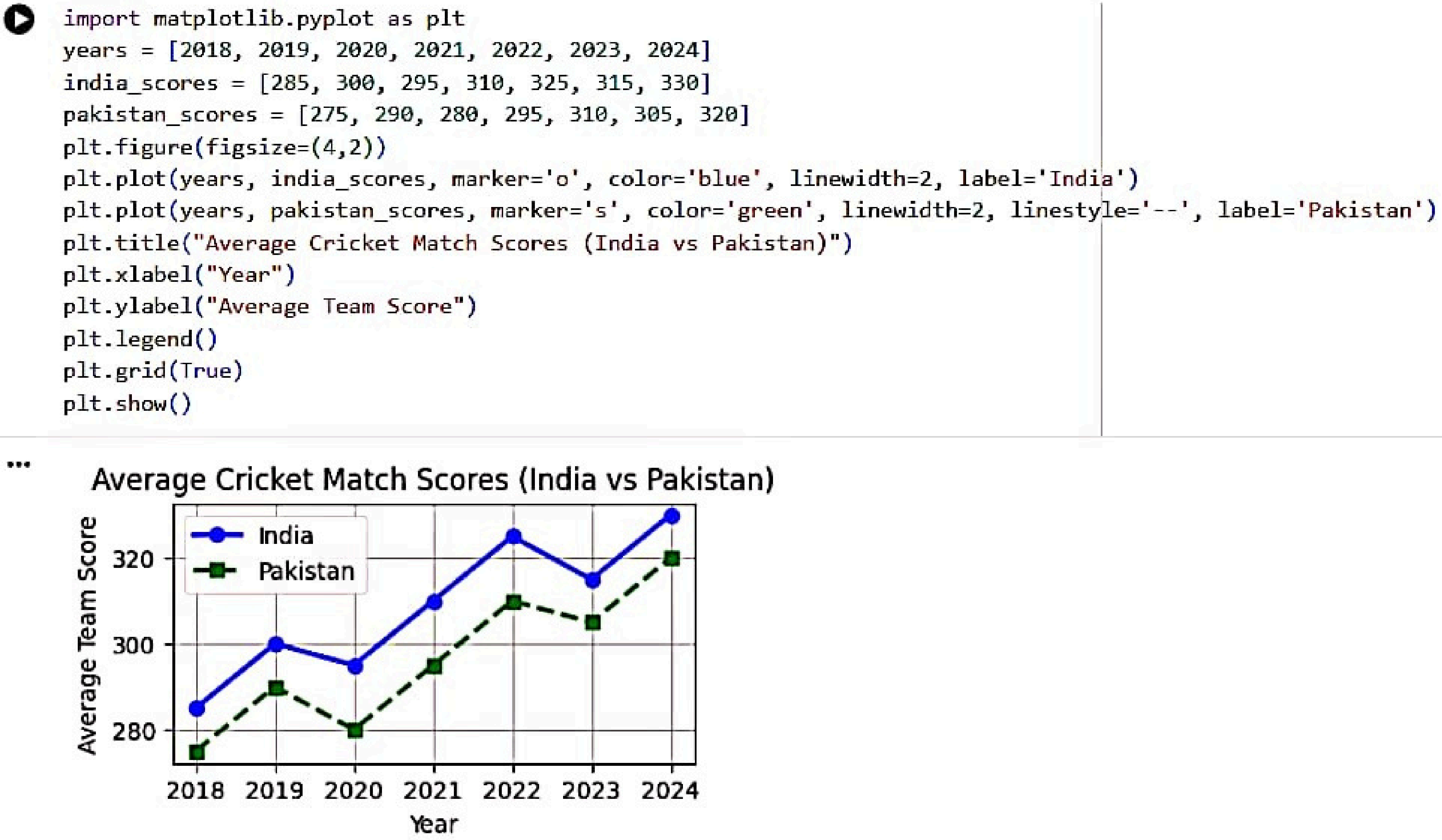
RAJALAKSHMI ENGINEERING COLLEGE

Approved by AICTE | Affiliated to Anna University | Accredited by NAAC

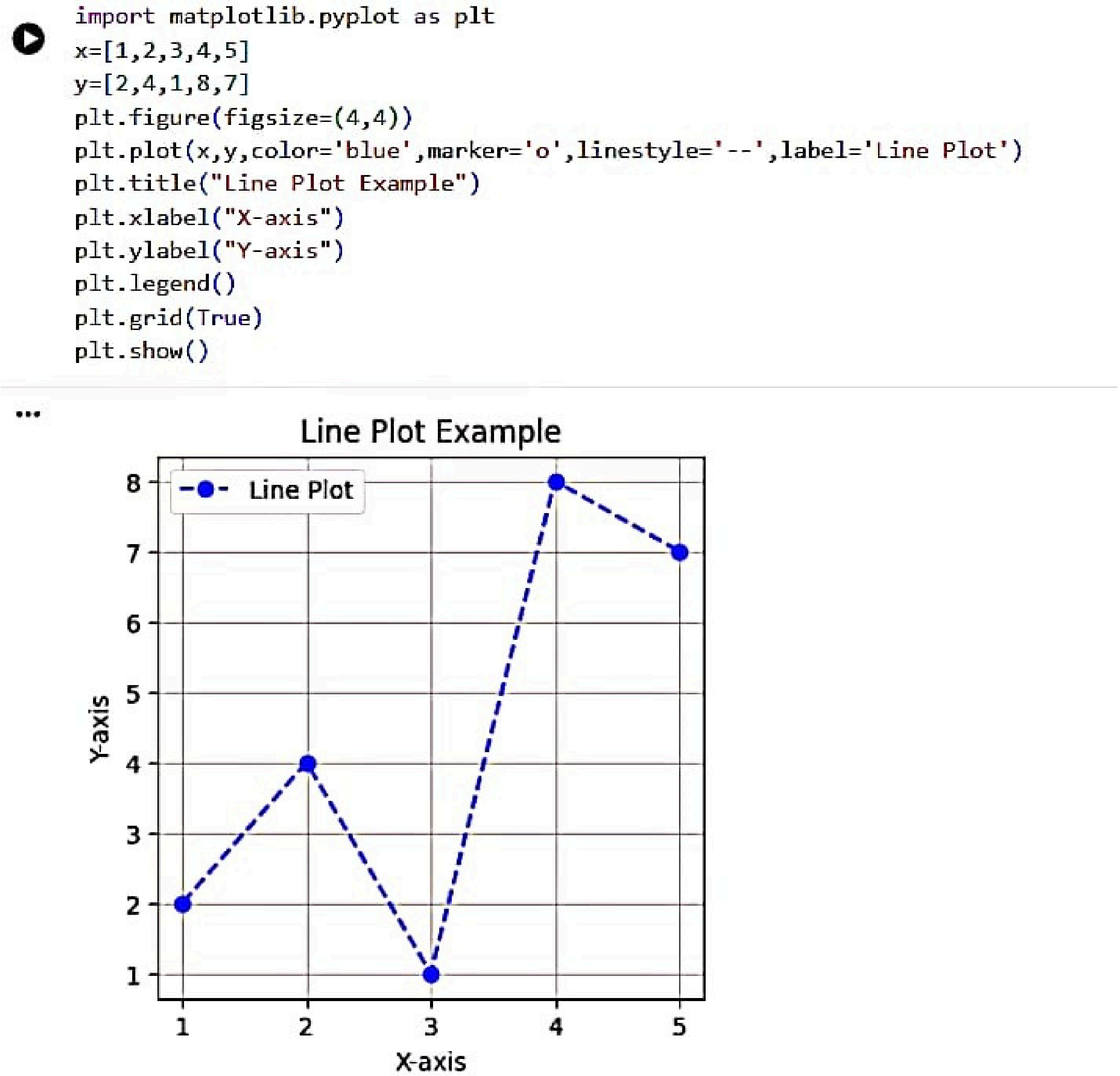
Department of Computer Science and Engineering

CS23334 Fundamentals of Data Science Lab
III semester II Year (2023R)

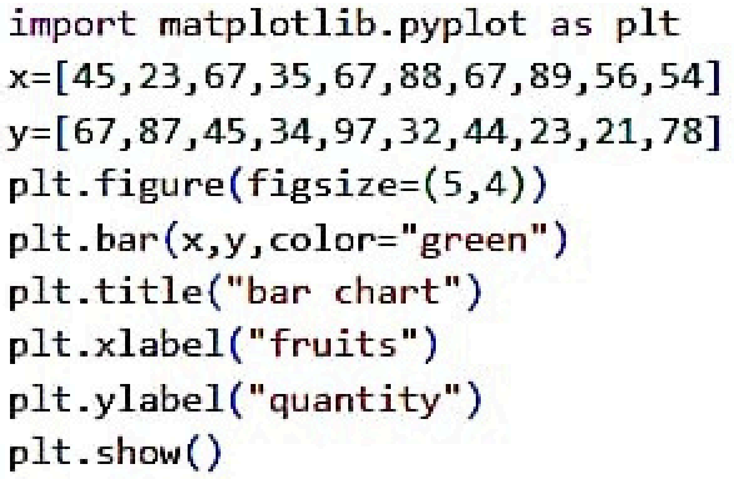
Name of the Student: **HEMALATHA ST**
Register Number: **240701189**



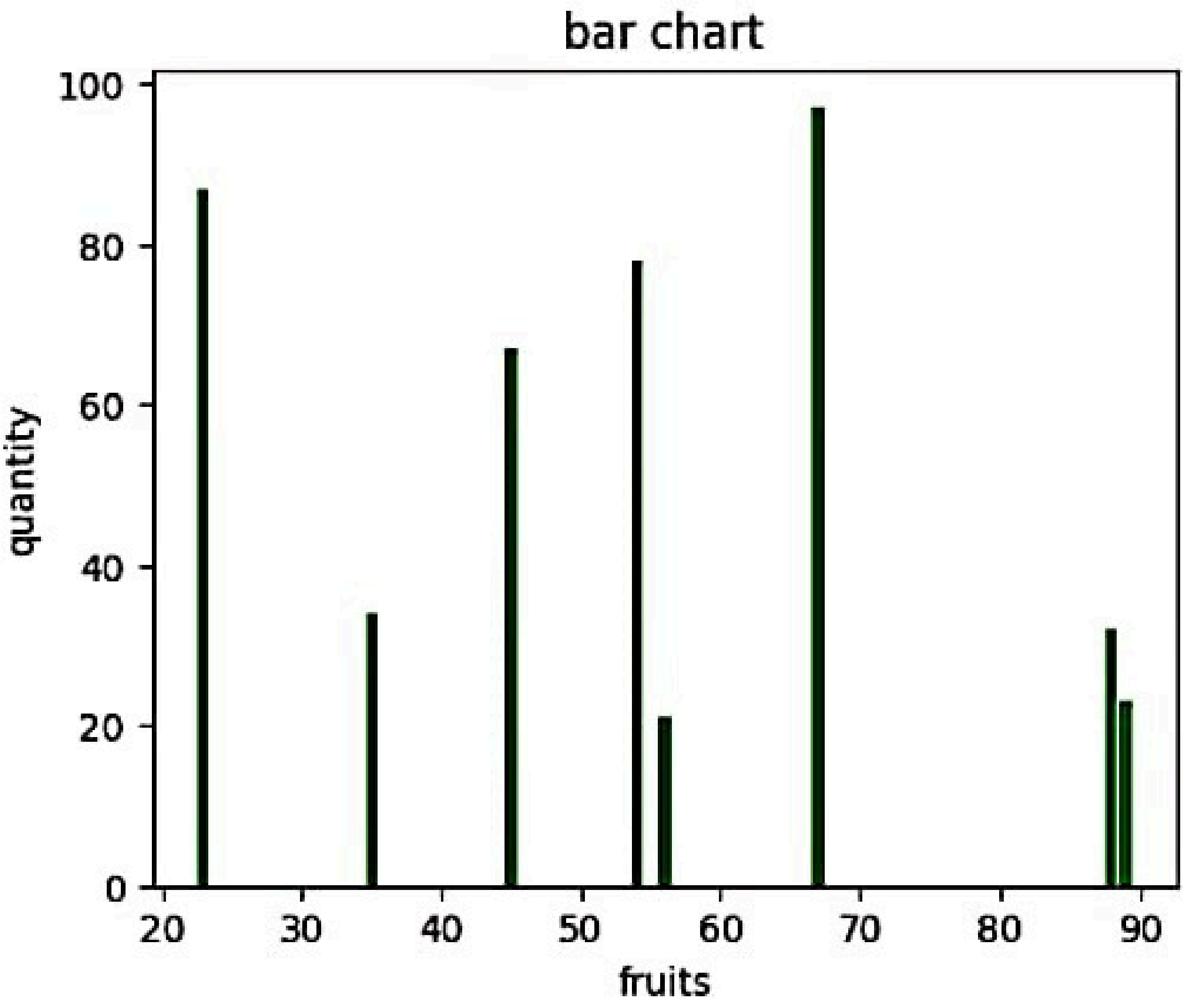
```
import matplotlib.pyplot as plt
years = [2018, 2019, 2020, 2021, 2022, 2023, 2024]
india_scores = [285, 300, 295, 310, 325, 315, 330]
pakistan_scores = [275, 290, 280, 295, 310, 305, 320]
plt.figure(figsize=(4,2))
plt.plot(years, india_scores, marker='o', color='blue', linewidth=2, label='India')
plt.plot(years, pakistan_scores, marker='s', color='green', linewidth=2, linestyle='--', label='Pakistan')
plt.title("Average Cricket Match Scores (India vs Pakistan)")
plt.xlabel("Year")
plt.ylabel("Average Team Score")
plt.legend()
plt.grid(True)
plt.show()
```

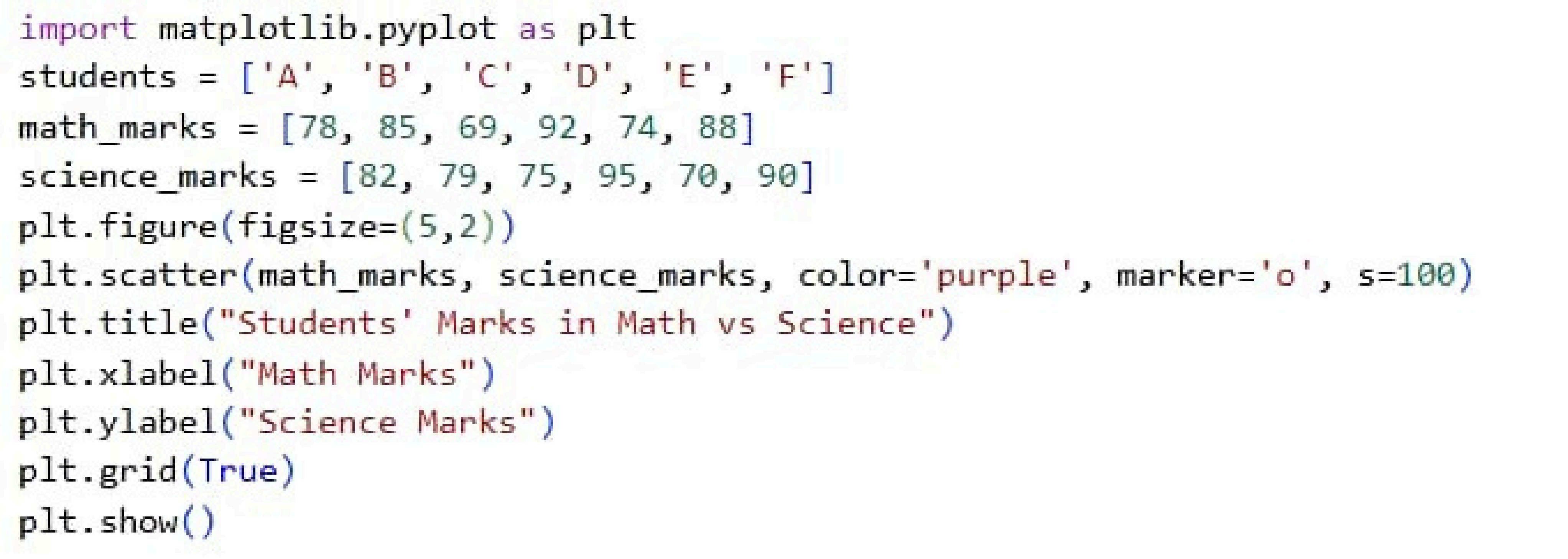


```
import matplotlib.pyplot as plt
x=[1,2,3,4,5]
y=[2,4,1,8,7]
plt.figure(figsize=(4,4))
plt.plot(x,y,color='blue',marker='o',linestyle='--',label='Line Plot')
plt.title("Line Plot Example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.grid(True)
plt.show()
```

