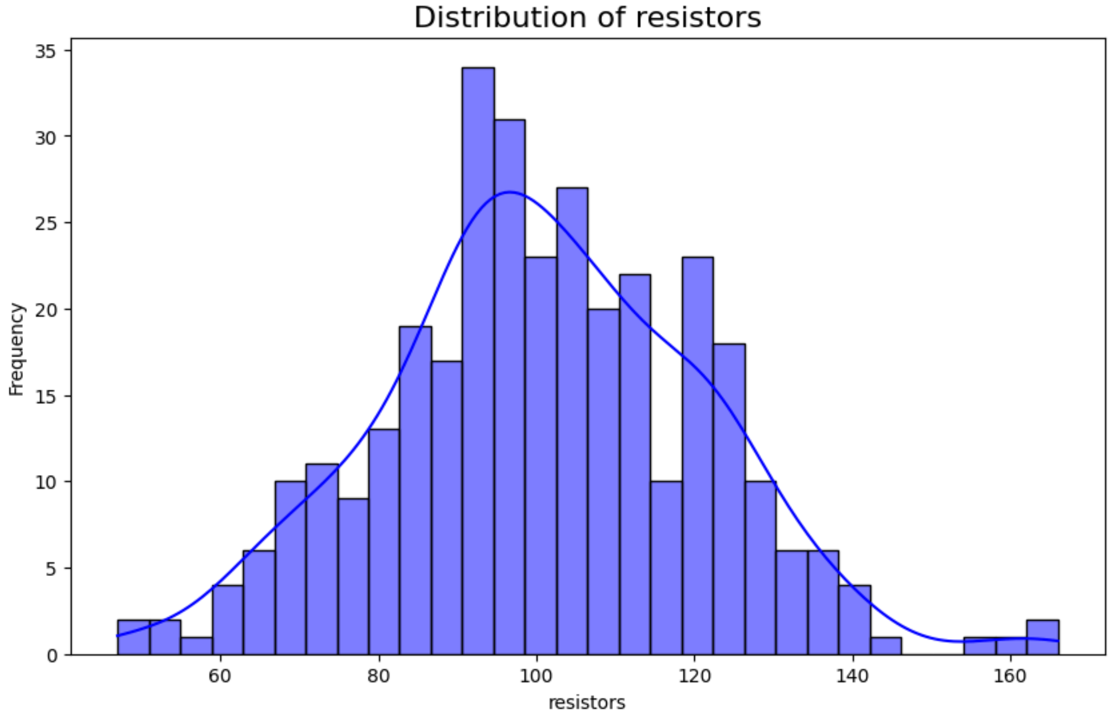
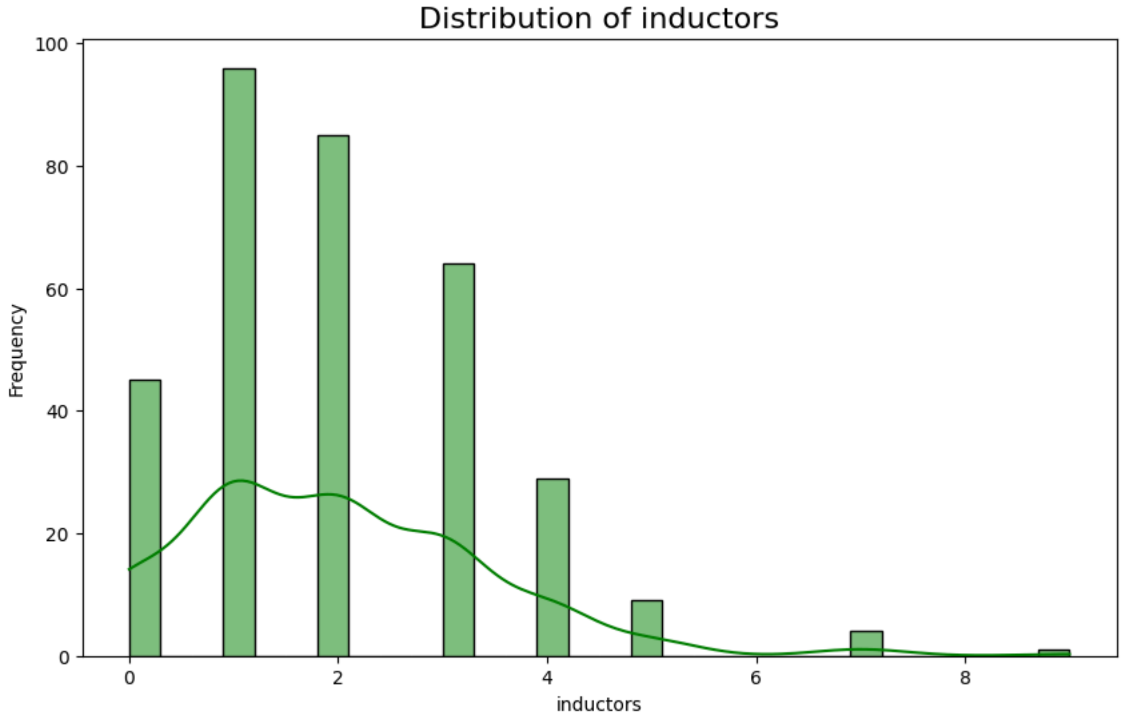
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import shapiro, normaltest, anderson


