Winter Semester 2024 –2025
PMDTS 504L – Regression Analysis and Predictive Modelling

Objective: Check the stationarity of time series model:

Here we illustrate the key differences between stationary and non-stationary time series using two real-world datasets. The Daily Female Births dataset represents a stationary series, where daily birth counts fluctuate around a constant mean without any visible trend or seasonal pattern. But, the Airline Passengers dataset is non-stationary, showing a clear upward trend and seasonal variations, indicating that its statistical properties change over time. By visualizing these series using Matplotlib, the script highlights how stationary data remains stable over time, while non-stationary data requires transformations like differencing or log scaling for meaningful analysis and forecasting.

Methods to check stationarity

Visual Inspection

- **Plot the Time Series:**
    - Look for trends, seasonality, or changing variance over time.
    - A stationary series will appear as a horizontal band around a constant mean.
- Rolling Statistics:
    - Rolling statistics involve calculating moving averages and moving standard deviations over a fixed window of past data points. This helps in analyzing trends and variability over time.
    - Why Use Rolling Statistics?
        - Helps in detecting trends in time series data.
        - Identifies seasonal patterns.
        - Used for checking stationarity (constant mean and variance).
        - Helps in smoothing out short-term fluctuations.
    - Plot rolling mean and rolling standard deviation. If they remain constant over time, the series may be stationary.
- **Summary Statistics**
    - Split the Series into Two or More Windows
    - Compare the mean and variance of different segments.
    - In a stationary series, these values should be similar across time segments.
- **Statistical Tests**

- **Augmented Dickey-Fuller (ADF) Test**

The Augmented Dickey-Fuller (ADF) Test is a statistical test used to check whether a given time series is stationary or not. Stationarity means that the properties of the series—like mean, variance, and autocorrelation—do not change over time, which is a crucial assumption in many time series forecasting models like ARIMA. The ADF test is an enhanced version of the basic Dickey-Fuller test. It adds lagged difference terms of the series to the regression equation to account for higher-order autocorrelation, hence the term "augmented."

In the ADF test, the null hypothesis (H_0) is that the time series has a unit root, which means it is non-stationary. The alternative hypothesis (H_1) is that the series is stationary. The test outputs an ADF statistic and a corresponding p-value. If the p-value is less than the chosen significance level (typically 0.05), we reject the null hypothesis and conclude that the series is stationary. The test also provides critical values at various confidence levels (1%, 5%, and 10%). If the ADF statistic is less than these critical values, it further supports the rejection of the null hypothesis. The ADF test helps confirm whether differencing or other transformations are needed to achieve stationarity before modeling.

    - **Hypotheses:**
        - **Null (H_0): The series is non-stationary.**
        - **Alternative (H_1): The series is stationary.**
- **Interpretation:**
    - **If the p-value is less than 0.05, reject $H_0 \Rightarrow$ the series is stationary**

Autocorrelation:

The Autocorrelation Function (ACF) measures the correlation between a time series and its lagged values at different time points. It helps us understand how past values influence future values.

How to Interpret ACF for Stationarity?

- Stationary Series:
    - ACF values drop to near zero quickly.
    - Only a few significant spikes (short-term dependence).
    - Suggests that the time series is stationary.
- **Non-Stationary Series:**
    - ACF values decline slowly (high values for many lags).
    - A sign of long-term correlation and possible trends.
    - Indicates non-stationarity (needs transformation like differencing).
- **Seasonality Present**
    - Repeating patterns or high autocorrelation at specific lags.
    - Suggests seasonal effects in the time series.

```python
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller, kpss
from statsmodels.graphics.tsaplots import plot_acf

# ------------------------------------------------------------------------------
# 1. Load the Time Series Dataset
# ------------------------------------------------------------------------------
# The dataset contains daily female births. We set the date as the index.
stationary_series = pd.read_csv('daily-total-female-births.csv', header=0, index_col=0)
stationary_series.index = pd.to_datetime(stationary_series.index)  # Convert index to datetime
series = stationary_series.squeeze()  # Convert DataFrame to Series if needed

# ------------------------------------------------------------------------------
# 2. Plot the Original Time Series
# ------------------------------------------------------------------------------
plt.figure(figsize=(10, 4))
plt.plot(series, label='Daily Female Births')
plt.title('Time Series Plot')
plt.xlabel('Date')
plt.ylabel('Births')
plt.grid(True)
plt.legend()
plt.show()
```