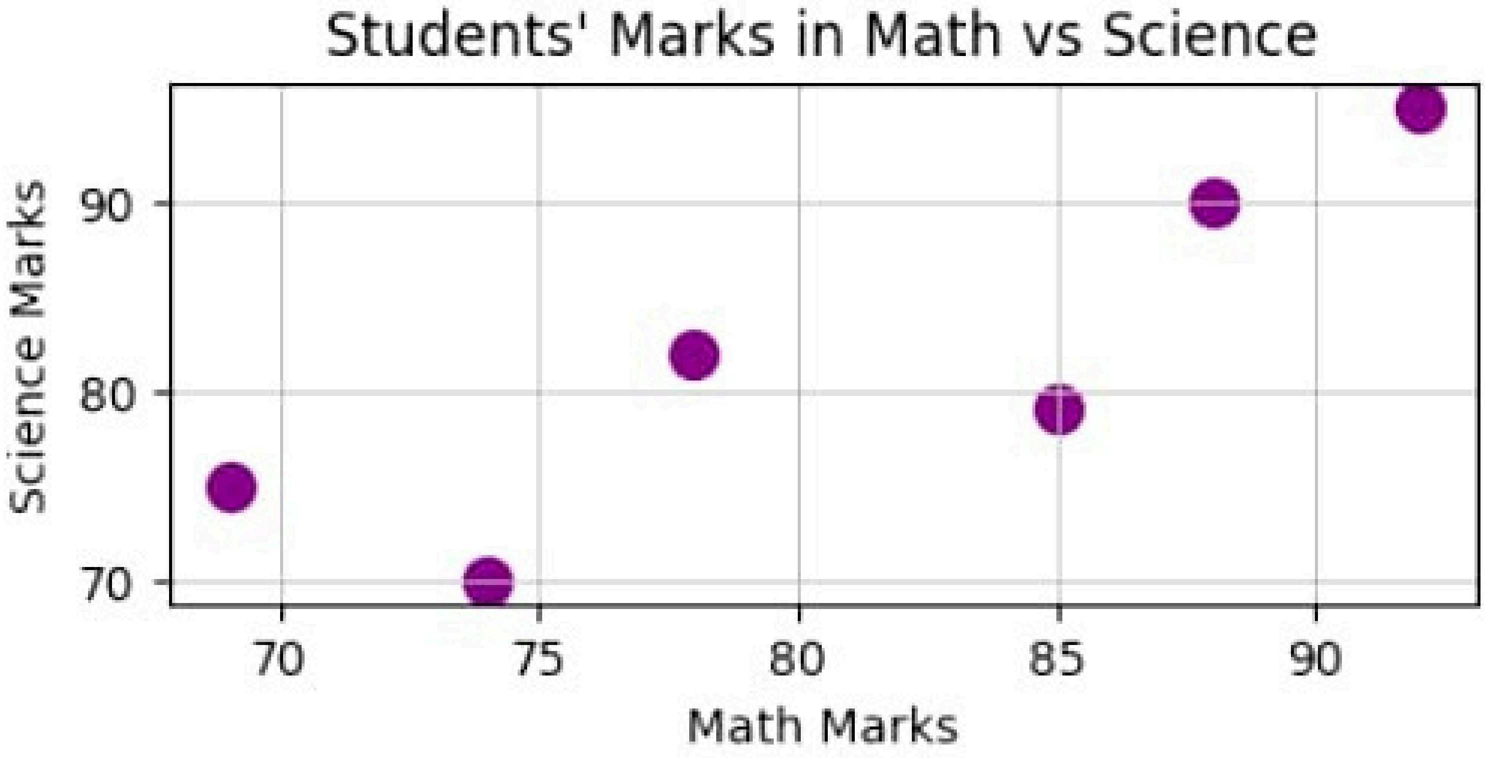


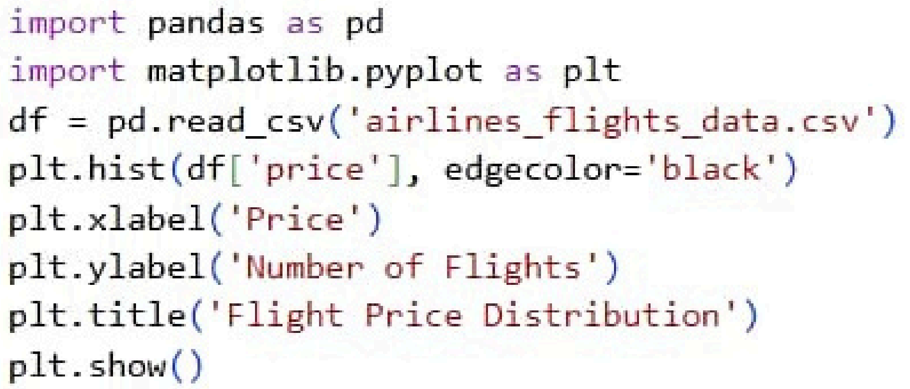
```
import matplotlib.pyplot as plt
x=[45,23,67,35,67,88,67,89,56,54]
y=[67,87,45,34,97,32,44,23,21,78]
plt.figure(figsize=(5,4))
plt.bar(x,y,color="green")
plt.title("bar chart")
plt.xlabel("fruits")
plt.ylabel("quantity")
plt.show()
```



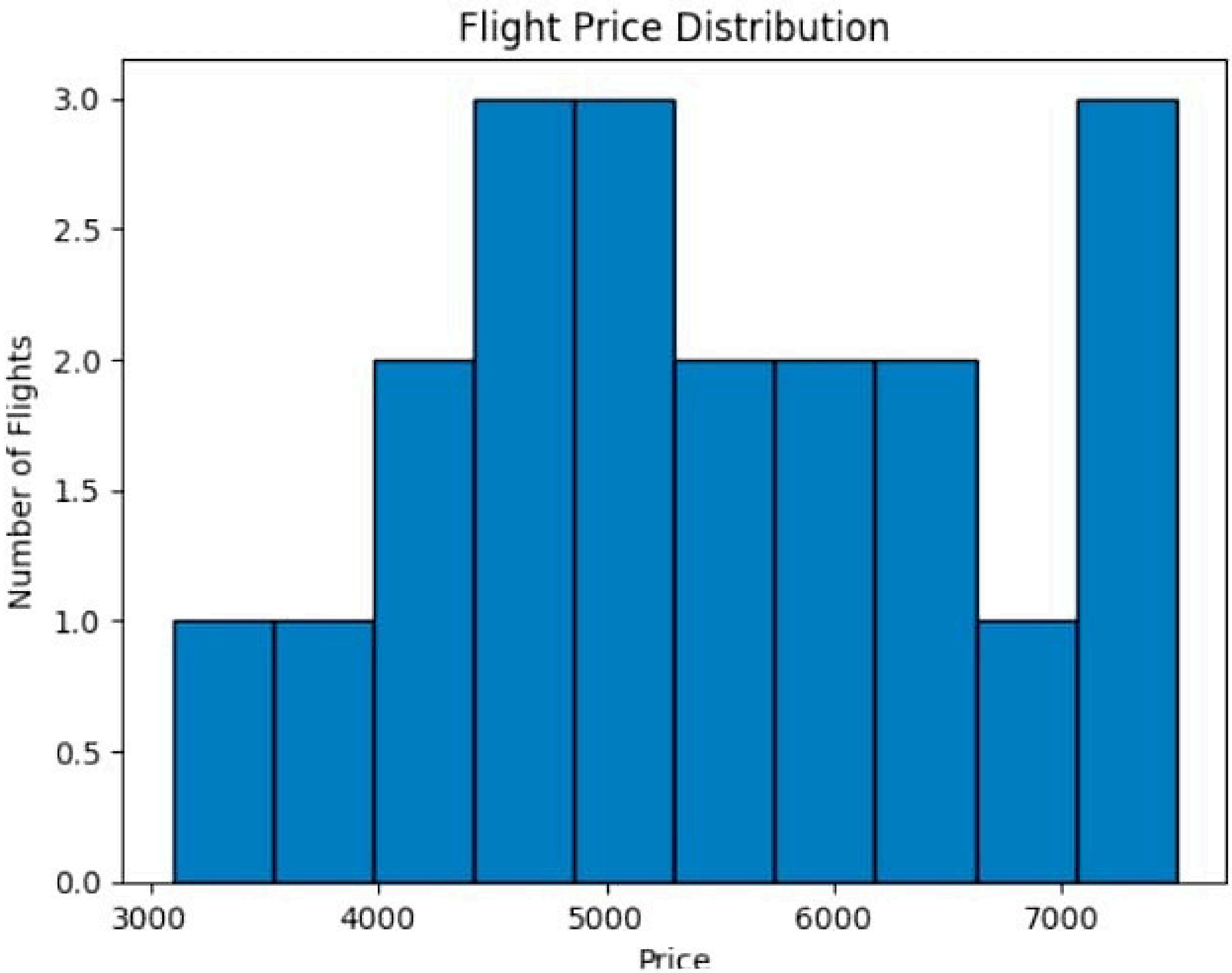


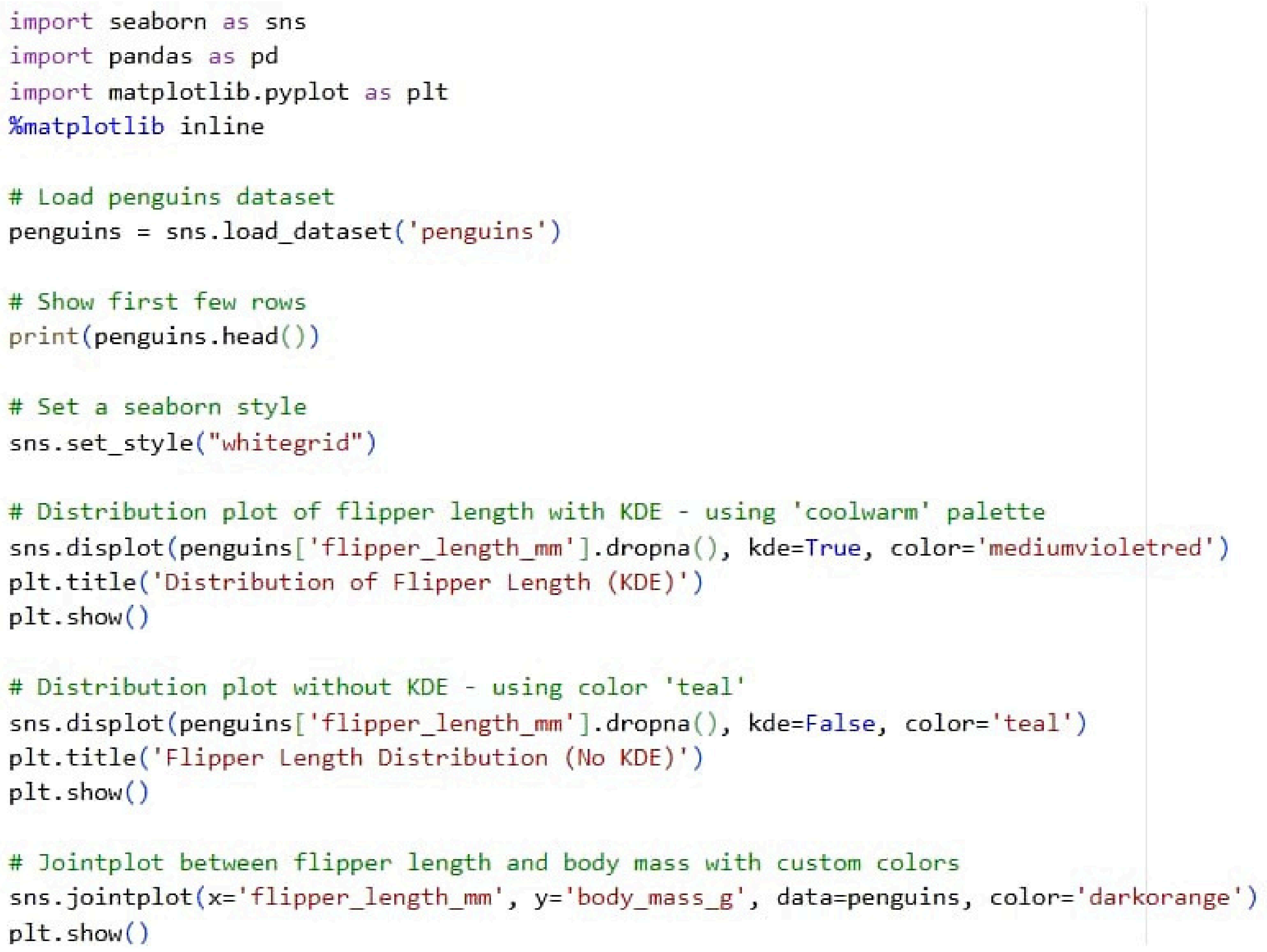
```python
import matplotlib.pyplot as plt
students = ['A', 'B', 'C', 'D', 'E', 'F']
math_marks = [78, 85, 69, 92, 74, 88]
science_marks = [82, 79, 75, 95, 70, 90]
plt.figure(figsize=(5,2))
plt.scatter(math_marks, science_marks, color='purple', marker='o', s=100)
plt.title("Students' Marks in Math vs Science")
plt.xlabel("Math Marks")
plt.ylabel("Science Marks")
plt.grid(True)
plt.show()
```



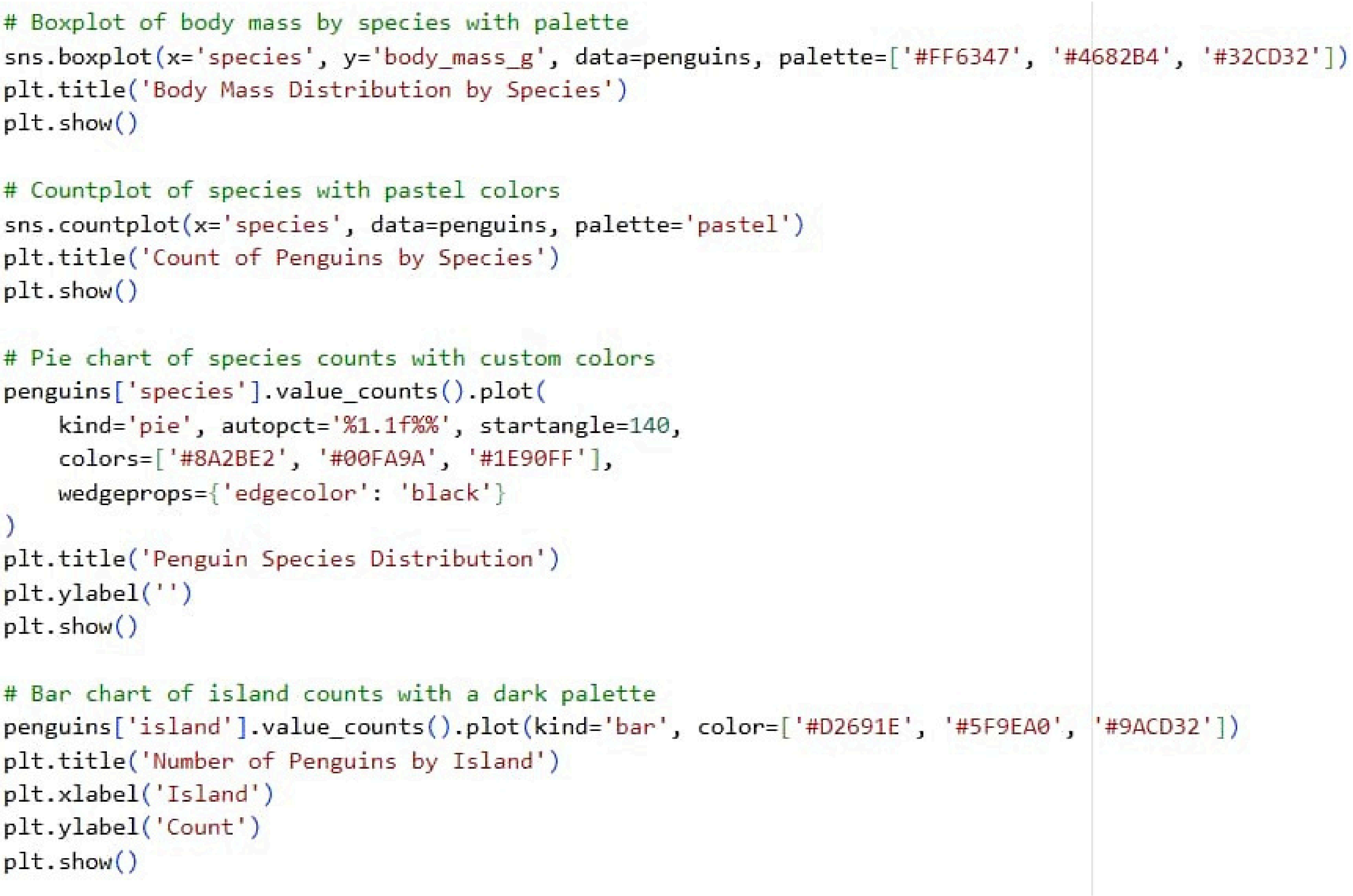


```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('airlines_flights_data.csv')
plt.hist(df['price'], edgecolor='black')
plt.xlabel('Price')
plt.ylabel('Number of Flights')
plt.title('Flight Price Distribution')
plt.show()
```

