```
pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.12/dist-packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2025.1
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrep
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrepo
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlre
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->panda
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
abalone = fetch_ucirepo(id=1)

# data (as pandas dataframes)
X = abalone.data.features
y = abalone.data.targets

# metadata
print(abalone.metadata)

# variable information
print(abalone.variables)
```

```
{'uci_id': 1, 'name': 'Abalone', 'repository_url': 'https://archive.ics.uci.edu/dataset/1/abalone', 'data_url': 'http
             name     role         type demographic  \
0             Sex  Feature  Categorical        None
1          Length  Feature   Continuous        None
2        Diameter  Feature   Continuous        None
3          Height  Feature   Continuous        None
4    Whole_weight  Feature   Continuous        None
5  Shucked_weight  Feature   Continuous        None
6  Viscera_weight  Feature   Continuous        None
7    Shell_weight  Feature   Continuous        None
8           Rings   Target      Integer        None

                   description  units missing_values
0         M, F, and I (infant)   None             no
1    Longest shell measurement     mm             no
2      perpendicular to length     mm             no
3           with meat in shell     mm             no
4                whole abalone  grams             no
5               weight of meat  grams             no
6  gut weight (after bleeding)  grams             no
7            after being dried  grams             no
8  +1.5 gives the age in years   None             no
```

```
# Lab 1: Pandas Library
# Heading: Data Frame Operations with Pandas

import pandas as pd
from ucimlrepo import fetch_ucirepo

# Fetch Abalone dataset
abalone = fetch_ucirepo(id=1)

# Convert to DataFrame
X = abalone.data.features
y = abalone.data.targets

# Display first few rows
print(X.head())
print(y.head())
```

```
  Sex  Length  Diameter  Height  Whole_weight  Shucked_weight  Viscera_weight  \
0   M   0.455     0.365   0.095        0.5140          0.2245          0.1010
1   M   0.350     0.265   0.090        0.2255          0.0995          0.0485
2   F   0.530     0.420   0.135        0.6770          0.2565          0.1415
3   M   0.440     0.365   0.125        0.5160          0.2155          0.1140
4   I   0.330     0.255   0.080        0.2050          0.0895          0.0395

   Shell_weight
0         0.150
1         0.070
2         0.210
3         0.155
4         0.055
   Rings
```

```
0     15
1      7
2      9
3     10
4      7
```

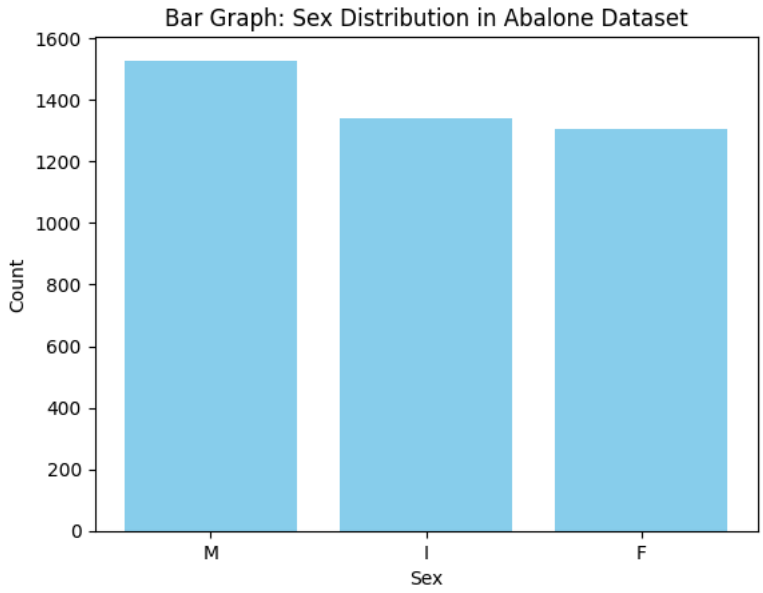
```
# Lab 1: Numpy Library
# Heading: Numpy Array Operations

import numpy as np

# Convert Pandas DataFrame to Numpy Array
X_array = X.values

# Display array shape and first row
print(f"Shape of X: {X_array.shape}")
print("First row of X:", X_array[0])
```

```
Shape of X: (4177, 8)
First row of X: ['M' 0.455 0.365 0.095 0.514 0.2245 0.101 0.15]
```