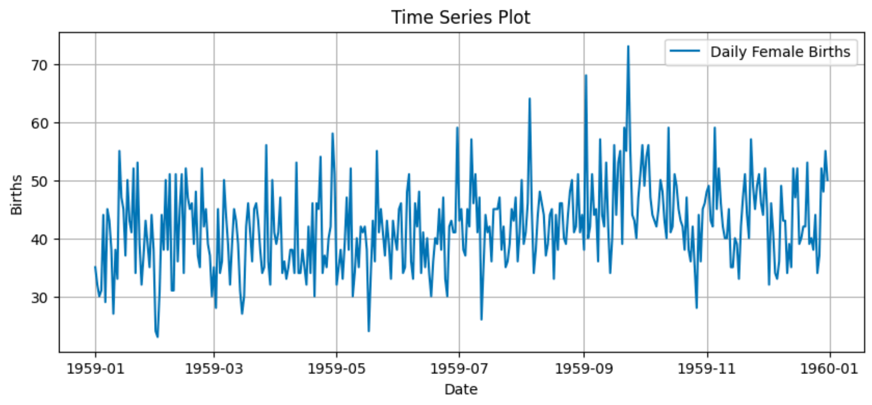
```python
# -------------------------------------------------------------------------------
# 3. Summary Statistics (Mean, Variance) for Entire Series and Subsets
# -------------------------------------------------------------------------------
# Split into two equal parts for comparison
split = len(series) // 2
first_half = series[:split]
second_half = series[split:]

print("Summary Statistics:")
print(f"Entire Series - Mean: {series.mean():.2f}, Variance: {series.var():.2f}")
print(f"First Half    - Mean: {first_half.mean():.2f}, Variance: {first_half.var():.2f}")
print(f"Second Half   - Mean: {second_half.mean():.2f}, Variance: {second_half.var():.2f}")

# If mean and variance are relatively constant, the series might be stationary.

# -------------------------------------------------------------------------------
# 4. Rolling Statistics (Mean and Standard Deviation)
# -------------------------------------------------------------------------------
rolling_mean = series.rolling(window=10).mean()
rolling_std = series.rolling(window=10).std()

plt.figure(figsize=(10, 4))
plt.plot(series, label='Original')
plt.plot(rolling_mean, label='Rolling Mean', color='red')
plt.plot(rolling_std, label='Rolling Std Dev', color='green')
plt.title('Rolling Mean and Standard Deviation (Window = 10)')
plt.legend()
plt.grid(True)
plt.show()

# Visual check: If rolling mean and std dev stay roughly constant, series is likely stationary.
```

```python
# -------------------------------------------------------------------------------
# 5. Augmented Dickey-Fuller (ADF) Test
# -------------------------------------------------------------------------------
# Null hypothesis: Series has a unit root (non-stationary)
# Perform Augmented Dickey-Fuller (ADF) Test
adf_result = adfuller(series)

# Print the results with explanations
print('\n--- Augmented Dickey-Fuller (ADF) Test Results ---\n')

# ADF Statistic: Measures stationarity, lower (more negative) values indicate stronger stationarity.
print(f'ADF Statistic : {adf_result[0]:.4f}')

# p-value: Determines statistical significance.
# If p-value < 0.05, reject the null hypothesis (i.e., the series is stationary).
print(f'p-value       : {adf_result[1]:.4f}')

# Number of Lags Used: Shows the number of lagged observations used in the test.
print(f'Number of Lags Used : {adf_result[2]}')

# Number of Observations Used: The sample size used for testing.
print(f'Number of Observations Used : {adf_result[3]}')

# Critical Values: Thresholds at different confidence levels (1%, 5%, 10%)
print('\nCritical Values for Stationarity:')
for key, value in adf_result[4].items():
    print(f'   {key}: {value:.4f}')
```

```
# Interpretation Based on p-value
if adf_result[1] < 0.05:
    print("The p-value is below 0.05. We reject the null hypothesis.")
    print(" Conclusion: The series is stationary.")
else:
    print(" Thep-value is above 0.05. We fail to reject the null hypothesis.")
    print(" Conclusion: The series is non-stationary.")

# If non-stationary, apply differencing
if adf_result[1] > 0.05:
    print("\nApplying first-order differencing to attempt stationarity...")
    differenced_series = series.diff().dropna()

    # Run ADF test again on differenced data
    adf_result_diff = adfuller(differenced_series)

    print("\n--- ADF Test After Differencing ---")
    print(f'ADF Statistic : {adf_result_diff[0]:.4f}')
    print(f'p-value       : {adf_result_diff[1]:.4f}')
    if adf_result_diff[1] < 0.05:
        print(" After differencing, the series is now stationary.")
    else:
        print("The series is still non-stationary. Consider additional transformations.")


# A small p-value (< 0.05) → reject null → data is stationary
```

```
# -----------------------------------------------------------------------------
# 6 : Autocorrelation Plot (ACF)
# -----------------------------------------------------------------------------
# Plot the Autocorrelation Function (ACF)
plot_acf(series, lags=30)  # Display up to 30 lags
plt.title('Autocorrelation Plot')  # Title of the plot
plt.grid(True)  # Enable grid for better visualization
plt.show()  # Display the plot
```

Results:

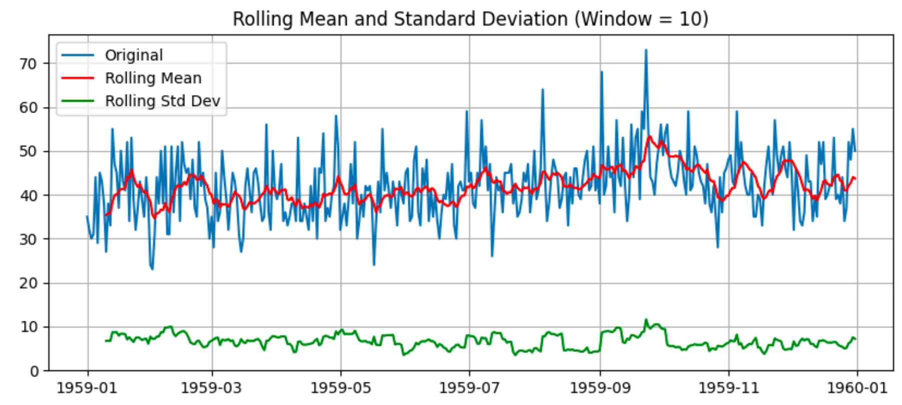


```
Summary Statistics:
Entire Series - Mean: 41.98, Variance: 54.00
First Half    - Mean: 39.76, Variance: 49.49
Second Half   - Mean: 44.19, Variance: 48.98
```

Interpretation:

The time series plot shows the daily female births over time, with fluctuations observed throughout the year 1959. The data does not exhibit a strong upward or downward trend, but there is visible variability, indicating potential stationarity. The presence of spikes suggests some periods with higher-than-average births, but overall, the series appears relatively stable.

The summary statistics compare the mean and variance of the entire series with two separate time windows. The entire series has a mean of 41.98 and a variance of 54.00, indicating that, on average, about 42 births occur per day, with some variation. When split into two halves, the first half has a slightly lower mean (39.76) and variance (49.49), while the second half shows a higher mean (44.19) and variance (48.98). The relatively small difference in these statistics suggests that the data is likely stationary, as the mean and variance remain fairly consistent over time.