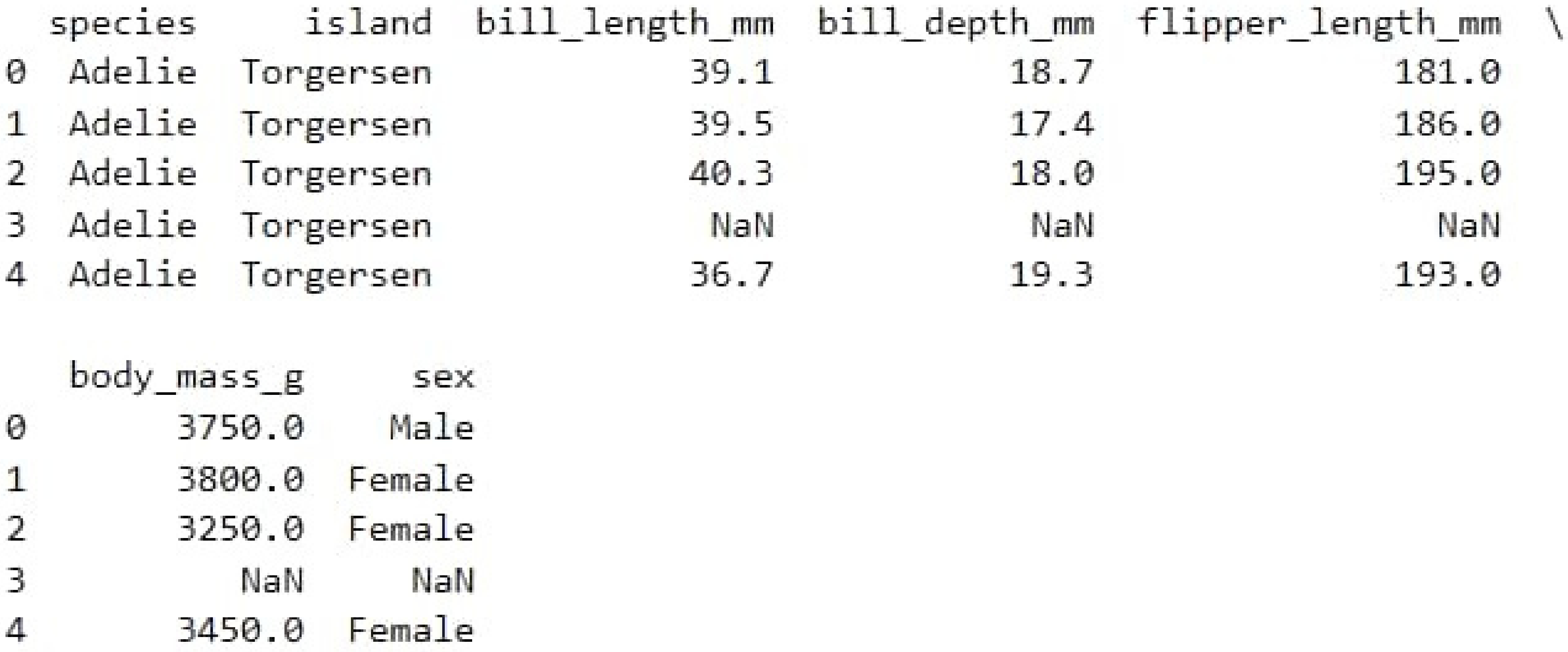




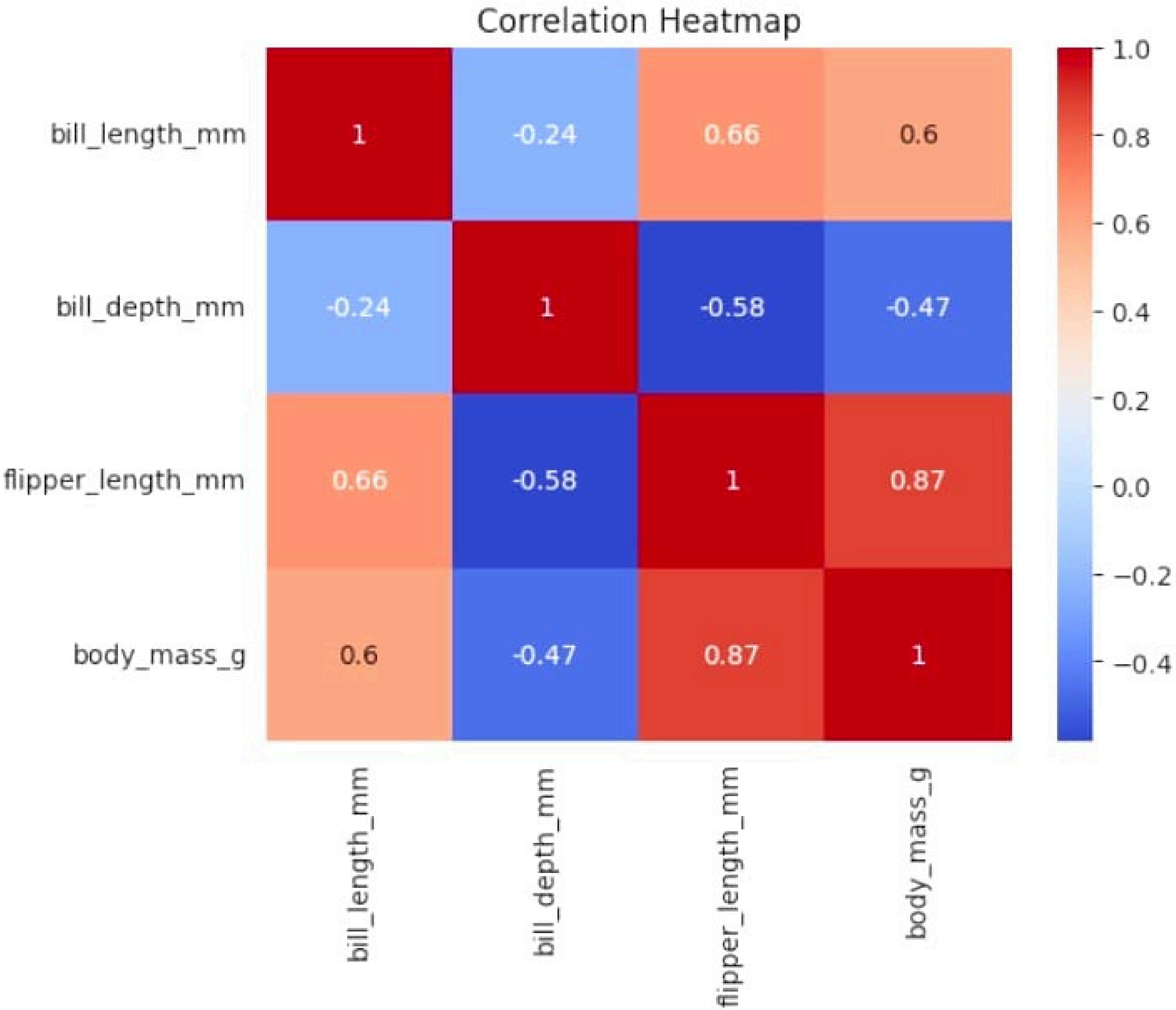
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

# Load penguins dataset
penguins = sns.load_dataset('penguins')

# Show first few rows
print(penguins.head())

# Set a seaborn style
sns.set_style("whitegrid")

# Distribution plot of flipper length with KDE - using 'coolwarm' palette
sns.displot(penguins['flipper_length_mm'].dropna(), kde=True, color='mediumvioletred')
plt.title('Distribution of Flipper Length (KDE)')
plt.show()

# Distribution plot without KDE - using color 'teal'
sns.displot(penguins['flipper_length_mm'].dropna(), kde=False, color='teal')
plt.title('Flipper Length Distribution (No KDE)')
plt.show()

# Jointplot between flipper length and body mass with custom colors
sns.jointplot(x='flipper_length_mm', y='body_mass_g', data=penguins, color='darkorange')
plt.show()
```

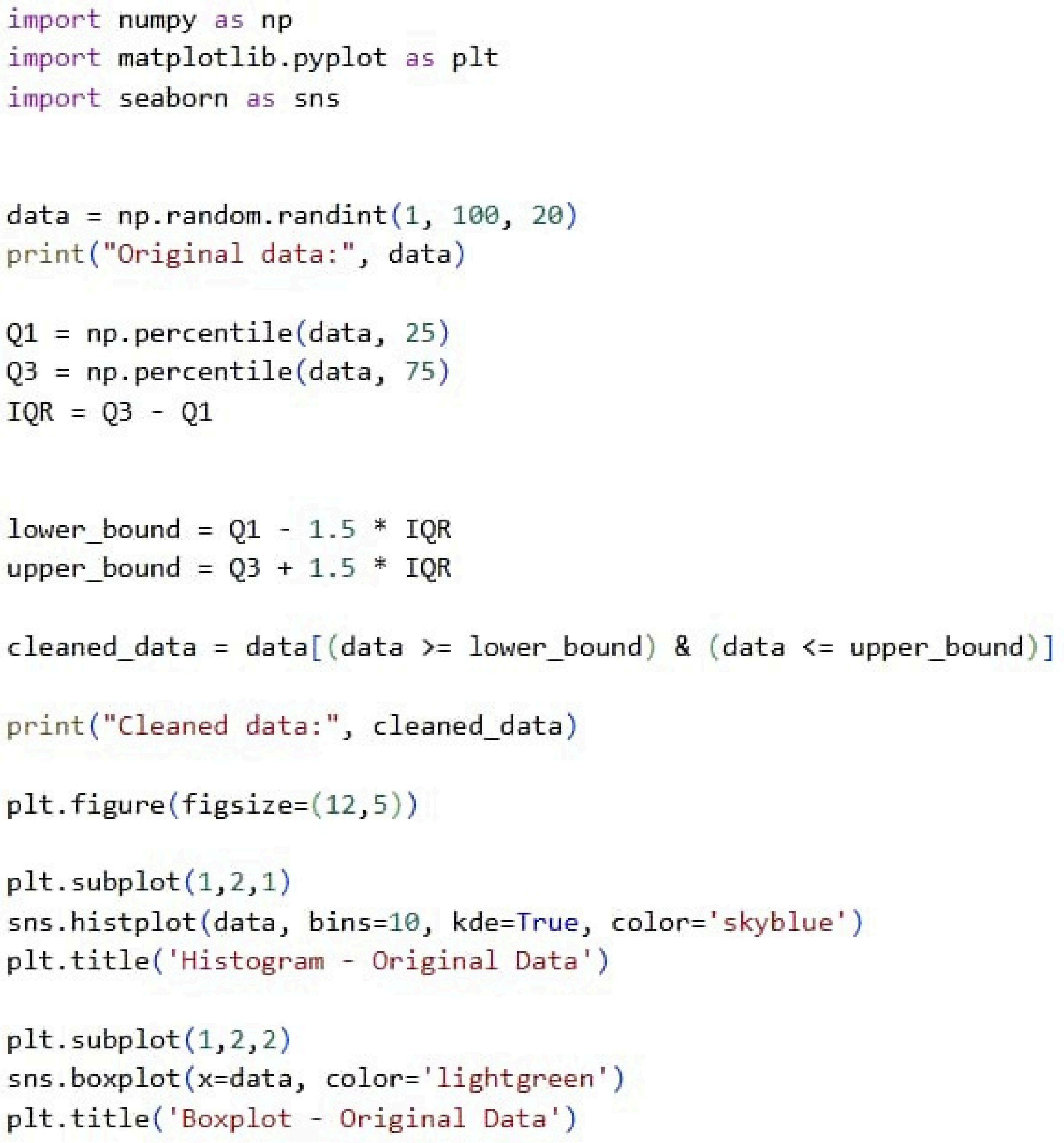


```python
# Boxplot of body mass by species with palette
sns.boxplot(x='species', y='body_mass_g', data=penguins, palette=['#FF6347', '#4682B4', '#32CD32'])
plt.title('Body Mass Distribution by Species')
plt.show()

# Countplot of species with pastel colors
sns.countplot(x='species', data=penguins, palette='pastel')
plt.title('Count of Penguins by Species')
plt.show()

# Pie chart of species counts with custom colors
penguins['species'].value_counts().plot(
    kind='pie', autopct='%1.1f%%', startangle=140,
    colors=['#8A2BE2', '#00FA9A', '#1E90FF'],
    wedgeprops={'edgecolor': 'black'}
)
plt.title('Penguin Species Distribution')
plt.ylabel('')
plt.show()