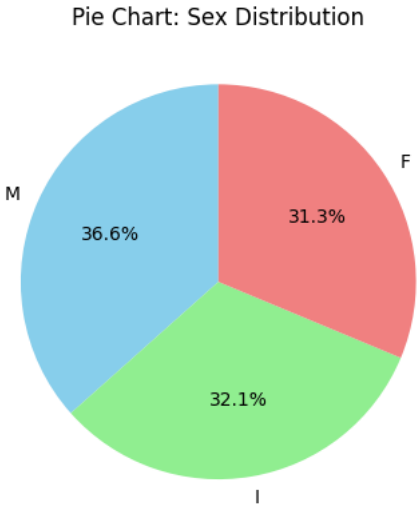
```
# Lab 1: Scikit Learn Library
# Heading: Data Preparation with Scikit Learn

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Identify categorical and numerical columns
categorical_features = ['Sex']
numerical_features = X.select_dtypes(include=np.number).columns.tolist()

# Create a column transformer for one-hot encoding and scaling
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ],
    remainder='passthrough' # Keep other columns (if any)
)

# Apply the preprocessing to the training and testing data
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)

print("Processed X_train shape:", X_train_processed.shape)
```

```
Processed X_train shape: (3341, 10)
```

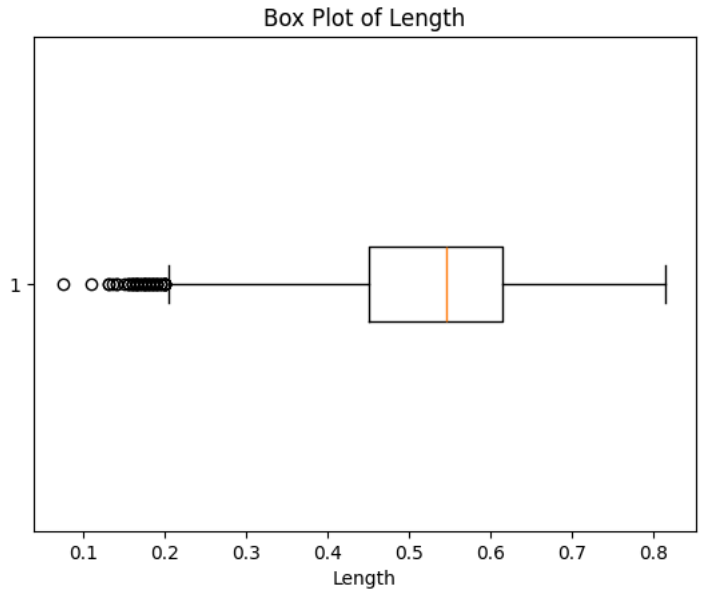
```
# Lab 1: Matplotlib Library
# Bar Graph: Sex Distribution in Abalone Dataset

import matplotlib.pyplot as plt

plt.bar(X['Sex'].value_counts().index, X['Sex'].value_counts().values, color='skyblue')
plt.title('Bar Graph: Sex Distribution in Abalone Dataset')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()
```