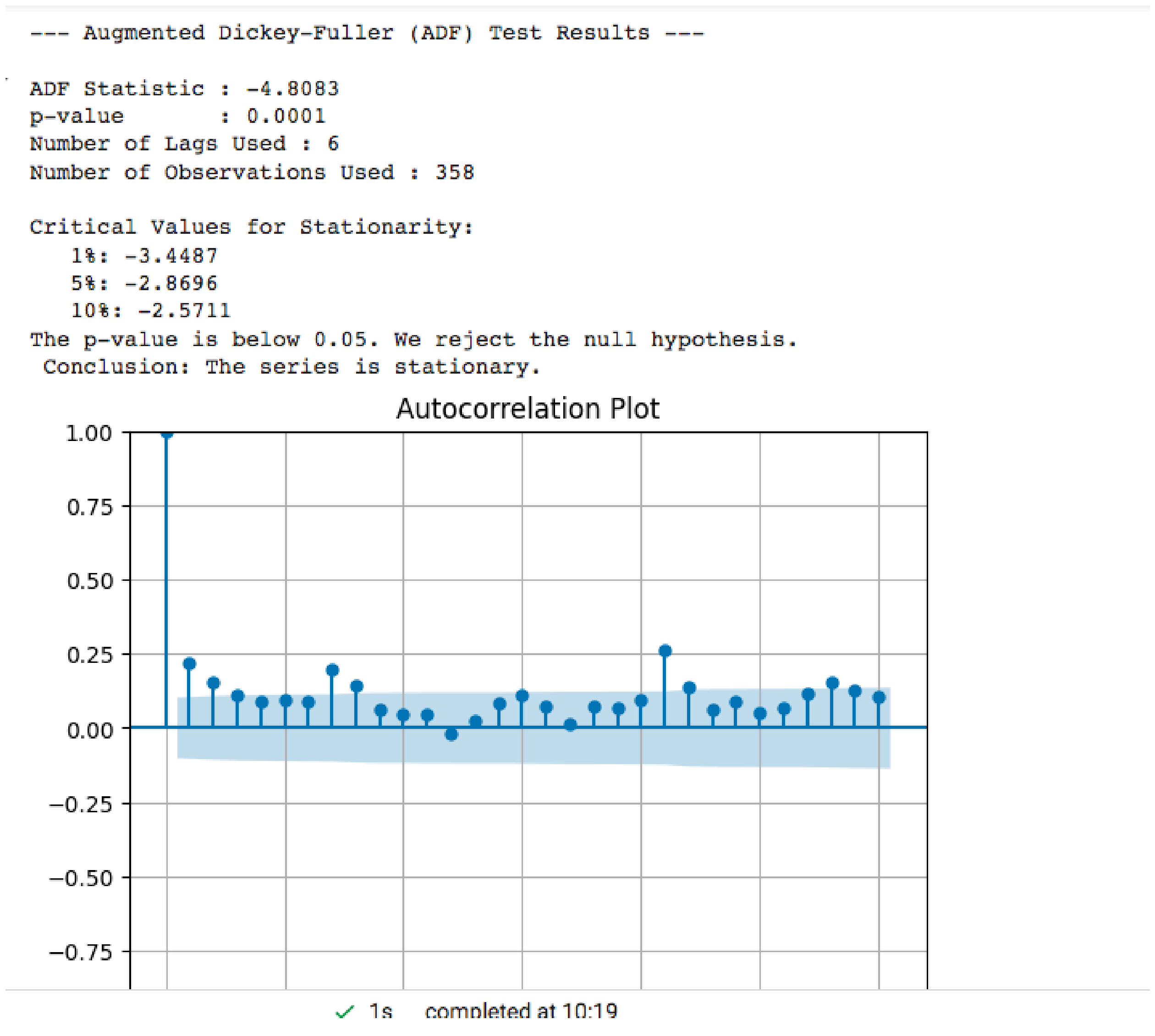


```
--- Augmented Dickey-Fuller (ADF) Test Results ---
```

Interpretation:

This plot visualizes the rolling mean (red line) and rolling standard deviation (green line) with a window size of 10, overlaid on the original time series (blue line). The rolling mean helps identify trends, while the rolling standard deviation indicates volatility in the data.

From the plot, the rolling mean appears relatively stable over time, with no significant upward or downward trend. This suggests that the average number of daily female births remains constant over time, an indicator of stationarity. The rolling standard deviation also shows relatively minor fluctuations, meaning that the variability in the data does not increase or decrease significantly over time.



```
--- Augmented Dickey-Fuller (ADF) Test Results ---

ADF Statistic : -4.8083
p-value       : 0.0001
Number of Lags Used : 6
Number of Observations Used : 358

Critical Values for Stationarity:
   1%: -3.4487
   5%: -2.8696
   10%: -2.5711
The p-value is below 0.05. We reject the null hypothesis.
 Conclusion: The series is stationary.
```

Interpretation of ADF Test and Autocorrelation Plot

Augmented Dickey-Fuller (ADF) Test Results:

- The ADF statistic is -4.8083, which is lower than all the critical values at the 1%, 5%, and 10% levels (-3.4487, -2.8696, and -2.5711, respectively).
- The p-value is 0.0001, which is well below 0.05. This means we reject the null hypothesis (which states that the series is non-stationary).

The time series is stationary, meaning its statistical properties (mean and variance) remain constant over time, making it suitable for time series modeling without differencing.

Autocorrelation Plot (ACF):

- The first lag has a high autocorrelation, which is expected as a time series is always correlated with itself at lag 0.
- The subsequent lags show a rapid decline and oscillate around zero, with most points falling within the blue confidence band. This indicates a weak autocorrelation structure, which is another sign of stationarity.

If the series were non-stationary, the ACF would exhibit a slow, gradual decline instead of a rapid drop.

Both the ADF test and ACF plot confirm that the given time series is stationary, making it suitable for further time series analysis and forecasting.