# Bar chart of island counts with a dark palette
penguins['island'].value_counts().plot(kind='bar', color=['#D2691E', '#5F9EA0', '#9ACD32'])
plt.title('Number of Penguins by Island')
plt.xlabel('Island')
plt.ylabel('Count')
plt.show()
```

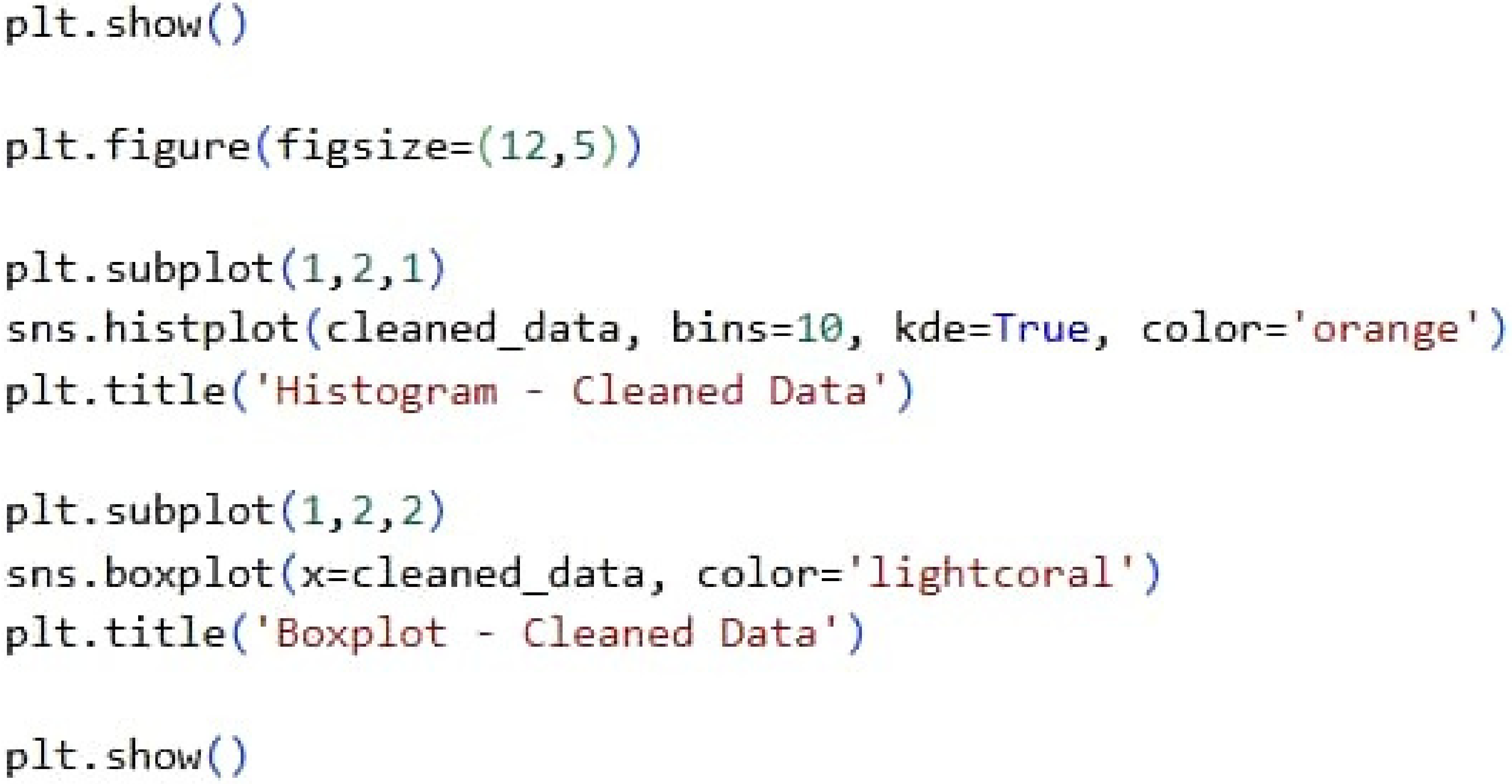


```
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0   
1  Adelie  Torgersen            39.5           17.4              186.0   
2  Adelie  Torgersen            40.3           18.0              195.0   
3  Adelie  Torgersen             NaN            NaN                NaN   
4  Adelie  Torgersen            36.7           19.3              193.0   

   body_mass_g     sex  
0       3750.0    Male  
1       3800.0  Female  
2       3250.0  Female  
3          NaN     NaN  
4       3450.0  Female  
```

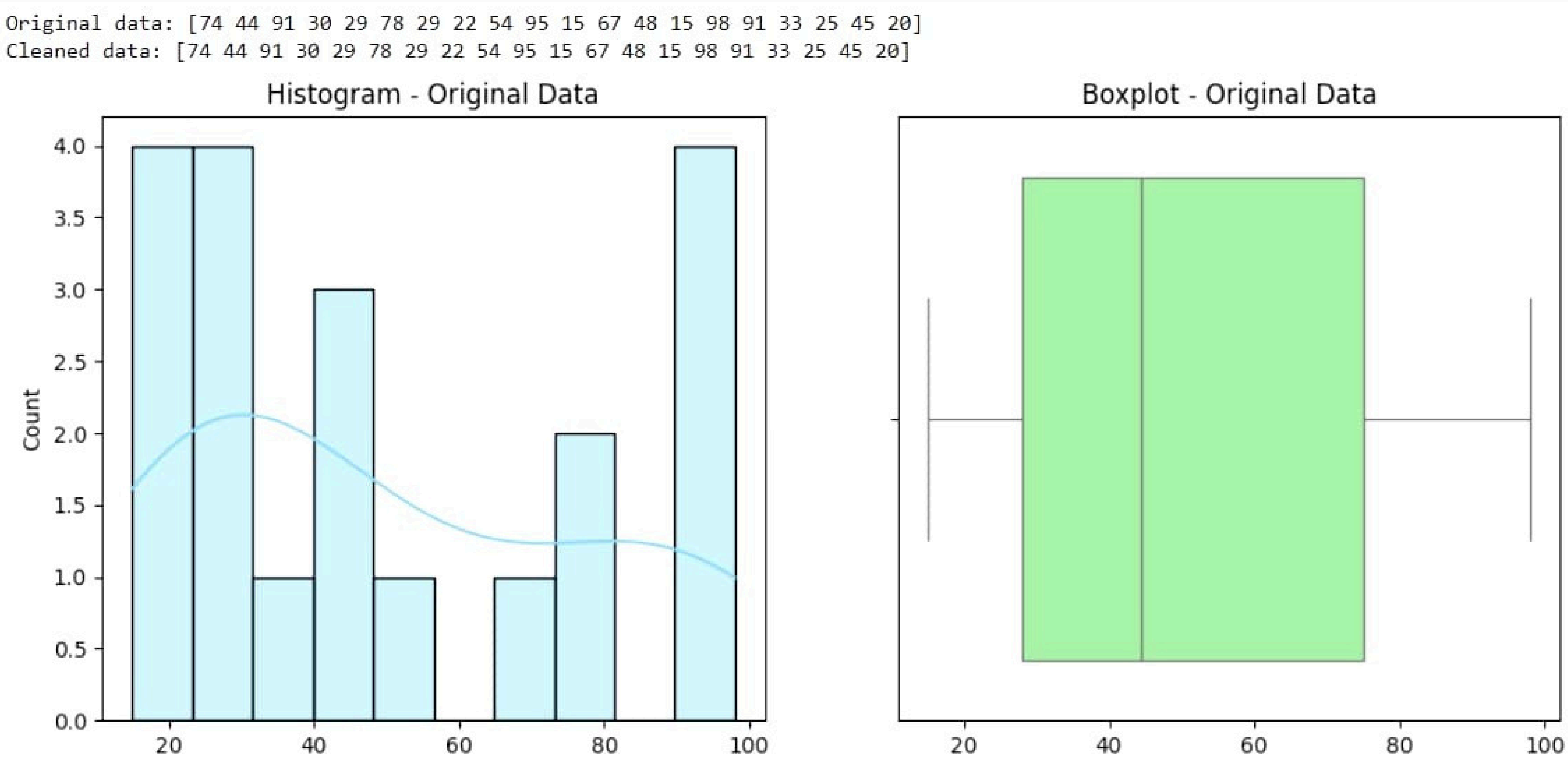

Correlation Heatmap
bill_length_mm
1
-0.24
0.66
0.6
bill_depth_mm
-0.24
1
-0.58
-0.47
flipper_length_mm
0.66
-0.58
1
0.87
body_mass_g
0.6
-0.47
0.87
1
bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4



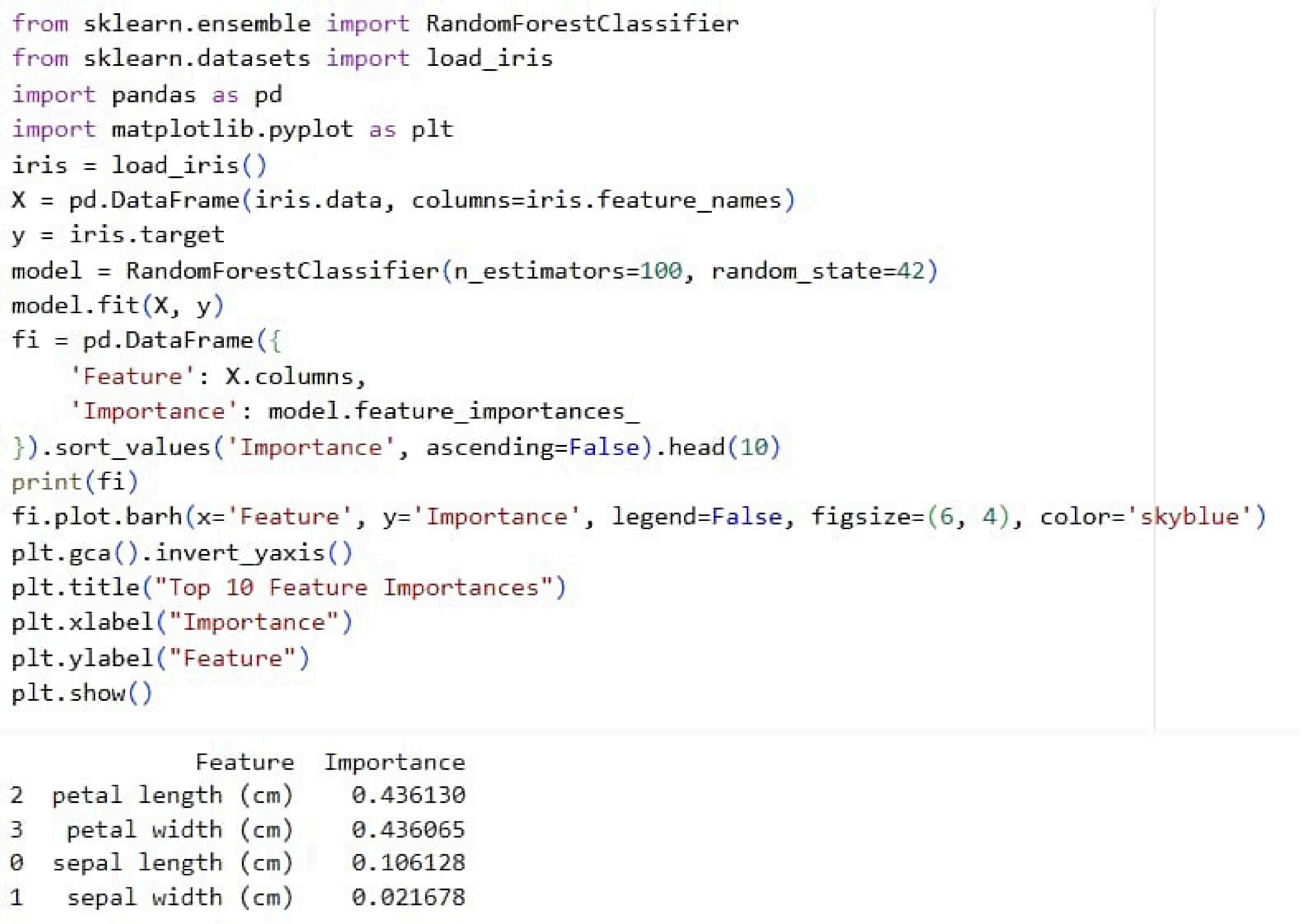
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


data = np.random.randint(1, 100, 20)
print("Original data:", data)

Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)
IQR = Q3 - Q1


lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

cleaned_data = data[(data >= lower_bound) & (data <= upper_bound)]

print("Cleaned data:", cleaned_data)

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
sns.histplot(data, bins=10, kde=True, color='skyblue')
plt.title('Histogram - Original Data')

plt.subplot(1,2,2)
sns.boxplot(x=data, color='lightgreen')
plt.title('Boxplot - Original Data')
```



```
plt.show()

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
sns.histplot(cleaned_data, bins=10, kde=True, color='orange')
plt.title('Histogram - Cleaned Data')

plt.subplot(1,2,2)
sns.boxplot(x=cleaned_data, color='lightcoral')
plt.title('Boxplot - Cleaned Data')

plt.show()
```

