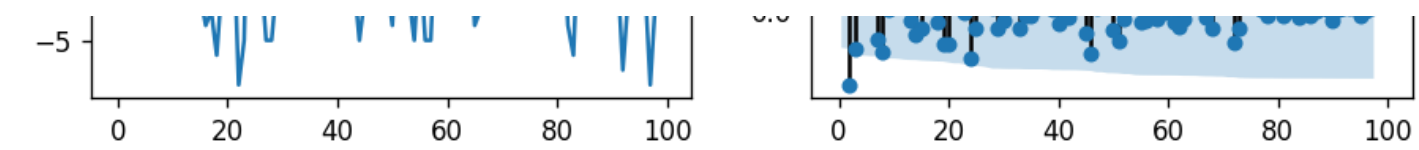
$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

In [ ]:

```
import numpy as np, pandas as pd
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.figsize':(9,7), 'figure.dpi':120})

# Import data
df = pd.read_csv('/content/drive/My Drive/SyncPC/Data Analytic/slide/TimeSeries/wwwusage.
csv', names=['value'], header=0)

# Original Series
fig, axes = plt.subplots(3, 2, sharex=True)
axes[0, 0].plot(df.value); axes[0, 0].set_title('Original Series')
plot_acf(df.value, ax=axes[0, 1],lags=99)

# 1st Differencing
axes[1, 0].plot(df.value.diff()); axes[1, 0].set_title('1st Order Differencing')
plot_acf(df.value.diff().dropna(), ax=axes[1, 1],lags=98)

# 2nd Differencing
axes[2, 0].plot(df.value.diff().diff()); axes[2, 0].set_title('2nd Order Differencing')
plot_acf(df.value.diff().diff().dropna(), ax=axes[2, 1],lags=97)

plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: p
andas.util.testing is deprecated. Use the functions in the public API at pandas.testing i
nstead.
  import pandas.util.testing as tm
```



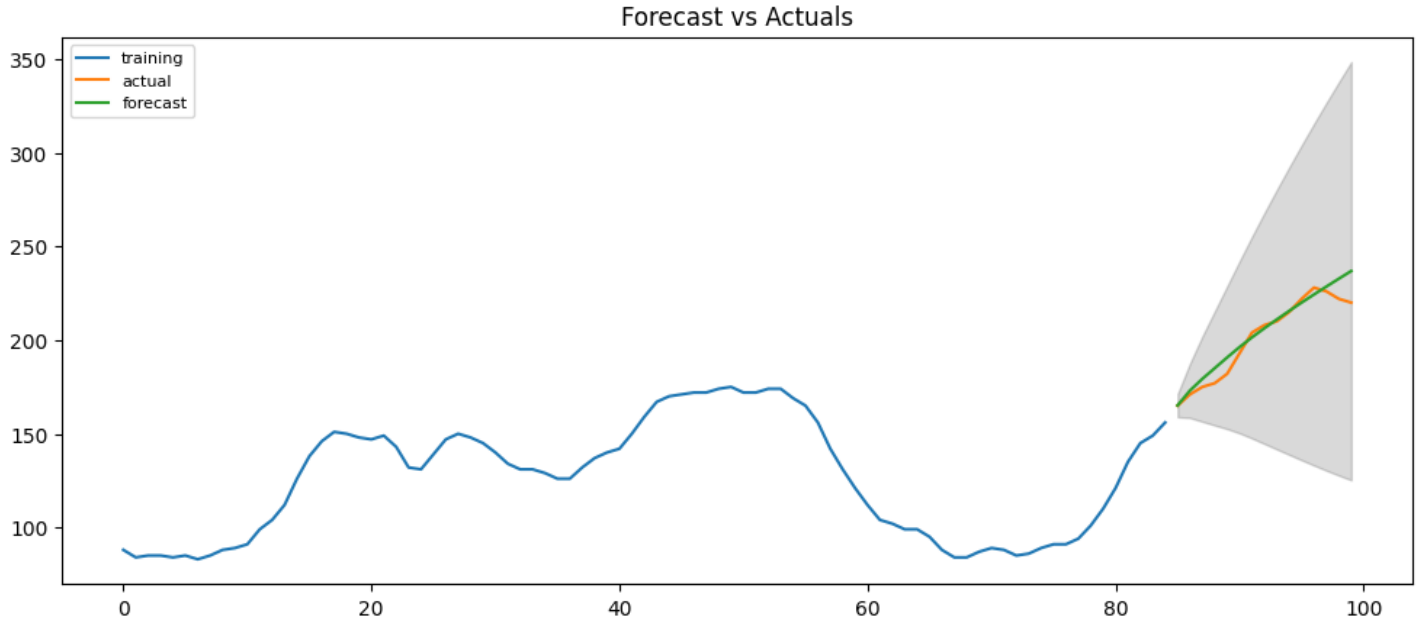


In [ ]:

```
from statsmodels.tsa.stattools import acf

# Create Training and Test
train = df.value[:85]
test = df.value[85:]
```