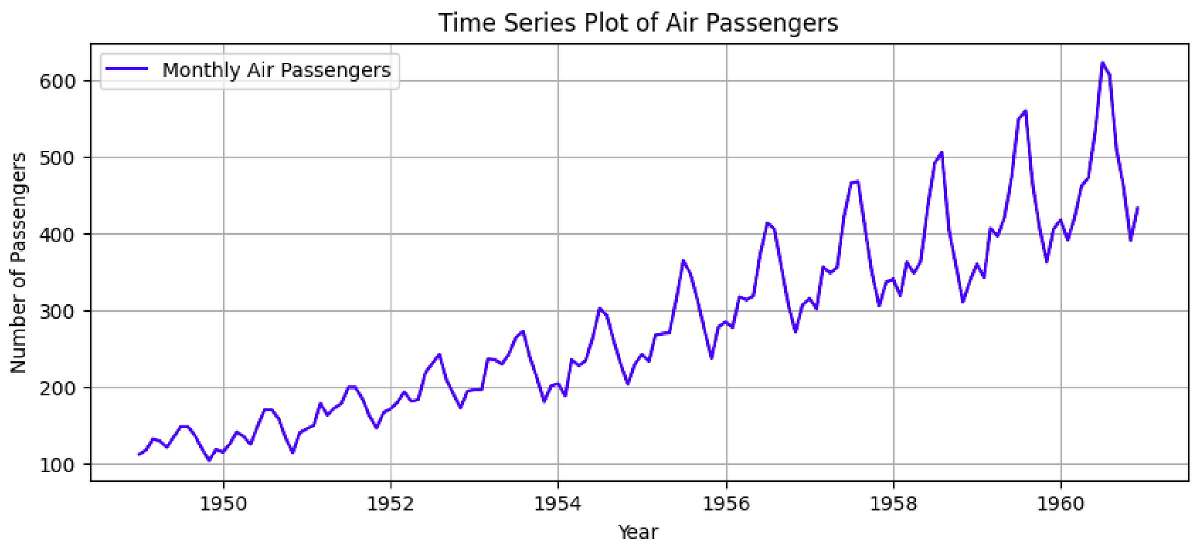
```python
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf

# ------------------------------------------------------------------------------
# 1. Load the Air Passengers Dataset
# ------------------------------------------------------------------------------
# The dataset contains monthly total airline passengers (in thousands)
# We load the dataset and set the 'Month' column as the index
air_passengers = pd.read_csv('AirPassengers.csv', header=0, index_col=0)

# Convert the index to datetime format for proper time series analysis
air_passengers.index = pd.to_datetime(air_passengers.index)

# Convert the DataFrame into a Series
series = air_passengers.squeeze()

# ------------------------------------------------------------------------------
# 2. Plot the Original Time Series
# ------------------------------------------------------------------------------
plt.figure(figsize=(10, 4))
plt.plot(series, label='Monthly Air Passengers', color='blue')
plt.title('Time Series Plot of Air Passengers')
plt.xlabel('Year')
plt.ylabel('Number of Passengers')
plt.grid(True)
plt.legend()
plt.show()
```

```python
# 3. Summary Statistics (Mean, Variance) for Entire Series and Subsets
# -----------------------------------------------------------------------------
# Split the data into two equal halves to compare statistics
split = len(series) // 2
first_half = series[:split]
second_half = series[split:]

# Display summary statistics for mean and variance
print("Summary Statistics:")
print(f"Entire Series - Mean: {series.mean():.2f}, Variance: {series.var():.2f}")
print(f"First Half    - Mean: {first_half.mean():.2f}, Variance: {first_half.var():.2f}")
print(f"Second Half   - Mean: {second_half.mean():.2f}, Variance: {second_half.var():.2f}")

# If the mean and variance are increasing over time, this indicates non-stationarity.

# -----------------------------------------------------------------------------
# 4. Rolling Statistics (Mean and Standard Deviation)
# -----------------------------------------------------------------------------
# Compute rolling mean and rolling standard deviation using a window size of 12 (1 year)
rolling_mean = series.rolling(window=12).mean()
rolling_std = series.rolling(window=12).std()

# Plot rolling statistics
plt.figure(figsize=(10, 4))
plt.plot(series, label='Original Series', color='blue')
plt.plot(rolling_mean, label='Rolling Mean (12 months)', color='red')
plt.plot(rolling_std, label='Rolling Std Dev (12 months)', color='green')
plt.title('Rolling Mean and Standard Deviation')
plt.legend()
plt.grid(True)
plt.show()
```

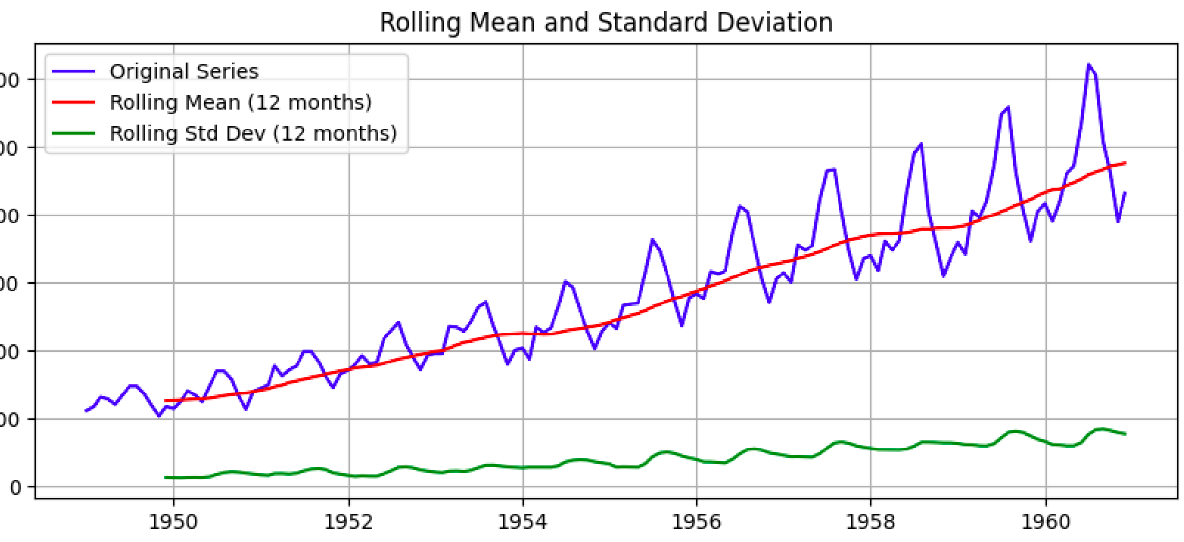
```python
# 5. Augmented Dickey-Fuller (ADF) Test
# -----------------------------------------------------------------------------
# The null hypothesis of the ADF test states that the series is non-stationary.
adf_result = adfuller(series)

# Print ADF test results
print('\n--- Augmented Dickey-Fuller (ADF) Test Results ---\n')

# ADF Statistic: A more negative value indicates stronger stationarity.
print(f'ADF Statistic : {adf_result[0]:.4f}')

# p-value: If p-value < 0.05, reject the null hypothesis (series is stationary).
print(f'p-value       : {adf_result[1]:.4f}')

# Number of lags used in the test
print(f'Number of Lags Used : {adf_result[2]}')

# Number of observations used in the test
print(f'Number of Observations Used : {adf_result[3]}')
```

```python
# Critical Values at 1%, 5%, and 10% significance levels
print('\nCritical Values for Stationarity:')
for key, value in adf_result[4].items():
    print(f'   {key}: {value:.4f}')

# Interpretation based on p-value
if adf_result[1] < 0.05:
    print("The p-value is below 0.05. We reject the null hypothesis.")
    print("Conclusion: The series is stationary.")
else:
    print("The p-value is above 0.05. We fail to reject the null hypothesis.")
    print("Conclusion: The series is non-stationary.")

# If the series is non-stationary, we apply differencing
if adf_result[1] > 0.05:
    print("\nApplying first-order differencing to attempt stationarity...")
    differenced_series = series.diff().dropna()

    # Perform ADF test again on the differenced series
    adf_result_diff = adfuller(differenced_series)

    print("\n--- ADF Test After Differencing ---")
    print(f'ADF Statistic : {adf_result_diff[0]:.4f}')
    print(f'p-value       : {adf_result_diff[1]:.4f}')

    # Check if stationarity is achieved after differencing
    if adf_result_diff[1] < 0.05:
        print("After differencing, the series is now stationary.")
    else:
        print("The series is still non-stationary. Consider additional transformations (e.g., log transformation).")
```