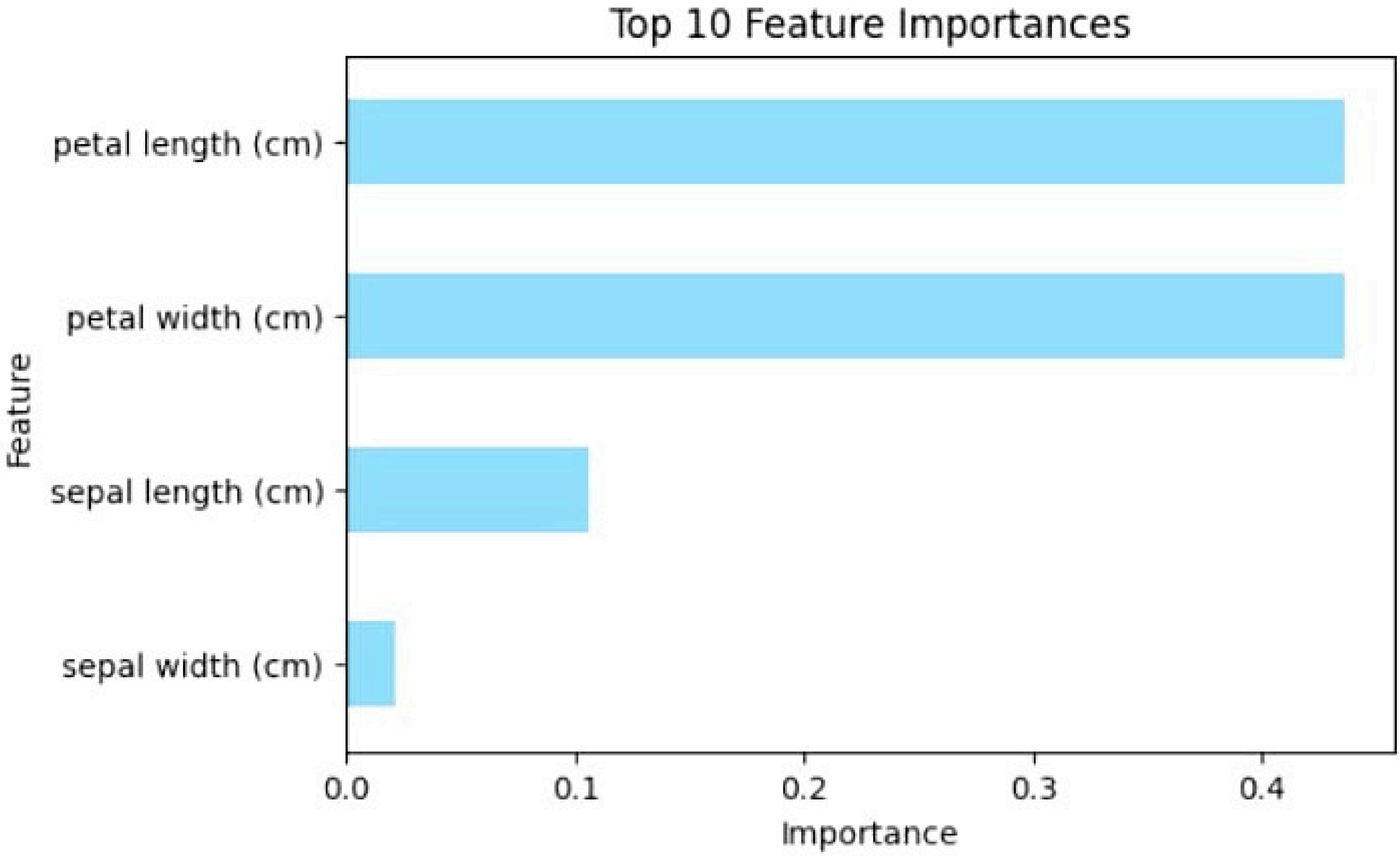


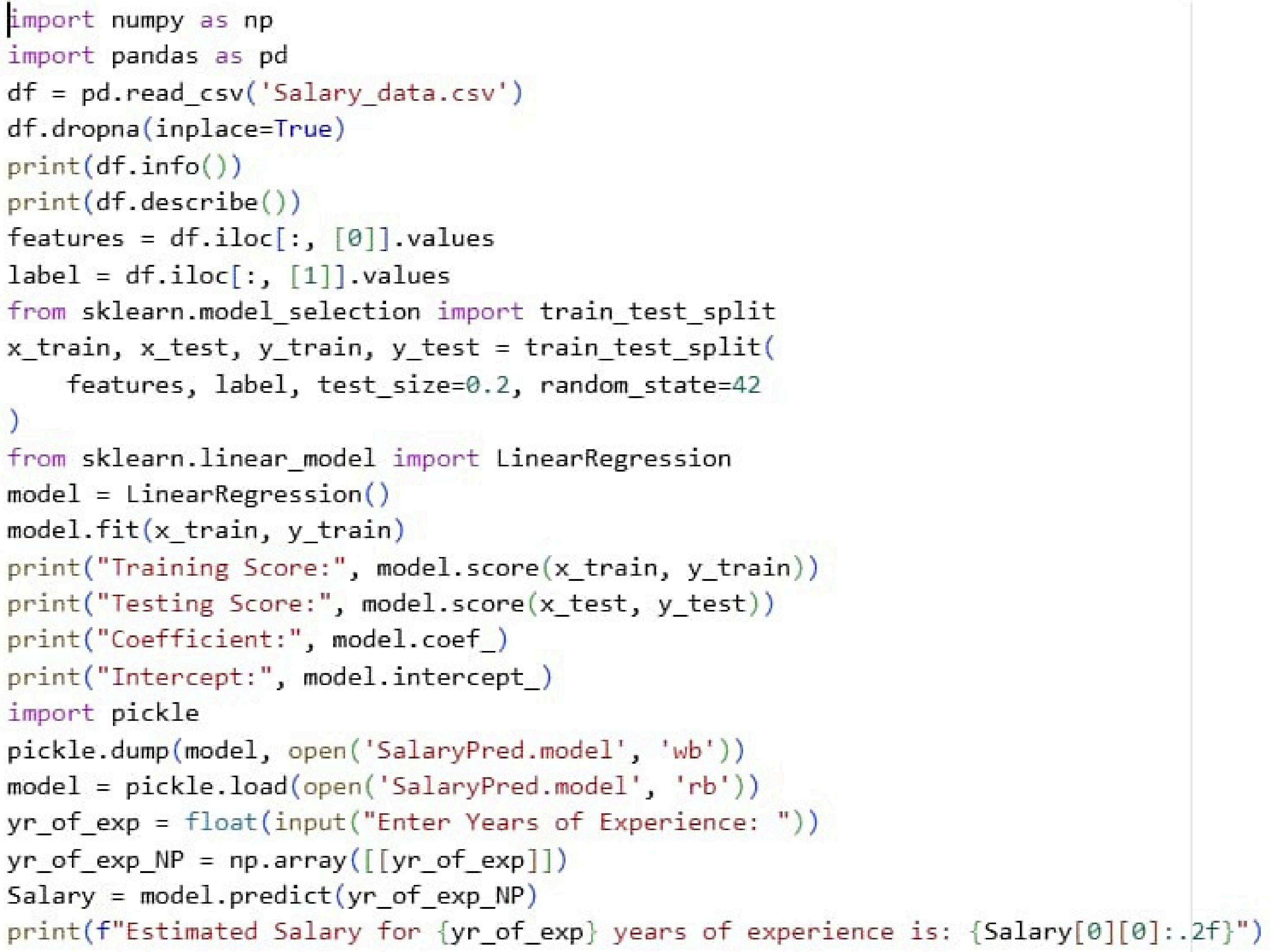
```
Original data: [74 44 91 30 29 78 29 22 54 95 15 67 48 15 98 91 33 25 45 20]
Cleaned data: [74 44 91 30 29 78 29 22 54 95 15 67 48 15 98 91 33 25 45 20]
```



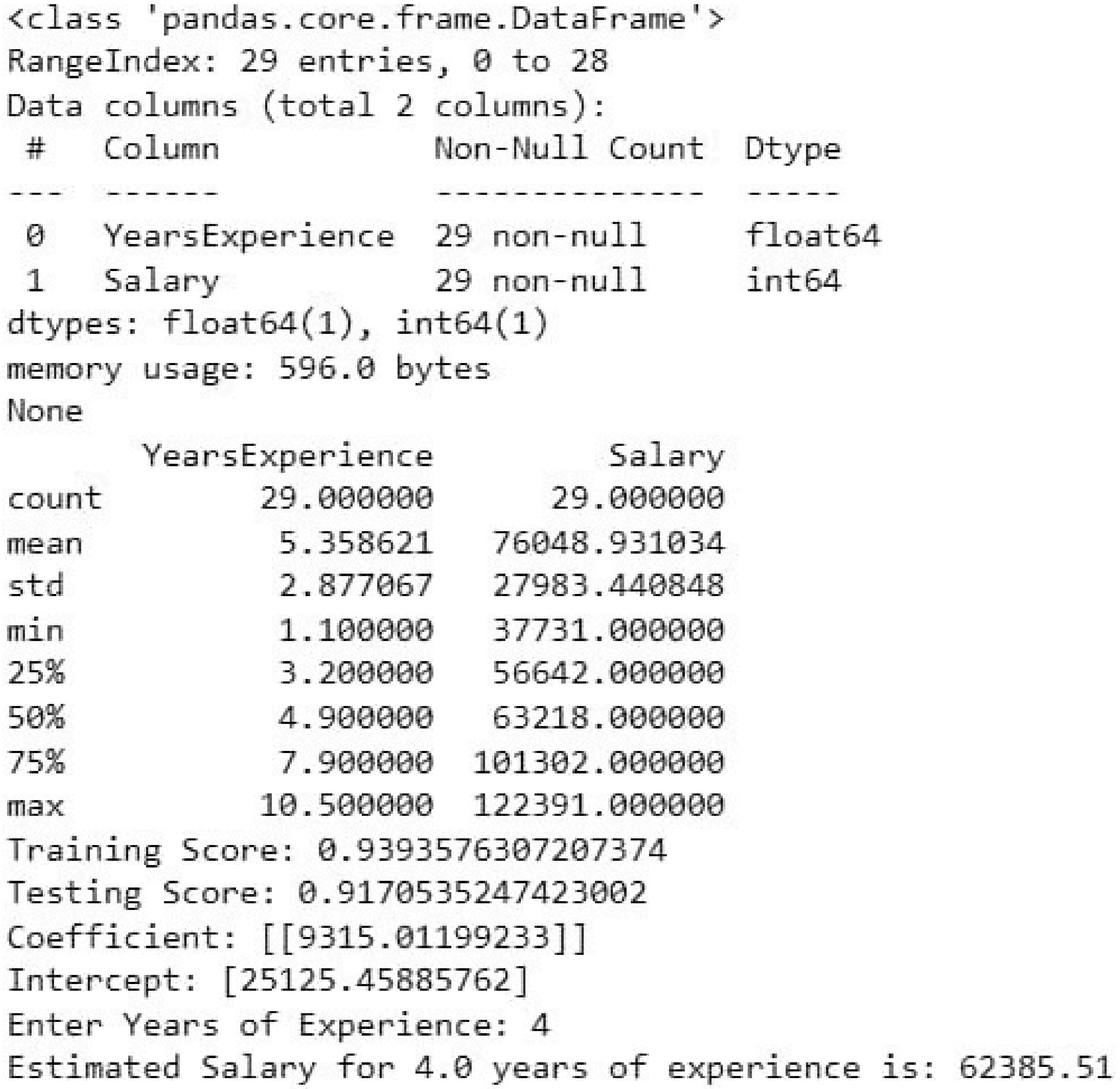
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
import pandas as pd
import matplotlib.pyplot as plt
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X, y)
fi = pd.DataFrame({
    'Feature': X.columns,
    'Importance': model.feature_importances_
}).sort_values('Importance', ascending=False).head(10)
print(fi)
fi.plot.barh(x='Feature', y='Importance', legend=False, figsize=(6, 4), color='skyblue')
plt.gca().invert_yaxis()
plt.title("Top 10 Feature Importances")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```

```
             Feature  Importance
2  petal length (cm)    0.436130
3   petal width (cm)    0.436065
0  sepal length (cm)    0.106128
1   sepal width (cm)    0.021678
```

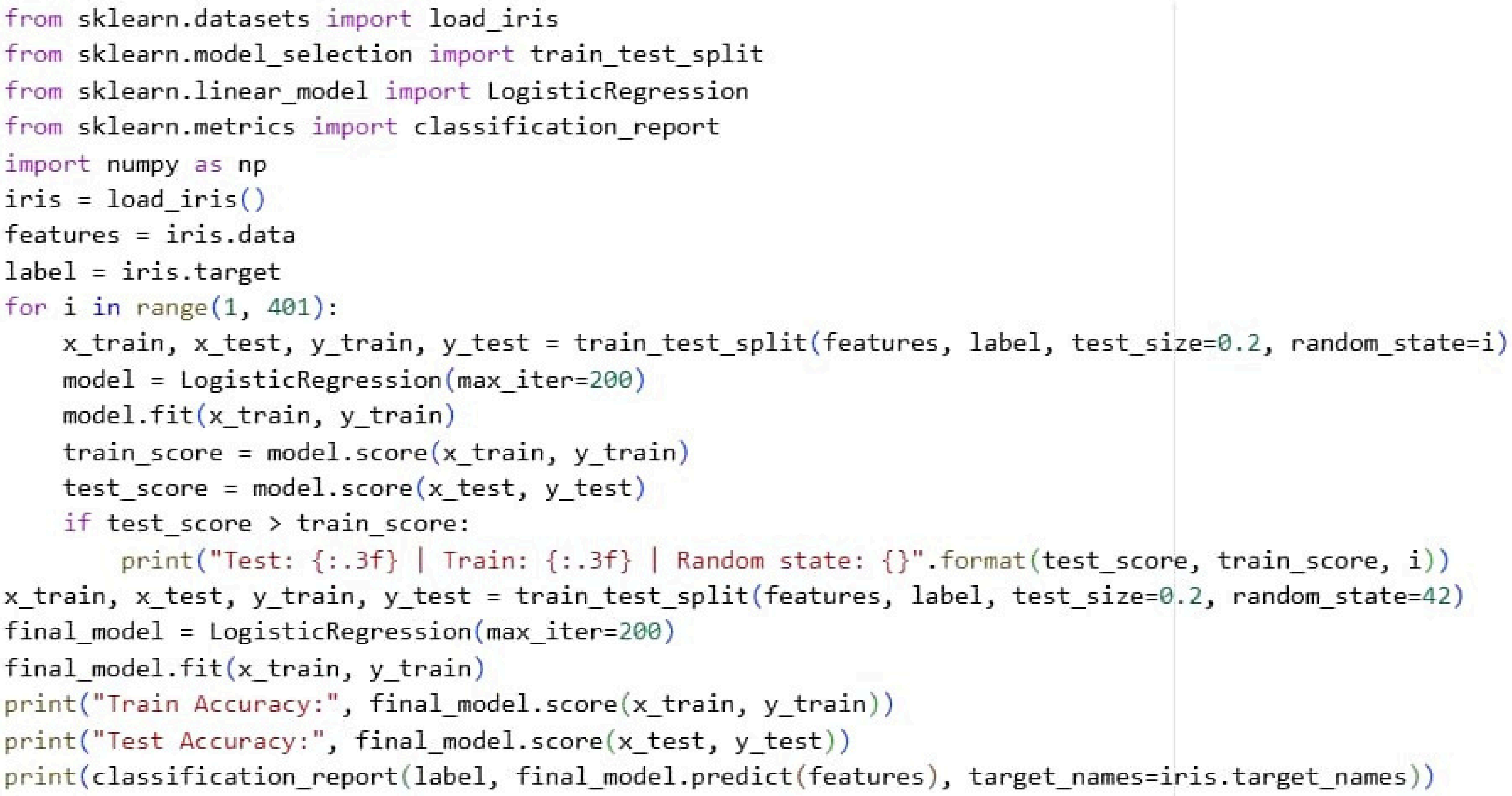

Top 10 Feature Importances
petal length (cm)
petal width (cm)
sepal length (cm)
sepal width (cm)
Feature
0.0
0.1
0.2
0.3
0.4
Importance



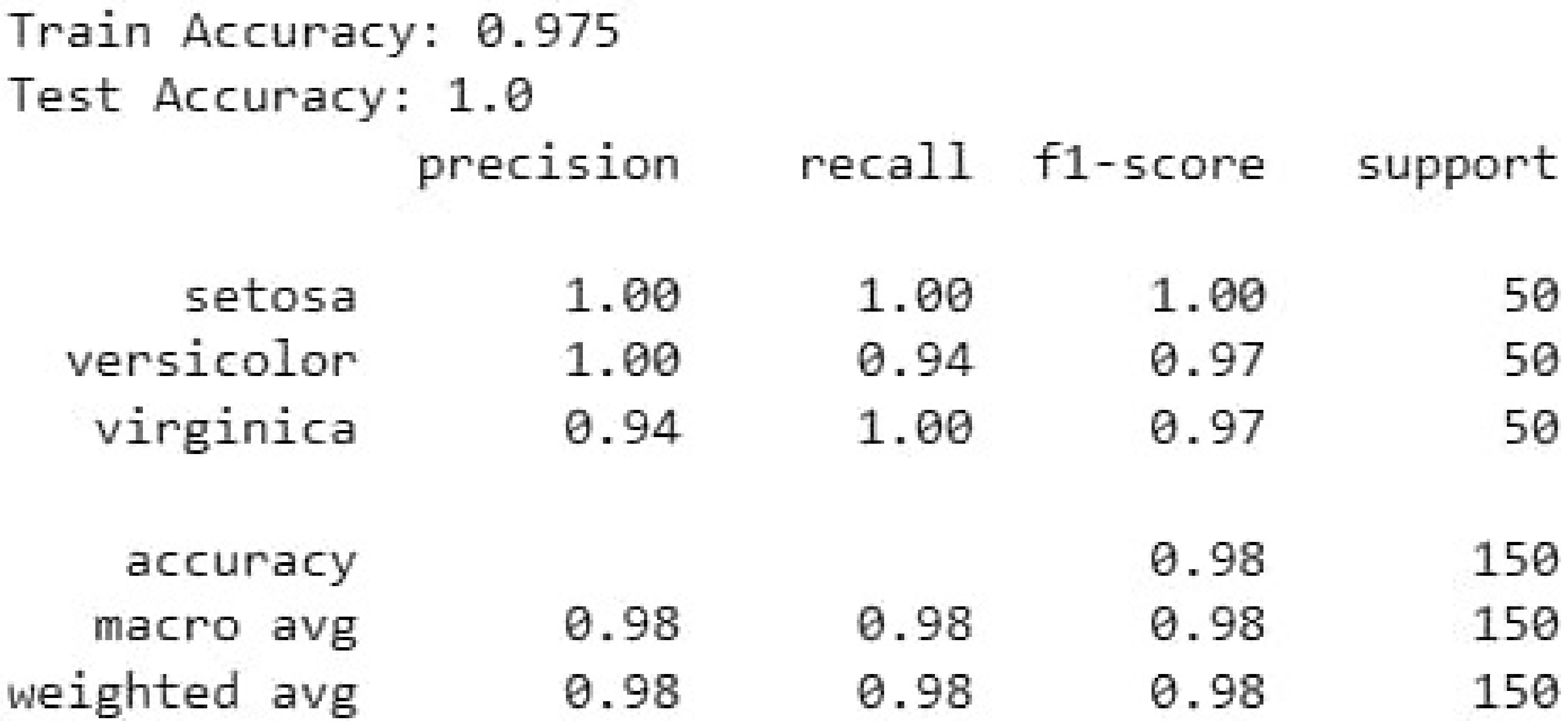
```python
import numpy as np
import pandas as pd
df = pd.read_csv('Salary_data.csv')
df.dropna(inplace=True)
print(df.info())
print(df.describe())
features = df.iloc[:, [0]].values
label = df.iloc[:, [1]].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    features, label, test_size=0.2, random_state=42
)
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, y_train)
print("Training Score:", model.score(x_train, y_train))
print("Testing Score:", model.score(x_test, y_test))
print("Coefficient:", model.coef_)
print("Intercept:", model.intercept_)
import pickle
pickle.dump(model, open('SalaryPred.model', 'wb'))
model = pickle.load(open('SalaryPred.model', 'rb'))
yr_of_exp = float(input("Enter Years of Experience: "))
yr_of_exp_NP = np.array([[yr_of_exp]])
Salary = model.predict(yr_of_exp_NP)
print(f"Estimated Salary for {yr_of_exp} years of experience is: {Salary[0][0]:.2f}")
```