In [ ]:

```
from statsmodels.tsa.arima_model import ARIMA
# Build Model
#model = ARIMA(train, order=(3,2,1))
model = ARIMA(train, order=(3, 2,1))
fitted = model.fit(disp=-1)

# Forecast
fc, se, conf = fitted.forecast(15, alpha=0.05)  # 95% conf

# Make as pandas series
fc_series = pd.Series(fc, index=test.index)
lower_series = pd.Series(conf[:, 0], index=test.index)
upper_series = pd.Series(conf[:, 1], index=test.index)

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index, lower_series, upper_series,
                 color='k', alpha=.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```



In [ ]:

```
# Accuracy metrics
def forecast_accuracy(forecast, actual):
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))  # MAPE
    me = np.mean(forecast - actual)             # ME
    mae = np.mean(np.abs(forecast - actual))    # MAE
    mpe = np.mean((forecast - actual)/actual)   # MPE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE
```

```
    corr = np.corrcoef(forecast, actual)[0,1]   # corr
    mins = np.amin(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    maxs = np.amax(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    minmax = 1 - np.mean(mins/maxs)             # minmax
    acf1 = acf(fc-test)[1]                      # ACF1
    return({'mape':mape, 'me':me, 'mae': mae,
            'mpe': mpe, 'rmse':rmse, 'acf1':acf1,
            'corr':corr, 'minmax':minmax})