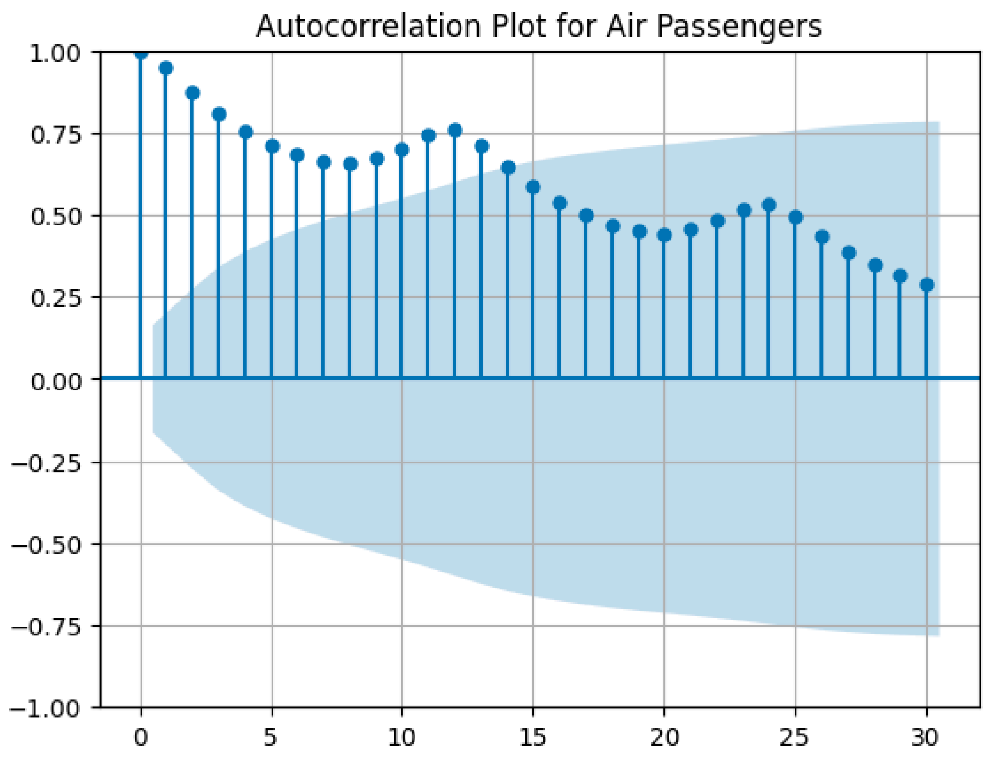
```python
# ---------------------------------------------------------------------------------
# 6. Autocorrelation Plot (ACF)
# ---------------------------------------------------------------------------------
# Plot the autocorrelation function to check for patterns
plot_acf(series, lags=30)  # Show autocorrelations up to 30 lags
plt.title('Autocorrelation Plot for Air Passengers')
plt.grid(True)
plt.show()

# If the autocorrelations decrease slowly, the series is non-stationary.
```



```
Summary Statistics:
Entire Series - Mean: 280.30, Variance: 14391.92
First Half    - Mean: 182.90, Variance: 2275.69
Second Half   - Mean: 377.69, Variance: 7471.74
```



```
--- Augmented Dickey-Fuller (ADF) Test Results ---

ADF Statistic : 0.8154
p-value       : 0.9919
Number of Lags Used : 13
Number of Observations Used : 130

Critical Values for Stationarity:
   1%: -3.4817
   5%: -2.8840
   10%: -2.5788
The p-value is above 0.05. We fail to reject the null hypothesis.
Conclusion: The series is non-stationary.

Applying first-order differencing to attempt stationarity...

--- ADF Test After Differencing ---
ADF Statistic : -2.8293
p-value       : 0.0542
The series is still non-stationary. Consider additional transformations (e.g., log transformation).
```



- **Strong Positive Autocorrelation**:
    - The first few lags (up to lag ~15) show significant positive autocorrelation.
    - This indicates that past values strongly influence future values, suggesting non-stationarity.