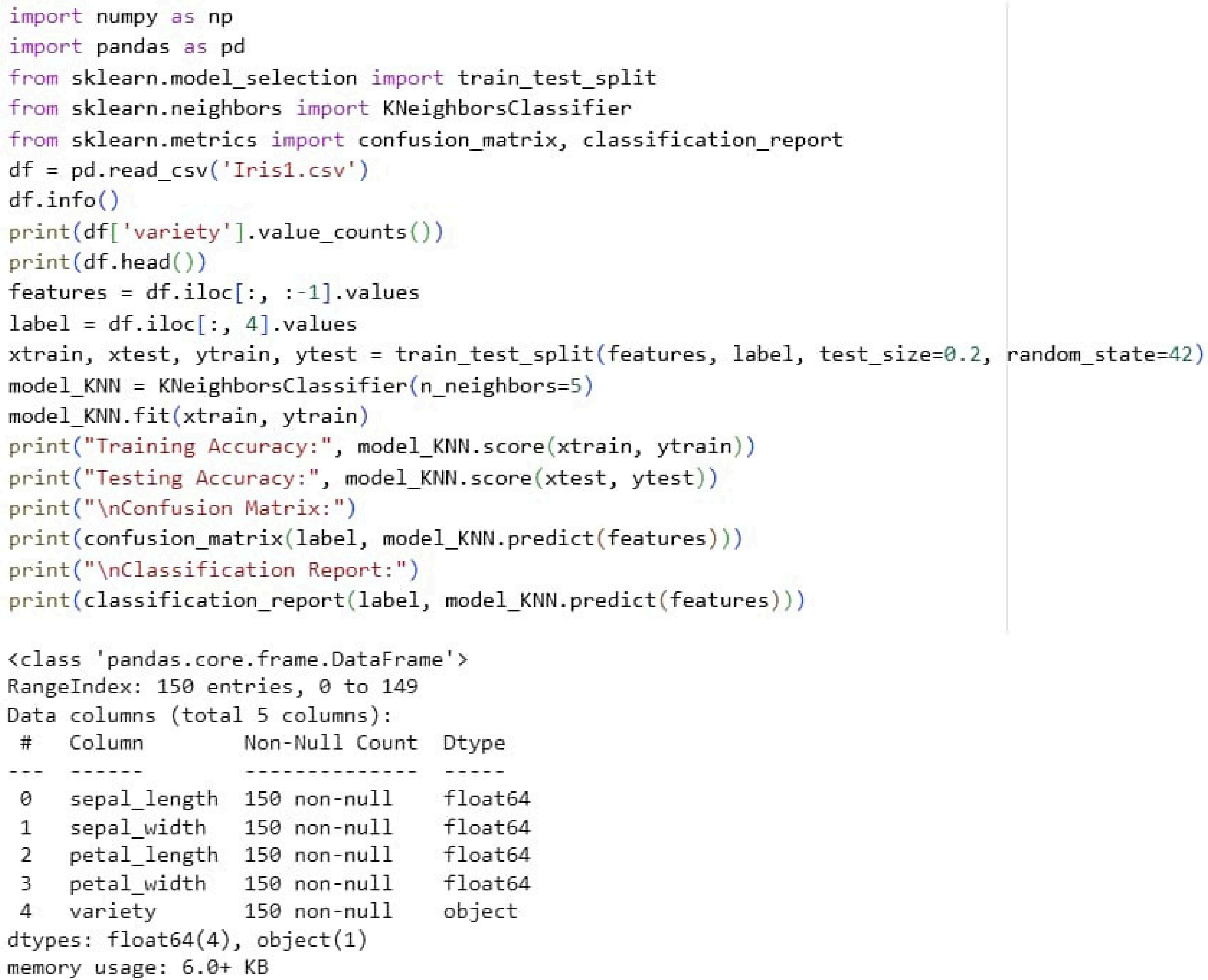
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29 entries, 0 to 28
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  29 non-null     float64
 1   Salary           29 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 596.0 bytes
None
       YearsExperience         Salary
count        29.000000      29.000000
mean          5.358621   76048.931034
std           2.877067   27983.440848
min           1.100000   37731.000000
25%           3.200000   56642.000000
50%           4.900000   63218.000000
75%           7.900000  101302.000000
max          10.500000  122391.000000
Training Score: 0.9393576307207374
Testing Score: 0.9170535247423002
Coefficient: [[9315.01199233]]
Intercept: [25125.45885762]
Enter Years of Experience: 4
Estimated Salary for 4.0 years of experience is: 62385.51
```



```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import numpy as np
iris = load_iris()
features = iris.data
label = iris.target
for i in range(1, 401):
    x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2, random_state=i)
    model = LogisticRegression(max_iter=200)
    model.fit(x_train, y_train)
    train_score = model.score(x_train, y_train)
    test_score = model.score(x_test, y_test)
    if test_score > train_score:
        print("Test: {:.3f} | Train: {:.3f} | Random state: {}".format(test_score, train_score, i))
x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2, random_state=42)
final_model = LogisticRegression(max_iter=200)
final_model.fit(x_train, y_train)
print("Train Accuracy:", final_model.score(x_train, y_train))
print("Test Accuracy:", final_model.score(x_test, y_test))
print(classification_report(label, final_model.predict(features), target_names=iris.target_names))
```

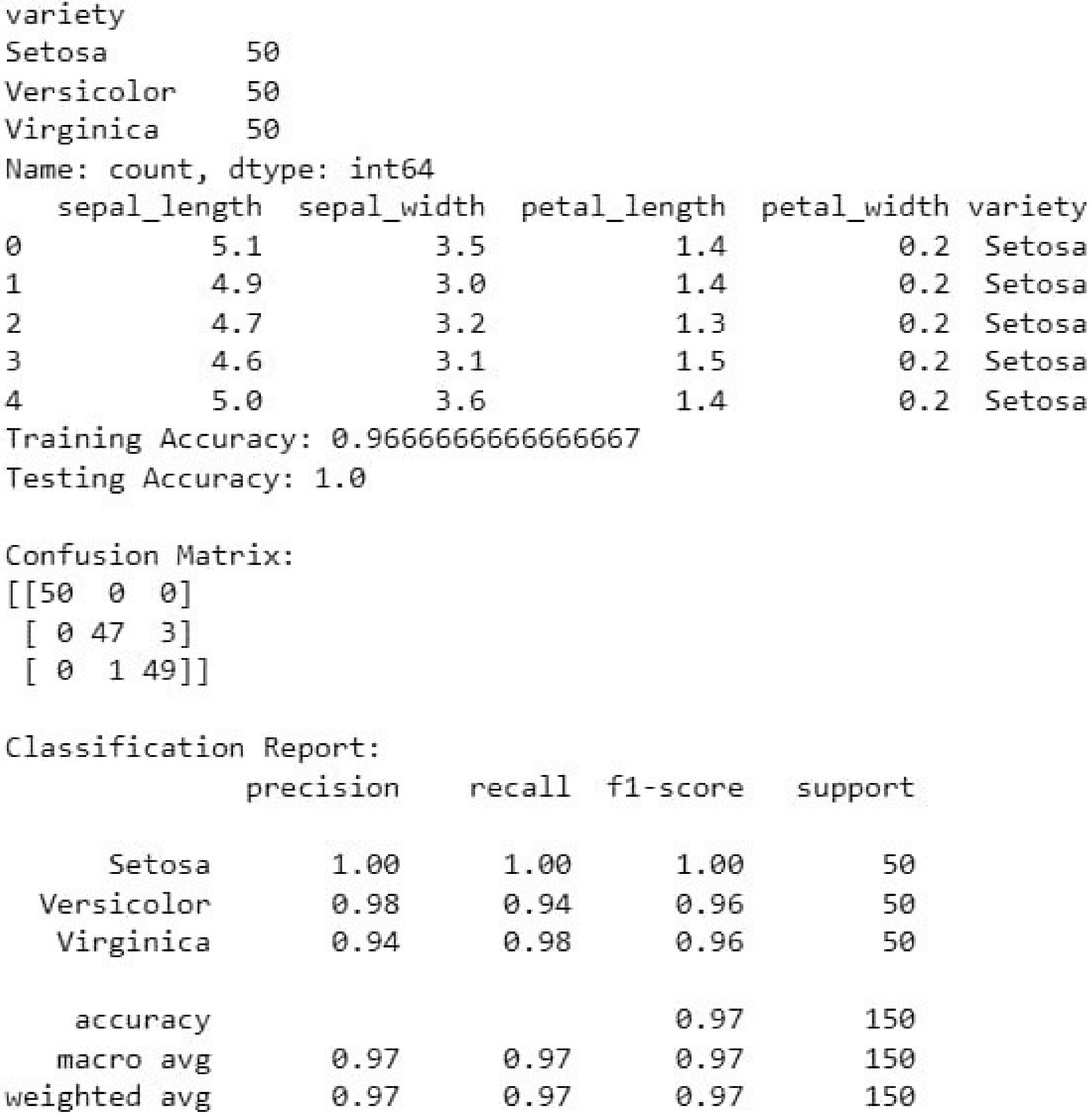


```
Train Accuracy: 0.975
Test Accuracy: 1.0
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        50
  versicolor       1.00      0.94      0.97        50
   virginica       0.94      1.00      0.97        50

    accuracy                           0.98       150
   macro avg       0.98      0.98      0.98       150
weighted avg       0.98      0.98      0.98       150
```

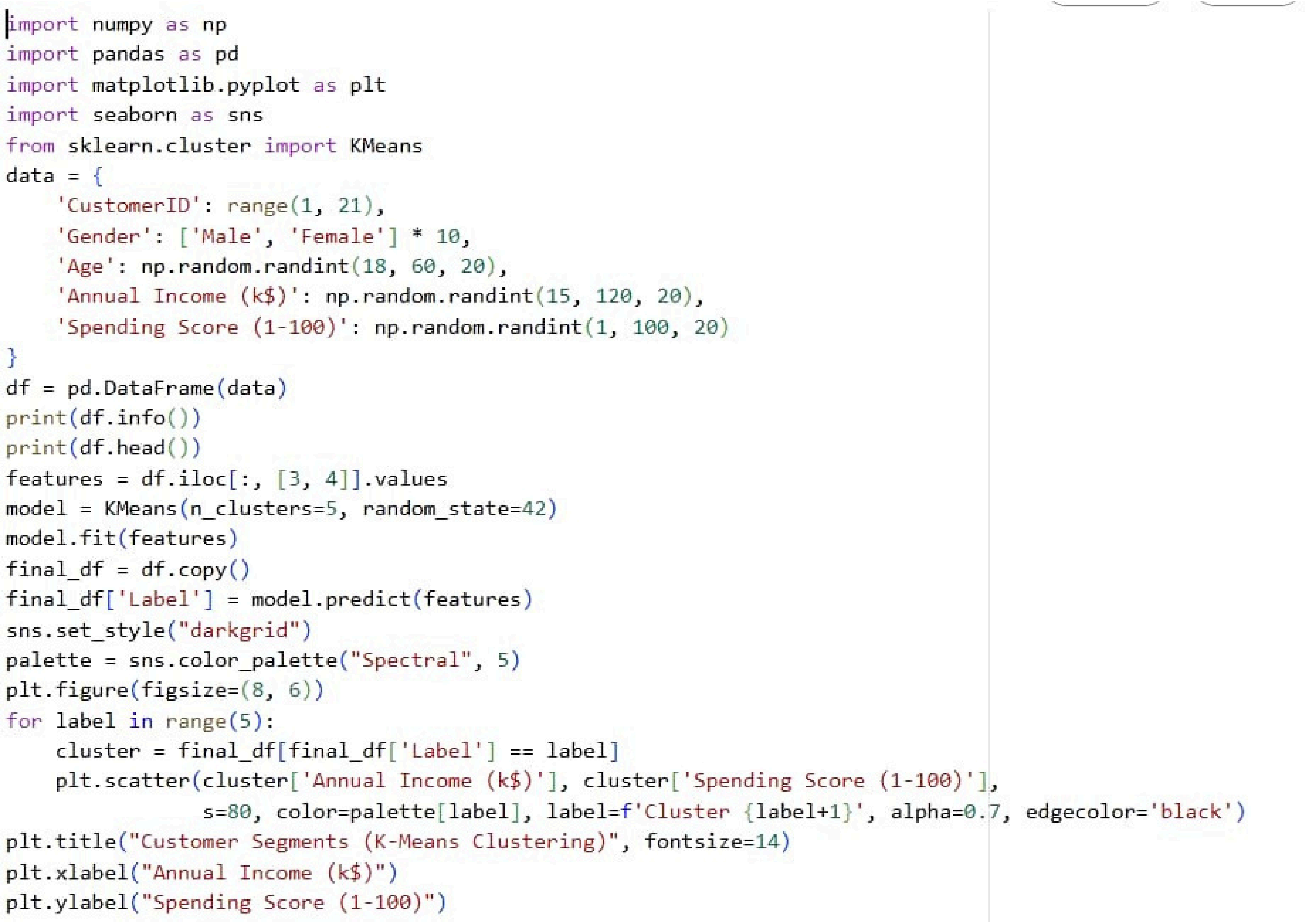