# A] Normal distribution simplifies data analysis by aligning with common statistical assumptions and enabling efficient modeling. Howev
# Parameter Estimation
# Parameters like the mean and standard deviation fully describe a normal distribution. This simplicity allows for easier and more relia

# Applicability of Z-Scores
# The standard normal distribution (with a mean of 0 and a standard deviation of 1) enables the use of z-scores to calculate probabiliti
# Advantages of Normally Distributed Data
# Simplified Hypothesis Testing

# Many hypothesis tests assume normality because it allows for the use of standard probability tables (e.g., z-tables, t-tables).
# It ensures robust Type I error rates (false positives).
# Predictable Data Behavior

# Approximately 68% of the data lies within 1 standard deviation of the mean, 95% within 2, and 99.7% within 3 (Empirical Rule).
# This predictability makes it easier to understand and interpret data distributions.
# Efficient Modeling

# In machine learning and predictive modeling, algorithms like linear regression, logistic regression, and Gaussian Naive Bayes work opt
# It reduces computational complexity and ensures better performance in such models.
# Robust Sampling and Inferencing

# Normally distributed data ensures that sample means accurately reflect the population mean.
# Confidence intervals and margins of error are more reliable.


#B]Several analytical tests are available to determine the distribution of data. These tests help evaluate whether the data follows a sp
# 1. Shapiro-Wilk Test
# Purpose: Tests whether a sample comes from a normal distribution.
# Test Statistic: W statistic (close to 1 indicates normality).
# Strengths:
# Sensitive for small to medium-sized datasets.
# Accurate and widely used for normality testing.
# Limitations:
# Less reliable for very large datasets, where even small deviations from normality are detected.
# 2. Kolmogorov-Smirnov (K-S) Test
# Purpose: Compares the sample distribution with a reference distribution (e.g., normal, uniform).
# Test Statistic: Maximum difference (D) between the empirical cumulative distribution function (ECDF) and the theoretical cumulative di
# Strengths:
# Can test against any theoretical distribution, not just normal.
# Limitations:
# Less sensitive to deviations in the tails of the distribution.
# Not very powerful for small sample sizes.
# 3. Anderson-Darling Test
# Purpose: A more sensitive alternative to the K-S test that gives extra weight to deviations in the tails of the distribution.
# Test Statistic: Modified D-statistic (tail-weighted).
# Strengths:
# Focuses on the tails of the distribution.
# Works well for normality testing.
# Limitations:
# Requires a sufficiently large sample size for reliable results.


#C]To check for normality of data using graphical methods, you can follow these commonly used techniques:

# 1. Histogram:
# A histogram shows the frequency distribution of the data. For a normally distributed dataset, the histogram should resemble a bell curv
# Steps to interpret:
# The data should be symmetrically distributed around the mean.
# The shape of the histogram should resemble a bell-shaped curve.
# 2. Q-Q (Quantile-Quantile) Plot:
# A Q-Q plot compares your data's quantiles with those of a normal distribution.
# Steps to interpret:
# If the points roughly follow the line (45-degree diagonal), it indicates that the data is approximately normally distributed.
# Deviations from this line suggest non-normality.
# 3. Box Plot:
# The box plot (or box-and-whisker plot) visually summarizes data distribution, including outliers.
# Steps to interpret:
# A normal distribution would typically have no clear outliers and a symmetric box plot.
# 4. Density Plot:
# A density plot shows the probability density function of a dataset.
# Steps to interpret:
# If the plot forms a smooth curve resembling a bell curve, it suggests normality.
# The curve should be unimodal (one peak) and symmetric.
# 5. Stem-and-Leaf Plot:
# This plot provides a graphical display of the shape of the distribution while retaining the actual data values.
```

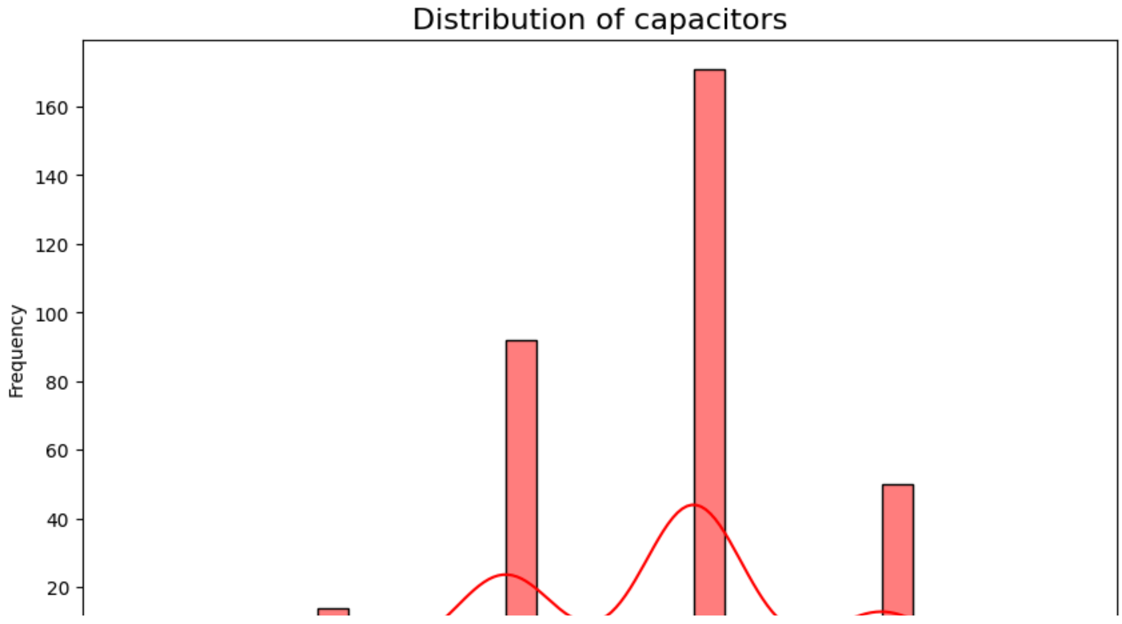
```
# Steps to interpret:
# A stem-and-leaf plot for normally distributed data would show values grouped around the center, and few outliers.


#D] Compare Graphical and Analytical Methods:
# After analyzing the data using both graphical and analytical methods:
# If both the graphical and analytical results indicate normality (e.g., similar bell-shaped curves, high p-value from tests, and skewnes
# If discrepancies occur (e.g., visual bell curve but p-value from the test is low), then the results might not agree, and further invest
# 4. Conclusion:
# The product that follows a normal distribution will show agreement between the graphical (bell curve, symmetric box plot, etc.) and ana


print("Dataset preview:")
print(data.head())
```

```
Dataset preview:
   day  inductors  capacitors  resistors  Unnamed: 4 Unnamed: 5 Unnamed: 6  \
0    1          0          10         96         NaN        NaN        NaN
1    2          3          10         92         NaN        NaN        NaN
2    3          1           9         84         NaN        NaN        NaN
3    4          2          10        113         NaN        NaN        NaN
4    5          1          10         65         NaN        NaN        NaN

  Unnamed: 7 Unnamed: 8 Unnamed: 9 Unnamed: 10
0        NaN        NaN        NaN         NaN
1        NaN        NaN        NaN         NaN
2        NaN        NaN        NaN         NaN
3        NaN        NaN        NaN         NaN
4        NaN        NaN        NaN         NaN
```

```
Resistors = data['resistors']
inductors = data['inductors']
capacitors = data['capacitors']

# Function to plot histograms and KDE
def plot_distributions(data, title, color):
    plt.figure(figsize=(10, 6))
    sns.histplot(data, kde=True, color=color, bins=30)
    plt.title(f'Distribution of {title}', fontsize=16)
    plt.xlabel(title)
    plt.ylabel('Frequency')
    plt.show()


plot_distributions(Resistors, "resistors", "blue")
plot_distributions(inductors, "inductors", "green")
plot_distributions(capacitors, "capacitors", "red")
```


Distribution of resistors
Frequency
resistors
0
5
10
15
20
25
30
35
60
80
100
120
140
160


Distribution of inductors
Frequency
inductors
0
20
40
60
80
100
0
2
4
6
8


Distribution of capacitors
Frequency
20
40
60
80
100
120
140
160


```
def normality_tests(data, label):
    print(f"\nNormality Tests for {label}:\n")



    # D'Agostino's K-squared Test
    stat, p = normaltest(data)
    print(f"D'Agostino's K-squared Test: Statistic = {stat:.4f}, p-value = {p:.4f}")
    if p > 0.05:
        print("The data is likely normal (fail to reject H0).")
    else:
        print("The data is not normal (reject H0).")


# Perform normality tests
normality_tests(Resistors, "resistors")
normality_tests(inductors, "inductors")
normality_tests(capacitors, "capacitors")
```

```
Normality Tests for resistors:

D'Agostino's K-squared Test: Statistic = 1.5714, p-value = 0.4558
```