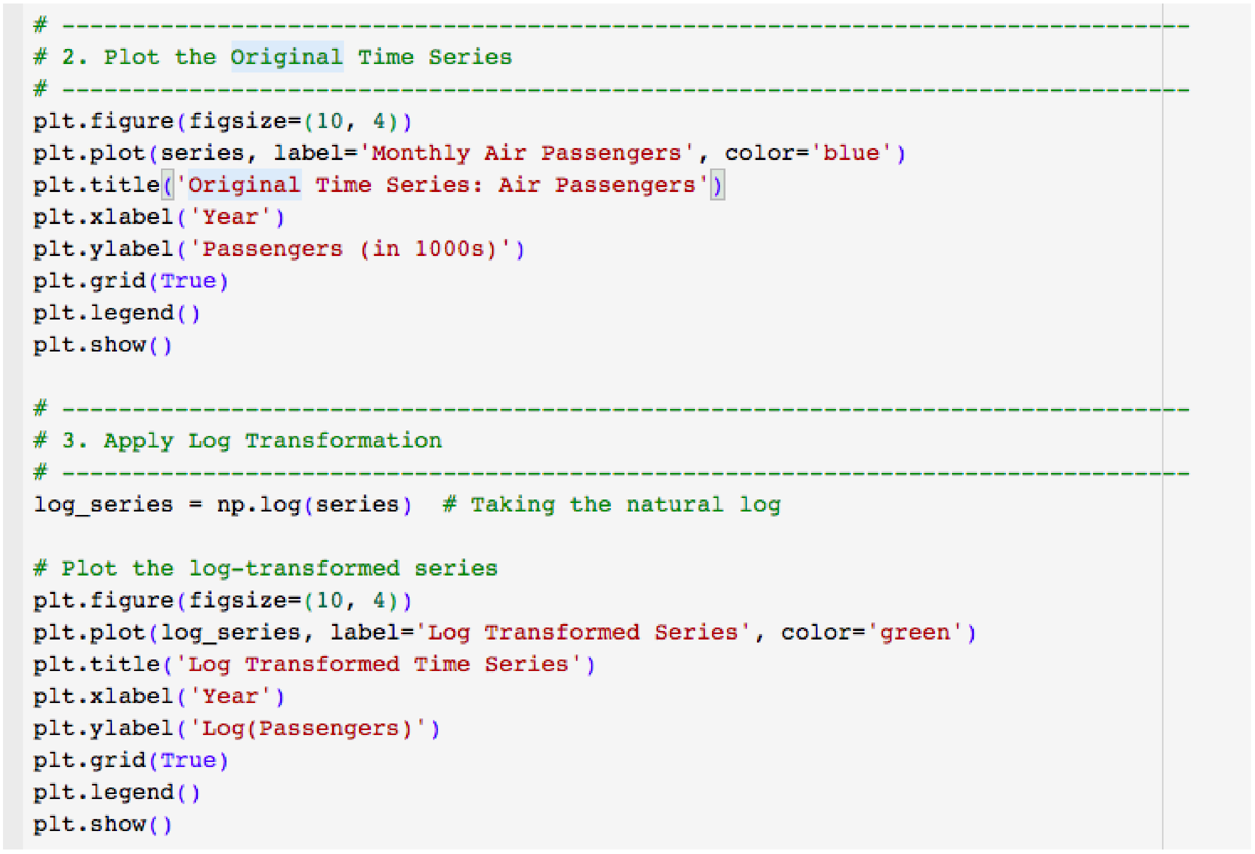
- **Slow Decay of ACF Values**:
    - The autocorrelation decreases gradually instead of dropping quickly to zero.

    - This slow decay suggests trend or seasonality, meaning the series is likely not stationary.

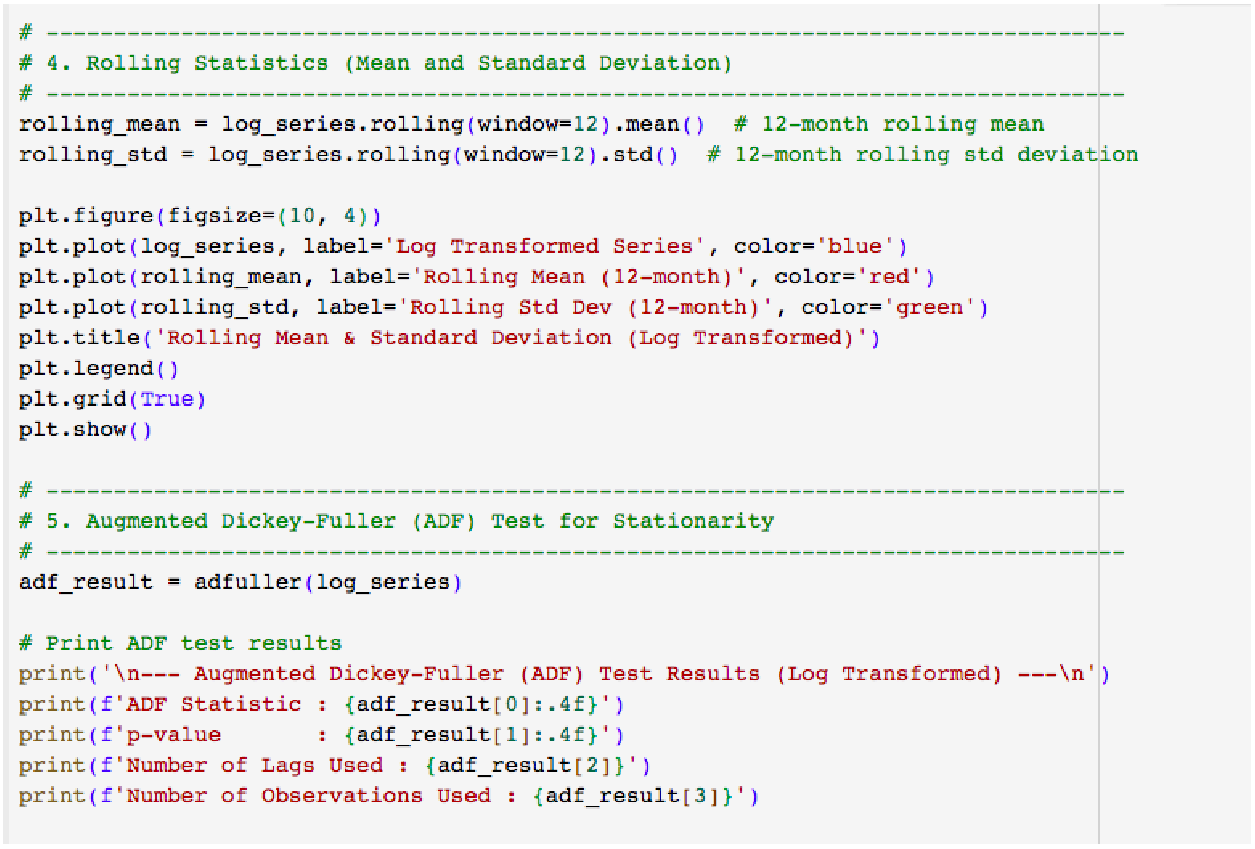
- **Seasonal Pattern (Possible Lag at 12)**:
    - There may be a noticeable pattern every 12 lags, which is typical for seasonal data.
    - This is common in datasets like monthly air passenger data, where demand varies seasonally.
- **Need for Differencing or Transformation**:
    - Since the ACF values are high and decay slowly, applying first-order differencing (to remove trend) and possibly seasonal differencing (lag 12) may be needed to make the series stationary.