```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report
df = pd.read_csv('Iris1.csv')
df.info()
print(df['variety'].value_counts())
print(df.head())
features = df.iloc[:, :-1].values
label = df.iloc[:, 4].values
xtrain, xtest, ytrain, ytest = train_test_split(features, label, test_size=0.2, random_state=42)
model_KNN = KNeighborsClassifier(n_neighbors=5)
model_KNN.fit(xtrain, ytrain)
print("Training Accuracy:", model_KNN.score(xtrain, ytrain))
print("Testing Accuracy:", model_KNN.score(xtest, ytest))
print("\nConfusion Matrix:")
print(confusion_matrix(label, model_KNN.predict(features)))
print("\nClassification Report:")
print(classification_report(label, model_KNN.predict(features)))
```

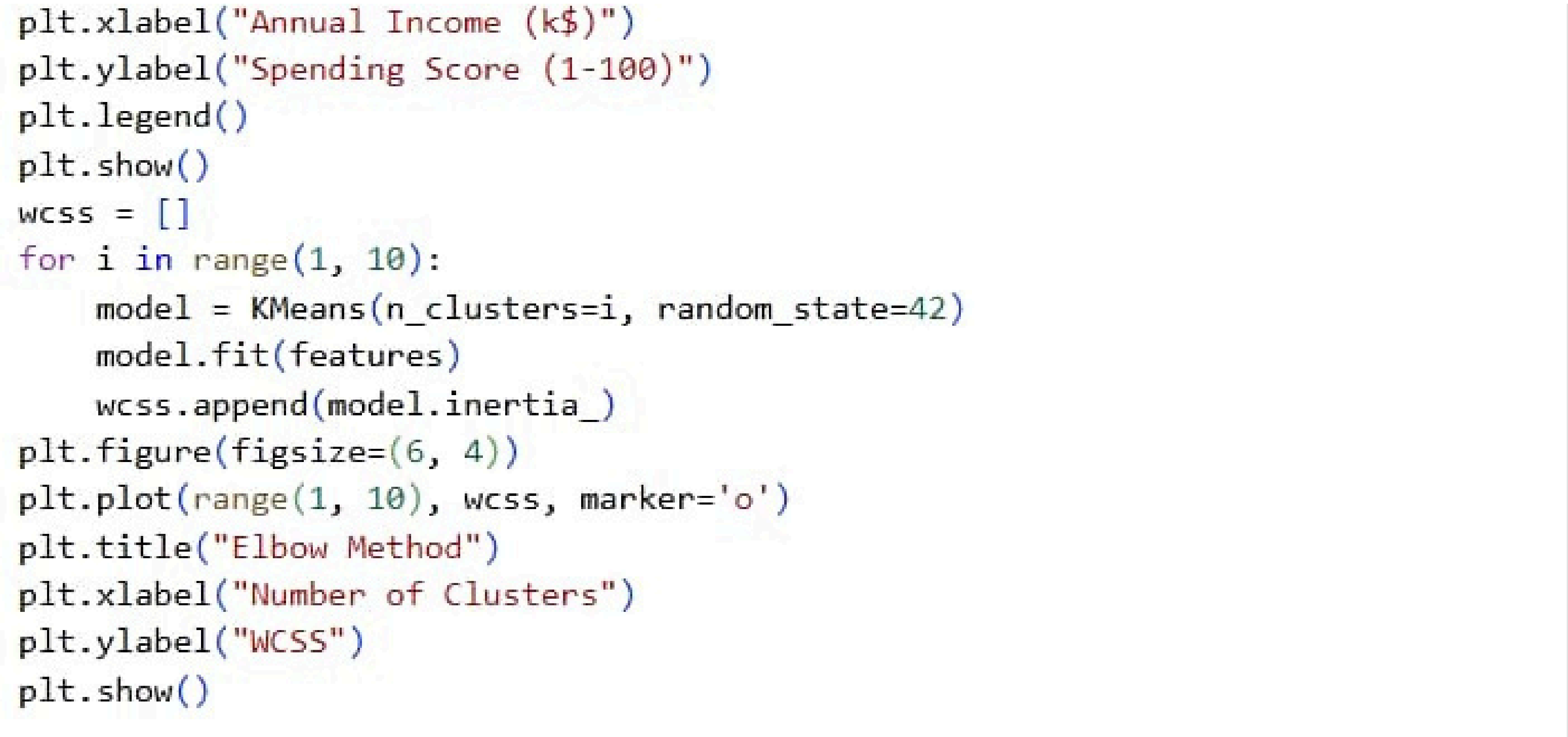
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   variety       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```



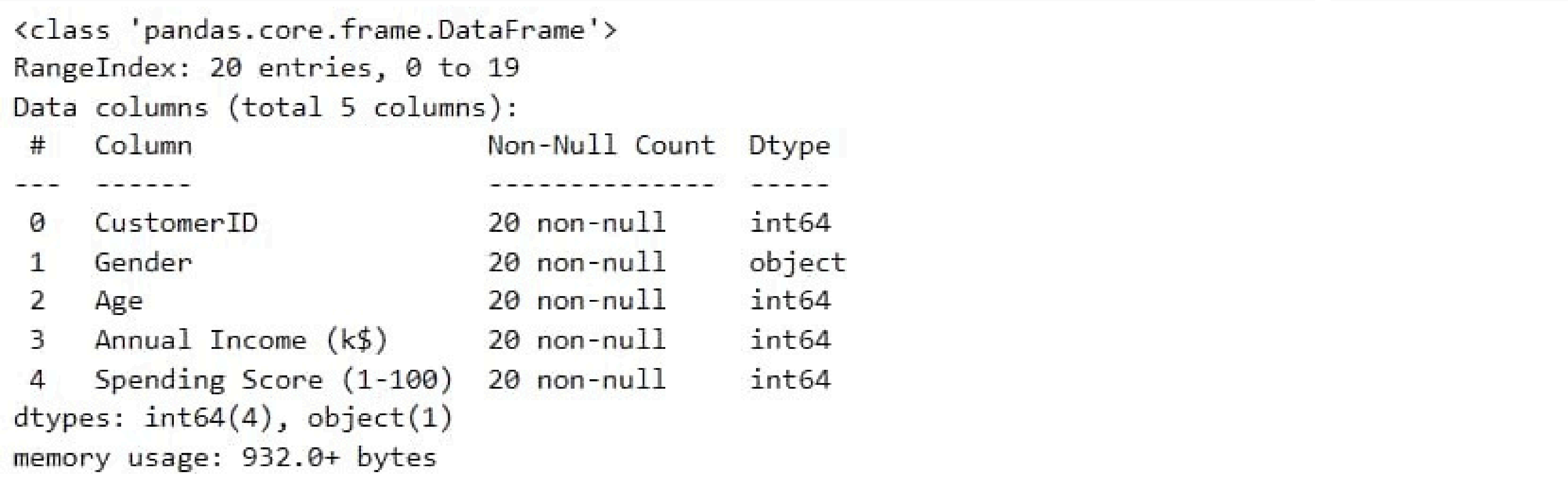
```
variety
Setosa        50
Versicolor    50
Virginica     50
Name: count, dtype: int64
   sepal_length  sepal_width  petal_length  petal_width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
Training Accuracy: 0.9666666666666667
Testing Accuracy: 1.0

Confusion Matrix:
[[50  0  0]
 [ 0 47  3]
 [ 0  1 49]]

Classification Report:
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        50
  Versicolor       0.98      0.94      0.96        50
   Virginica       0.94      0.98      0.96        50

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150
```



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
data = {
    'CustomerID': range(1, 21),
    'Gender': ['Male', 'Female'] * 10,
    'Age': np.random.randint(18, 60, 20),
    'Annual Income (k$)': np.random.randint(15, 120, 20),
    'Spending Score (1-100)': np.random.randint(1, 100, 20)
}
df = pd.DataFrame(data)
print(df.info())
print(df.head())
features = df.iloc[:, [3, 4]].values
model = KMeans(n_clusters=5, random_state=42)
model.fit(features)
final_df = df.copy()
final_df['Label'] = model.predict(features)
sns.set_style("darkgrid")
palette = sns.color_palette("Spectral", 5)
plt.figure(figsize=(8, 6))
for label in range(5):
    cluster = final_df[final_df['Label'] == label]
    plt.scatter(cluster['Annual Income (k$)'], cluster['Spending Score (1-100)'],
                s=80, color=palette[label], label=f'Cluster {label+1}', alpha=0.7, edgecolor='black')
plt.title("Customer Segments (K-Means Clustering)", fontsize=14)
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
```

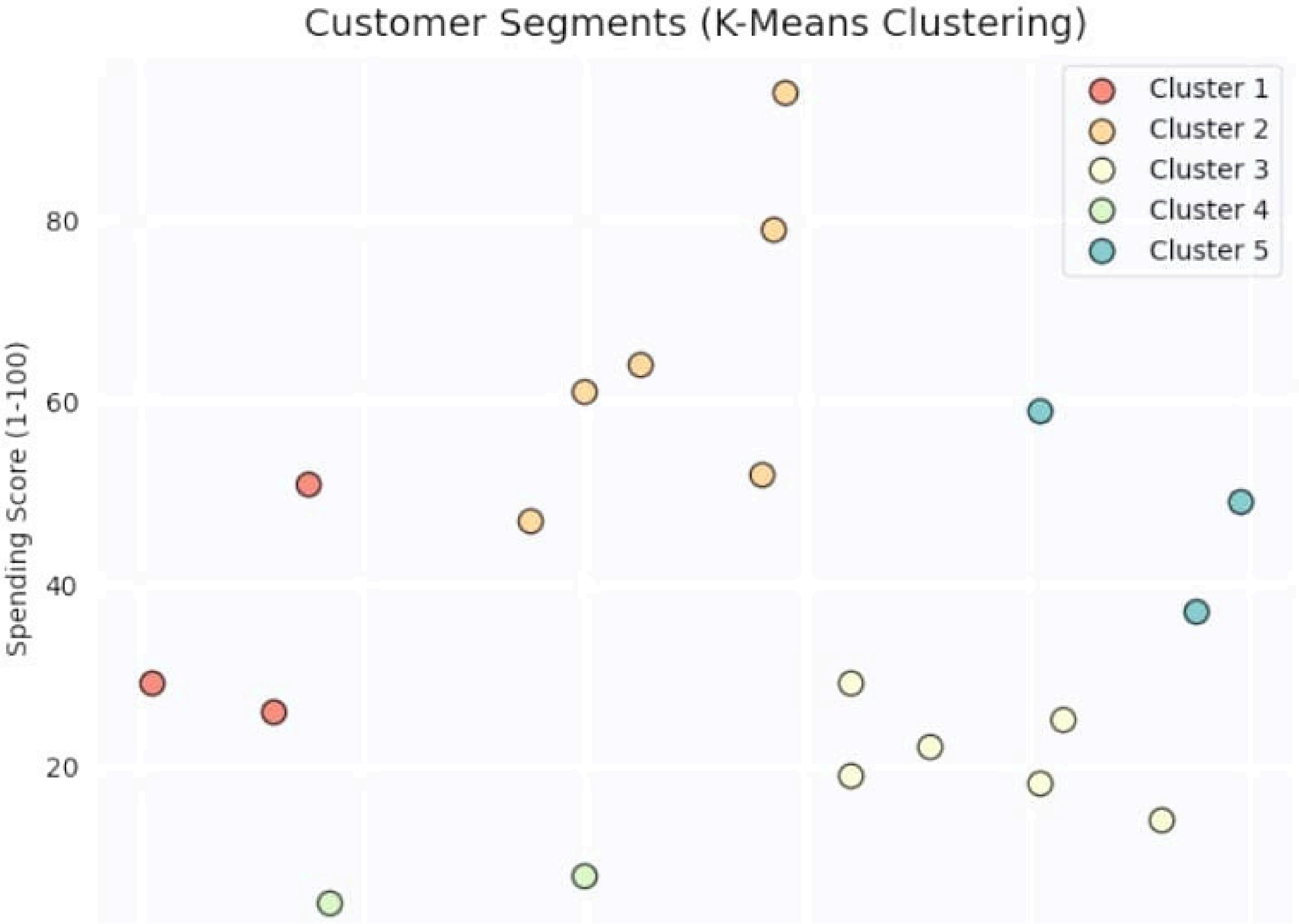


```
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
wcss = []
for i in range(1, 10):
    model = KMeans(n_clusters=i, random_state=42)
    model.fit(features)
    wcss.append(model.inertia_)
plt.figure(figsize=(6, 4))
plt.plot(range(1, 10), wcss, marker='o')
plt.title("Elbow Method")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```

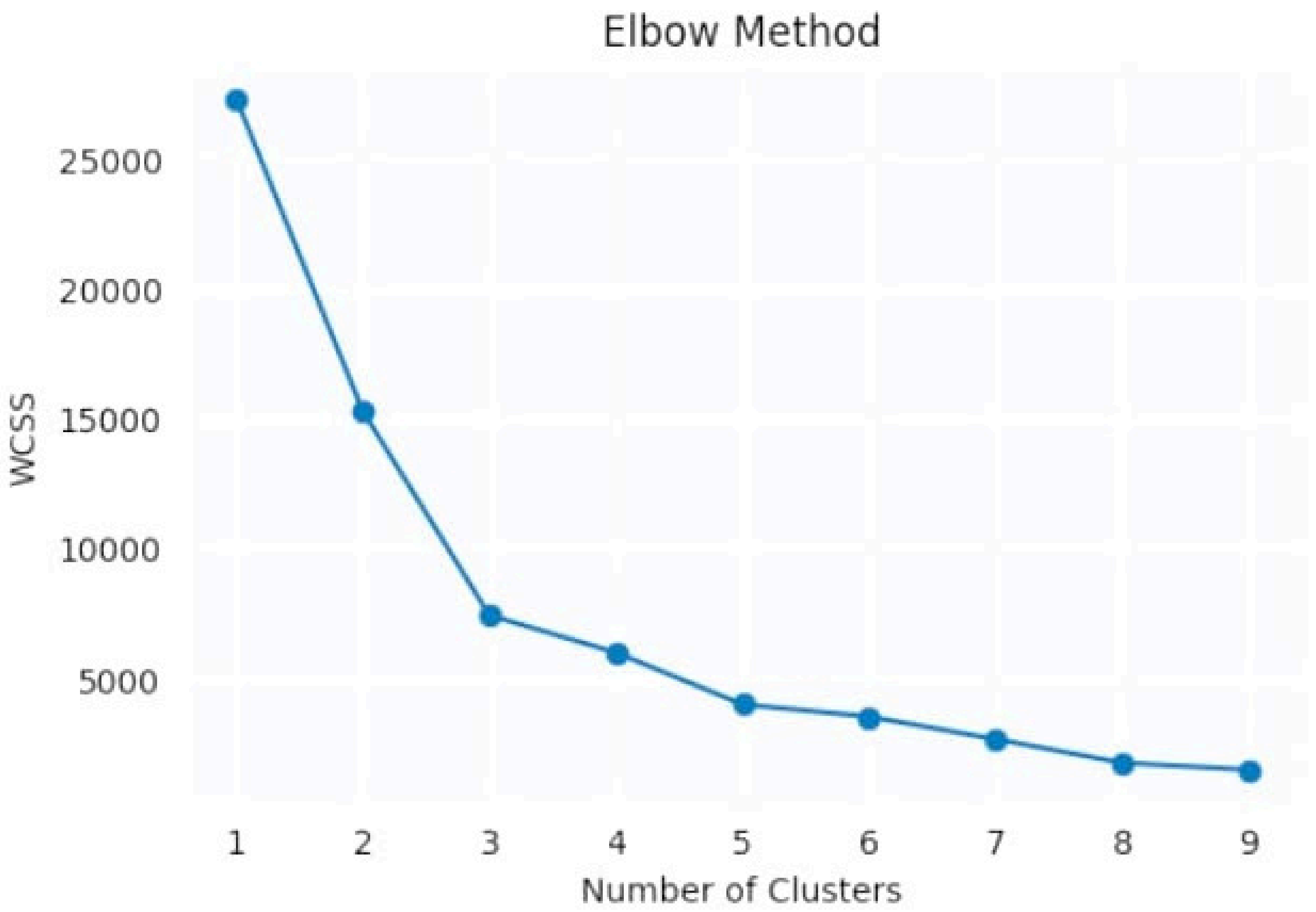


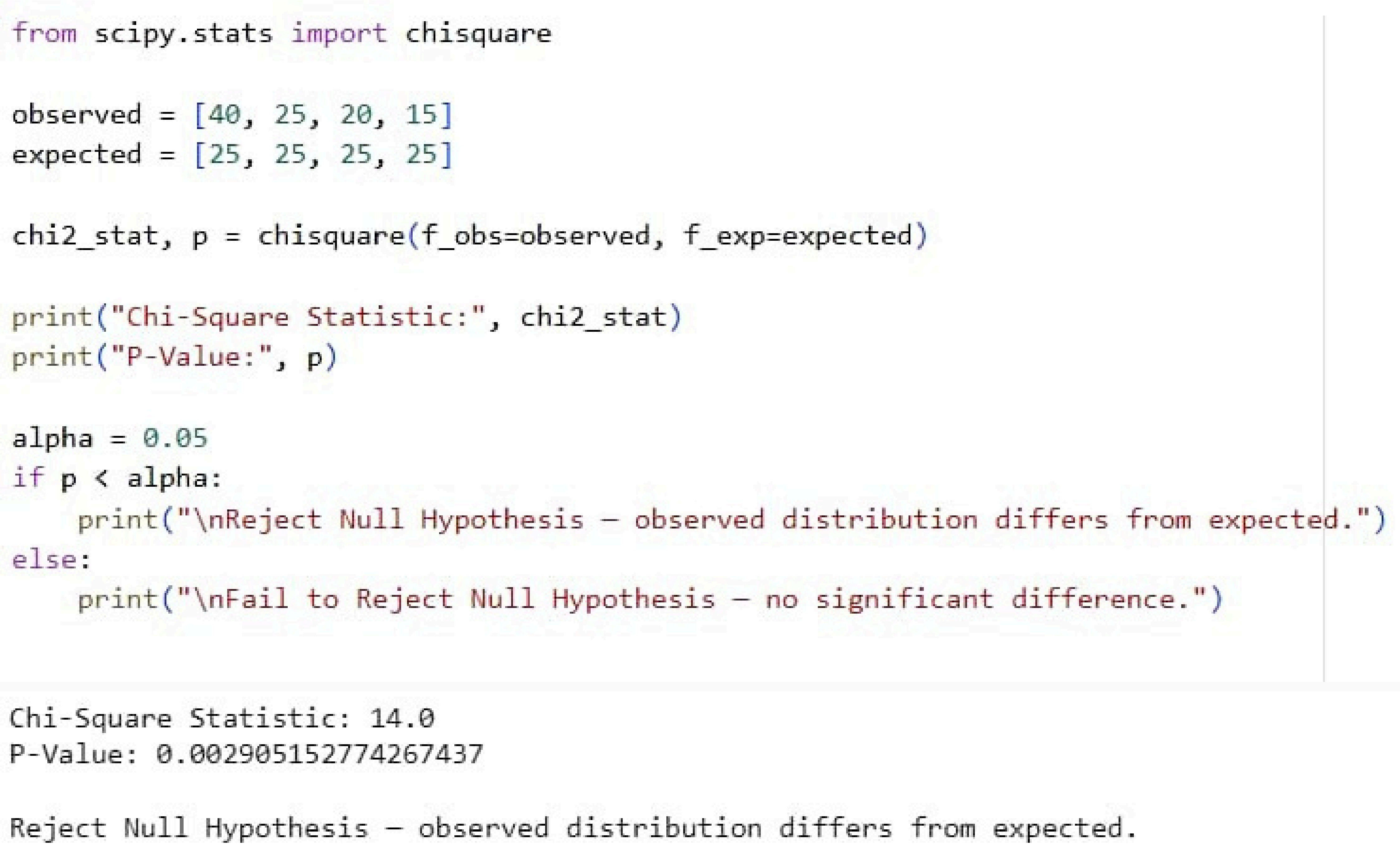
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              20 non-null     int64 
 1   Gender                  20 non-null     object
 2   Age                     20 non-null     int64 
 3   Annual Income (k$)      20 non-null     int64 
 4   Spending Score (1-100)  20 non-null     int64 
dtypes: int64(4), object(1)
memory usage: 932.0+ bytes
```

```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   24                  91                      22
1           2  Female   39                  84                      19
2           3    Male   58                 101                      59
3           4  Female   34                  84                      29
4           5    Male   37                  32                      26
```