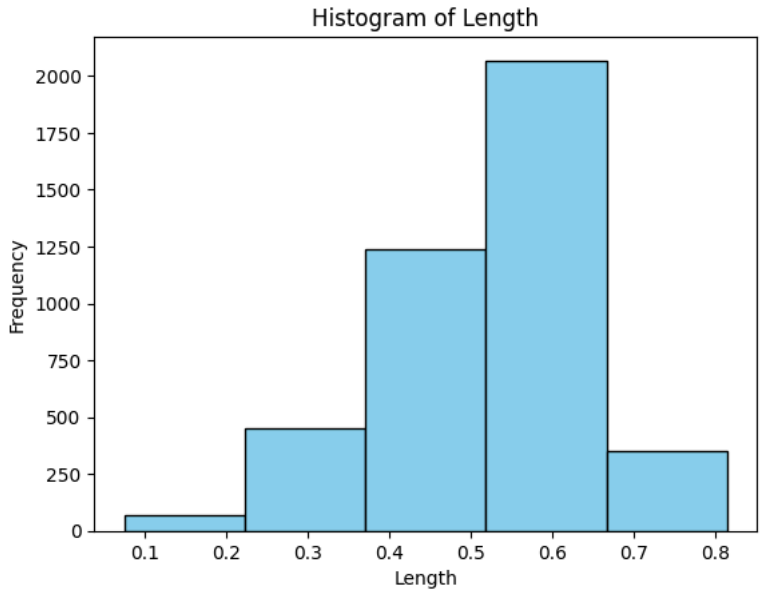

```
# Lab 1: Matplotlib Library
# Pie Chart: Sex Distribution

plt.pie(X['Sex'].value_counts().values, labels=X['Sex'].value_counts().index, autopct='%1.1f%%',
        startangle=90, colors=['skyblue', 'lightgreen', 'lightcoral'])
plt.title('Pie Chart: Sex Distribution')
plt.show()
```



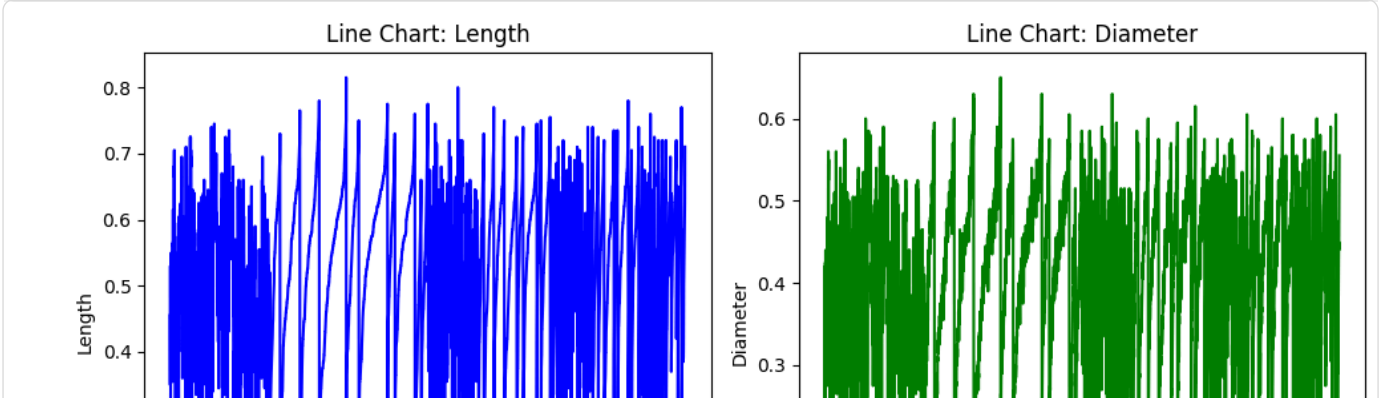

```
# Lab 1: Matplotlib Library
# Box Plot: Length

plt.boxplot(X['Length'], vert=False)
plt.title('Box Plot of Length')
plt.xlabel('Length')
plt.show()
```




```
# Lab 1: Matplotlib Library
# Histogram: Length

plt.hist(X['Length'], bins=5, color='skyblue', edgecolor='black')
plt.title('Histogram of Length')
plt.xlabel('Length')
plt.ylabel('Frequency')
plt.show()
```




```
# Lab 1: Matplotlib Library
# Line Chart and Subplots: Length & Diameter

fig, axes = plt.subplots(1, 2, figsize=(10, 5))
axes[0].plot(X['Length'], color='blue')
axes[0].set_title('Line Chart: Length')
axes[0].set_xlabel('Index')
axes[0].set_ylabel('Length')

axes[1].plot(X['Diameter'], color='green')
axes[1].set_title('Line Chart: Diameter')
axes[1].set_xlabel('Index')
axes[1].set_ylabel('Diameter')

plt.tight_layout()
plt.show()
```


Line Chart: Length
Length
0.8
0.7
0.6
0.5
0.4
Line Chart: Diameter
Diameter
0.6
0.5
0.4
0.3