Home Work:

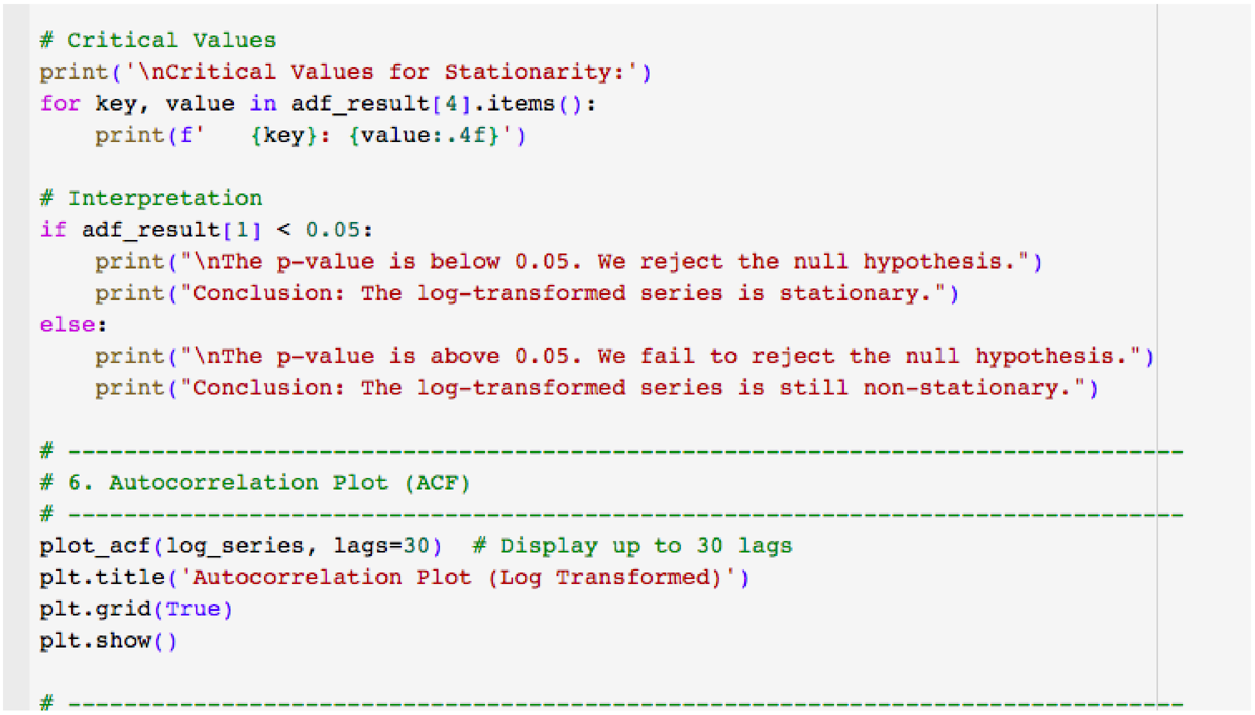
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf

# ---------------------------------------------------------------------------------
# 1. Load the Air Passengers Dataset
# ---------------------------------------------------------------------------------
# Dataset contains monthly total airline passengers (in thousands)
air_passengers = pd.read_csv('AirPassengers.csv', header=0, index_col=0)