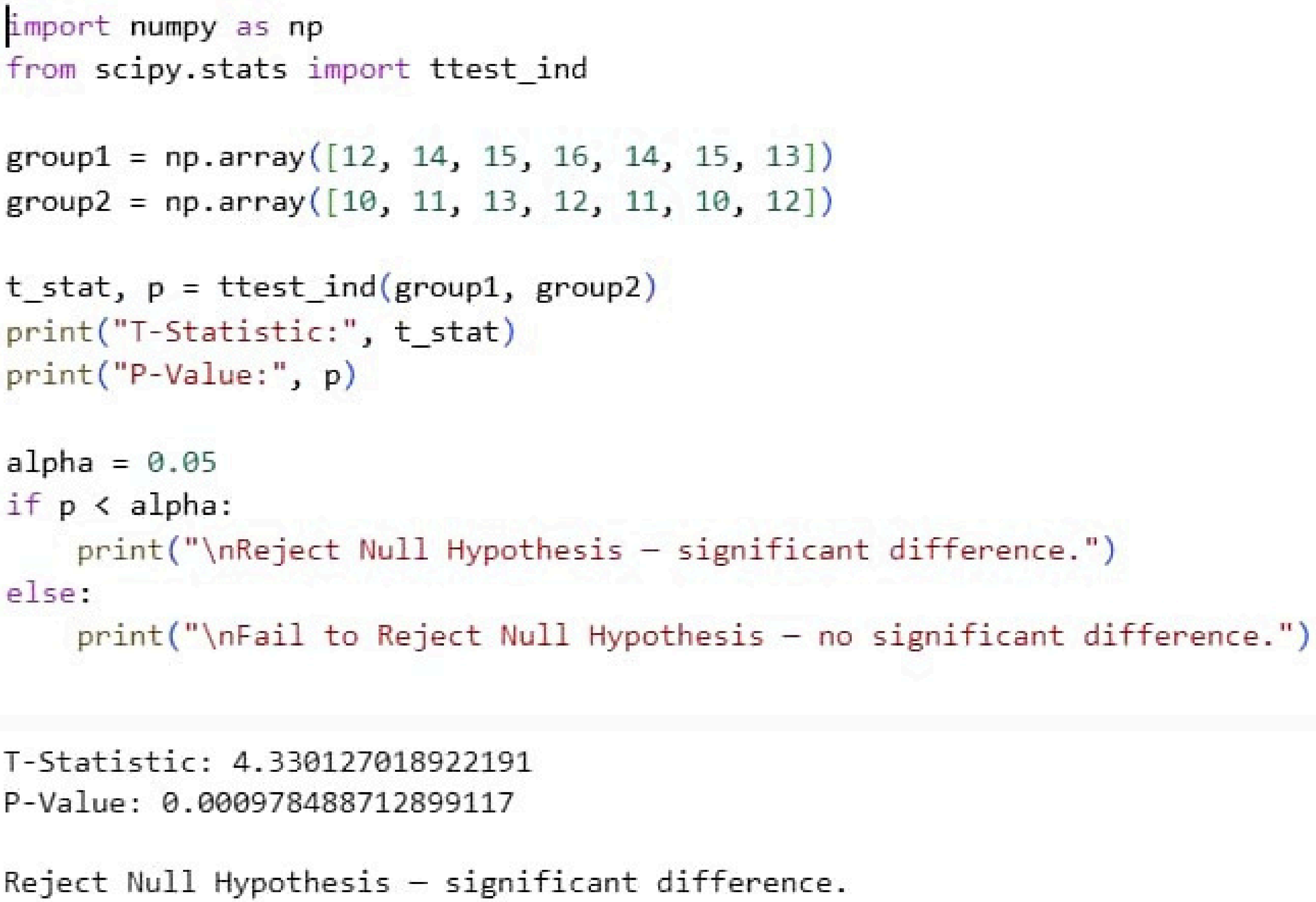
20
40
60
80
100
120
Annual Income (k$)


Elbow Method
WCSS
25000
20000
15000
10000
5000
1
2
3
4
5
6
7
8
9
Number of Clusters



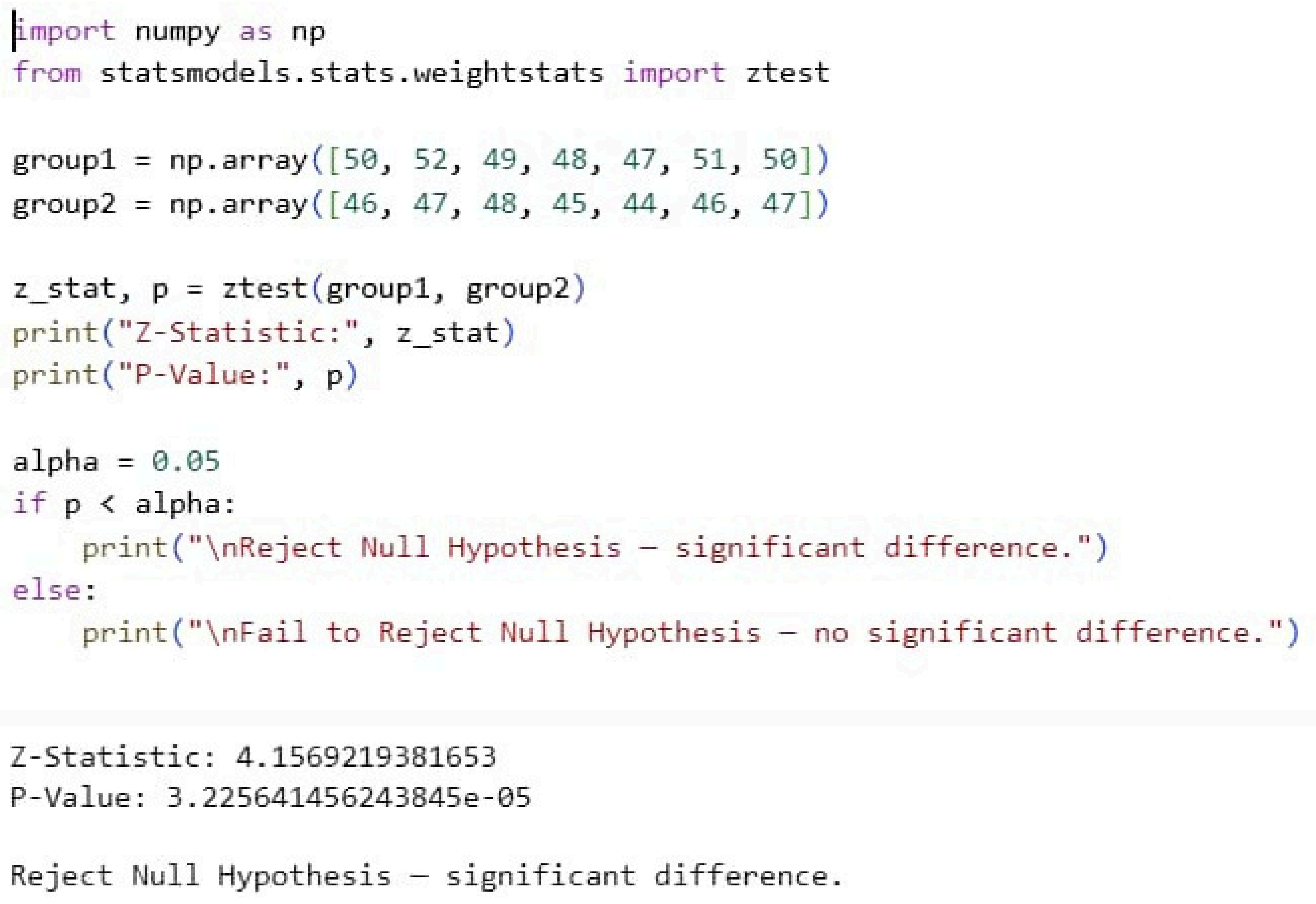
```python
from scipy.stats import chisquare

observed = [40, 25, 20, 15]
expected = [25, 25, 25, 25]

chi2_stat, p = chisquare(f_obs=observed, f_exp=expected)

print("Chi-Square Statistic:", chi2_stat)
print("P-Value:", p)

alpha = 0.05
if p < alpha:
    print("\nReject Null Hypothesis – observed distribution differs from expected.")
else:
    print("\nFail to Reject Null Hypothesis – no significant difference.")
```

```
Chi-Square Statistic: 14.0
P-Value: 0.002905152774267437

Reject Null Hypothesis – observed distribution differs from expected.
```



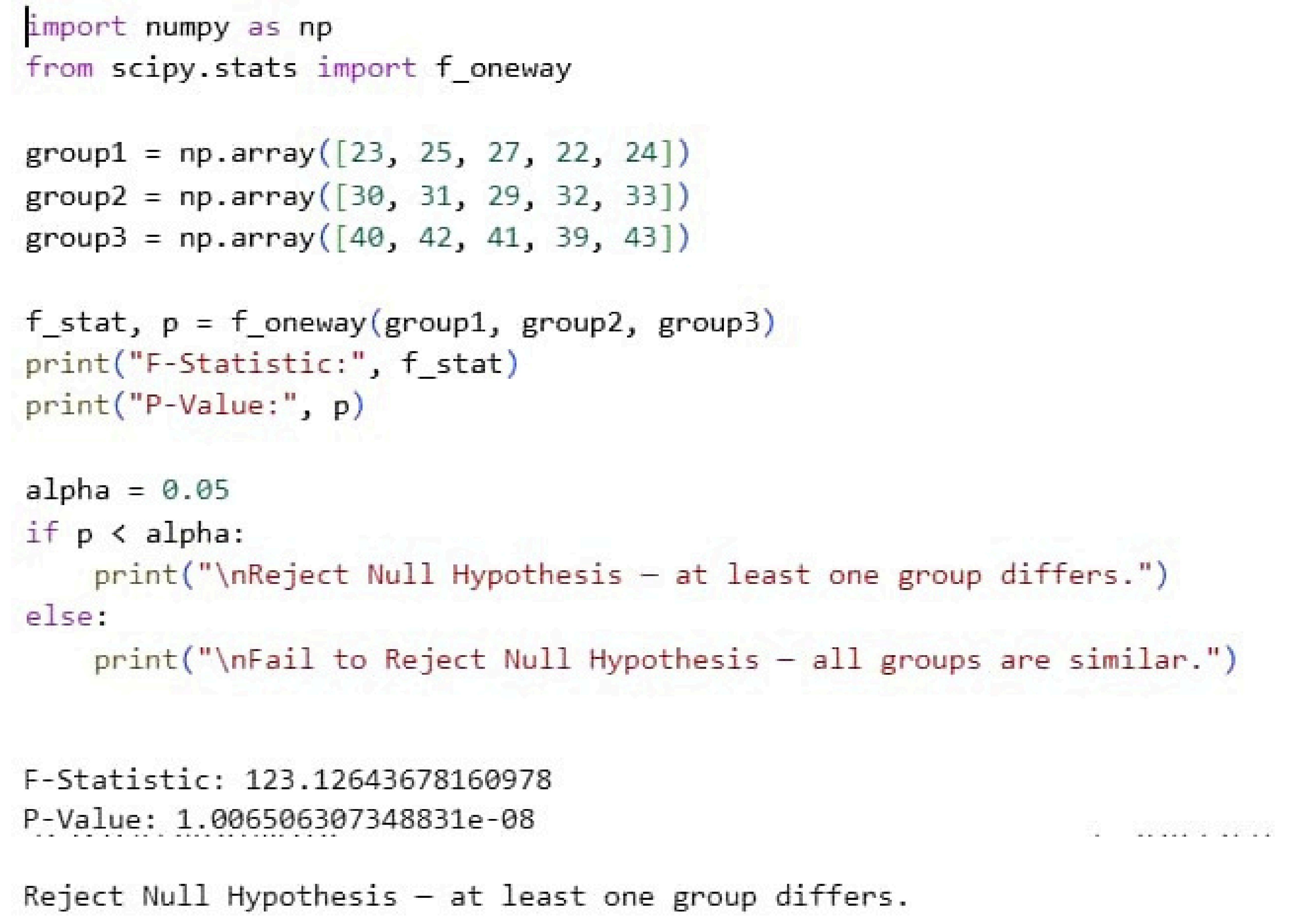
```python
import numpy as np
from scipy.stats import ttest_ind

group1 = np.array([12, 14, 15, 16, 14, 15, 13])
group2 = np.array([10, 11, 13, 12, 11, 10, 12])

t_stat, p = ttest_ind(group1, group2)
print("T-Statistic:", t_stat)
print("P-Value:", p)

alpha = 0.05
if p < alpha:
    print("\nReject Null Hypothesis – significant difference.")
else:
    print("\nFail to Reject Null Hypothesis – no significant difference.")
```

```
T-Statistic: 4.330127018922191
P-Value: 0.000978488712899117

Reject Null Hypothesis – significant difference.
```



```python
import numpy as np
from statsmodels.stats.weightstats import ztest

group1 = np.array([50, 52, 49, 48, 47, 51, 50])
group2 = np.array([46, 47, 48, 45, 44, 46, 47])

z_stat, p = ztest(group1, group2)
print("Z-Statistic:", z_stat)
print("P-Value:", p)

alpha = 0.05
if p < alpha:
    print("\nReject Null Hypothesis – significant difference.")
else:
    print("\nFail to Reject Null Hypothesis – no significant difference.")
```

```
Z-Statistic: 4.1569219381653
P-Value: 3.225641456243845e-05

Reject Null Hypothesis – significant difference.
```



```python
import numpy as np
from scipy.stats import f_oneway

group1 = np.array([23, 25, 27, 22, 24])
group2 = np.array([30, 31, 29, 32, 33])
group3 = np.array([40, 42, 41, 39, 43])

f_stat, p = f_oneway(group1, group2, group3)
print("F-Statistic:", f_stat)
print("P-Value:", p)

alpha = 0.05
if p < alpha:
    print("\nReject Null Hypothesis – at least one group differs.")
else:
    print("\nFail to Reject Null Hypothesis – all groups are similar.")
```

```
F-Statistic: 123.12643678160978
P-Value: 1.006506307348831e-08

Reject Null Hypothesis – at least one group differs.
```