forecast_accuracy(fc, test.values)
```

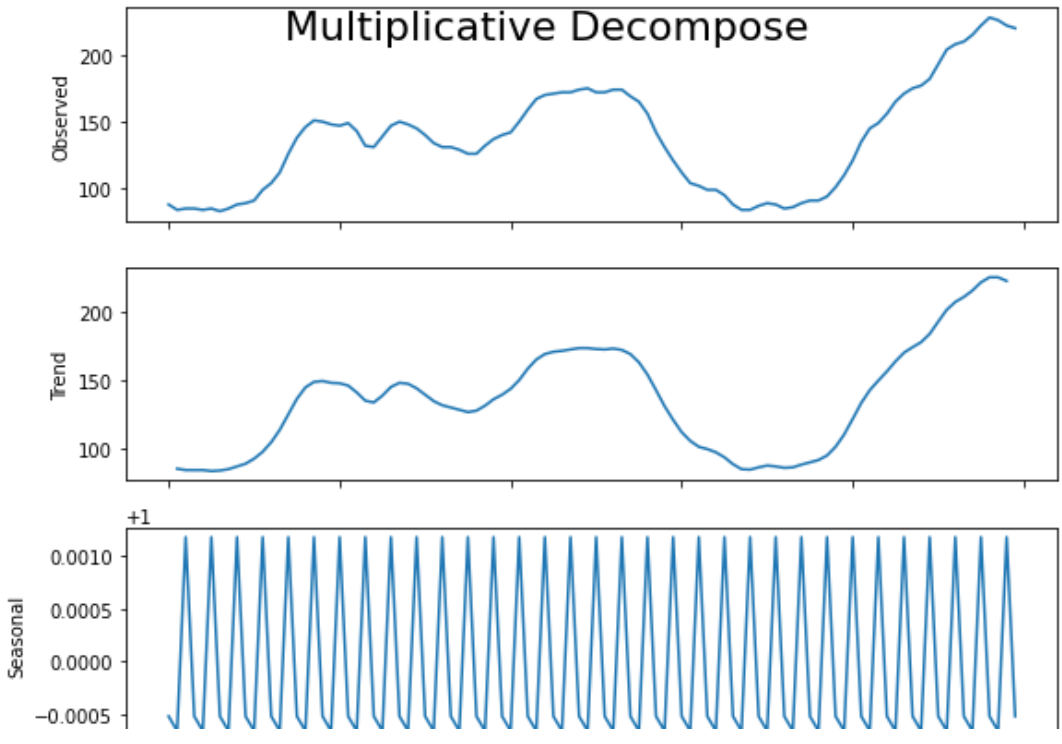
```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/stattools.py:541: FutureWarning: f
ft=True will become the default in a future version of statsmodels. To suppress this warn
ing, explicitly set fft=False.
  warnings.warn(msg, FutureWarning)
```

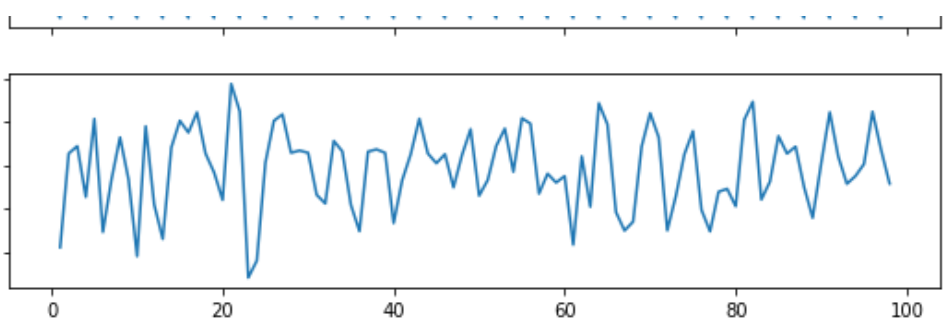
Out[ ]:

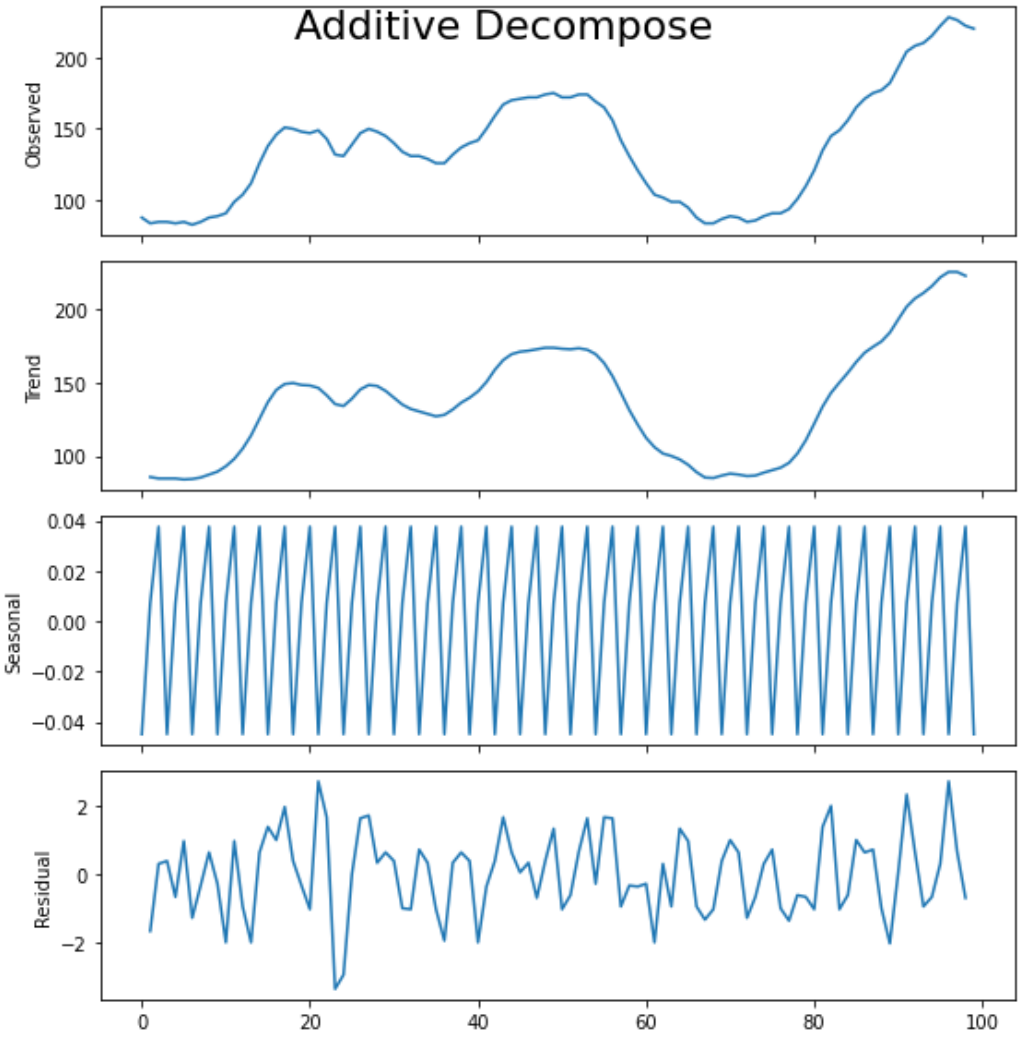
```
{'acf1': 0.51054935065353,
 'corr': 0.9674576270862356,
 'mae': 4.548347730567881,
 'mape': 0.022501437547584012,
 'me': 3.230854890946811,
 'minmax': 0.02163165748301432,
 'mpe': 0.01642133821060209,
 'rmse': 6.373284525171391}
```

In [ ]:

```
from statsmodels.tsa.seasonal import seasonal_decompose

# Import data
df = pd.read_csv('/content/drive/My Drive/SyncPC/Data Analytic/slide/TimeSeries/wwwusage.
csv', names=['value'], header=0)

# Multiplicative Decomposition
result_mul = seasonal_decompose(df['value'], model='multiplicative',freq=3)

# Additive Decomposition
result_add = seasonal_decompose(df['value'], model='additive',freq=3)

# Plot
plt.rcParams.update({'figure.figsize': (8,8)})
result_mul.plot().suptitle('Multiplicative Decompose', fontsize=22)
result_add.plot().suptitle('Additive Decompose', fontsize=22)
plt.show()
```




Residual
1.02
1.01
1.00
0.99
0.98
0
20
40
60
80
100


Additive Decompose
Observed
200
150
100
Trend
200
150
100
Seasonal
0.04
0.02
0.00
−0.02
−0.04
Residual
2
0
−2
0
20
40
60
80
100