# Convert index to datetime format
air_passengers.index = pd.to_datetime(air_passengers.index)

# Convert DataFrame to Series if needed
series = air_passengers.squeeze()
```



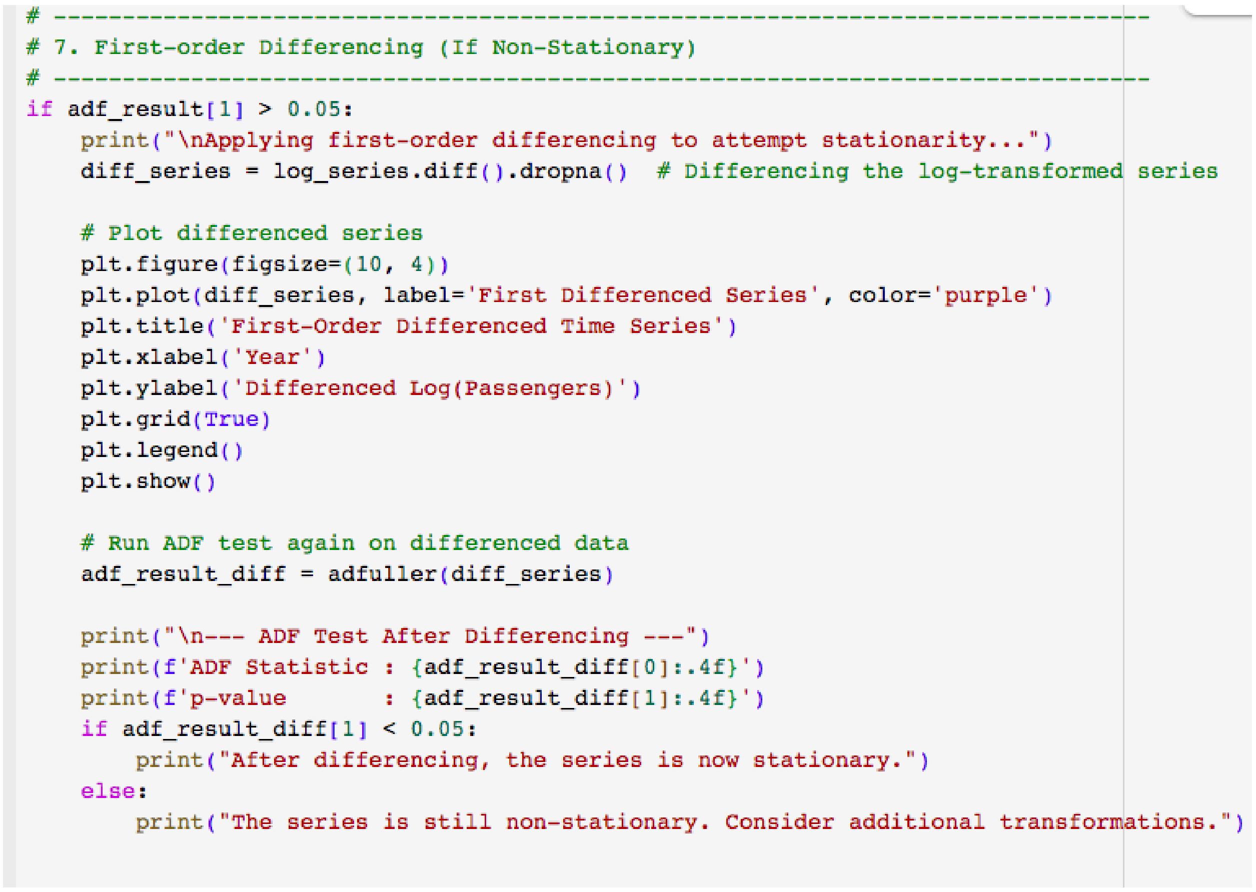
```python
# --------------------------------------------------------------------------------
# 2. Plot the Original Time Series
# --------------------------------------------------------------------------------
plt.figure(figsize=(10, 4))
plt.plot(series, label='Monthly Air Passengers', color='blue')
plt.title('Original Time Series: Air Passengers')
plt.xlabel('Year')
plt.ylabel('Passengers (in 1000s)')
plt.grid(True)
plt.legend()
plt.show()

# --------------------------------------------------------------------------------
# 3. Apply Log Transformation
# --------------------------------------------------------------------------------
log_series = np.log(series)  # Taking the natural log

# Plot the log-transformed series
plt.figure(figsize=(10, 4))
plt.plot(log_series, label='Log Transformed Series', color='green')
plt.title('Log Transformed Time Series')
plt.xlabel('Year')
plt.ylabel('Log(Passengers)')
plt.grid(True)
plt.legend()
plt.show()
```



```python
# --------------------------------------------------------------------------------
# 4. Rolling Statistics (Mean and Standard Deviation)
# --------------------------------------------------------------------------------
rolling_mean = log_series.rolling(window=12).mean()  # 12-month rolling mean
rolling_std = log_series.rolling(window=12).std()  # 12-month rolling std deviation

plt.figure(figsize=(10, 4))
plt.plot(log_series, label='Log Transformed Series', color='blue')
plt.plot(rolling_mean, label='Rolling Mean (12-month)', color='red')
plt.plot(rolling_std, label='Rolling Std Dev (12-month)', color='green')
plt.title('Rolling Mean & Standard Deviation (Log Transformed)')
plt.legend()
plt.grid(True)
plt.show()

# --------------------------------------------------------------------------------
# 5. Augmented Dickey-Fuller (ADF) Test for Stationarity
# --------------------------------------------------------------------------------
adf_result = adfuller(log_series)

# Print ADF test results
print('\n--- Augmented Dickey-Fuller (ADF) Test Results (Log Transformed) ---\n')
print(f'ADF Statistic : {adf_result[0]:.4f}')
print(f'p-value       : {adf_result[1]:.4f}')
print(f'Number of Lags Used : {adf_result[2]}')
print(f'Number of Observations Used : {adf_result[3]}')
```



```python
# Critical Values
print('\nCritical Values for Stationarity:')
for key, value in adf_result[4].items():
    print(f'   {key}: {value:.4f}')

# Interpretation
if adf_result[1] < 0.05:
    print("\nThe p-value is below 0.05. We reject the null hypothesis.")
    print("Conclusion: The log-transformed series is stationary.")
else:
    print("\nThe p-value is above 0.05. We fail to reject the null hypothesis.")
    print("Conclusion: The log-transformed series is still non-stationary.")

# ---------------------------------------------------------------------------------
# 6. Autocorrelation Plot (ACF)
# ---------------------------------------------------------------------------------
plot_acf(log_series, lags=30)  # Display up to 30 lags
plt.title('Autocorrelation Plot (Log Transformed)')
plt.grid(True)
plt.show()

# ---------------------------------------------------------------------------------
```



```python
# ---------------------------------------------------------------------------------
# 7. First-order Differencing (If Non-Stationary)
# ---------------------------------------------------------------------------------
if adf_result[1] > 0.05:
    print("\nApplying first-order differencing to attempt stationarity...")
    diff_series = log_series.diff().dropna()  # Differencing the log-transformed series

    # Plot differenced series
    plt.figure(figsize=(10, 4))
    plt.plot(diff_series, label='First Differenced Series', color='purple')
    plt.title('First-Order Differenced Time Series')
    plt.xlabel('Year')
    plt.ylabel('Differenced Log(Passengers)')
    plt.grid(True)
    plt.legend()
    plt.show()

    # Run ADF test again on differenced data
    adf_result_diff = adfuller(diff_series)

    print("\n--- ADF Test After Differencing ---")
    print(f'ADF Statistic : {adf_result_diff[0]:.4f}')
    print(f'p-value       : {adf_result_diff[1]:.4f}')
    if adf_result_diff[1] < 0.05:
        print("After differencing, the series is now stationary.")
    else:
        print("The series is still non-stationary. Consider additional transformations.")
```