Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. It assumes that there is a linear relationship between the independent variables and the dependent variable. The linear regression equation can be represented as:

$$y=\beta_0+\beta_1x_1+\beta_2x_2+\ldots+\beta_nx_n+\epsilon$$

y is the dependent variable (the variable we want to predict).

$x_1,x_2,\ldots,x_n$ are the independent variables (the predictors).

β_0 is the intercept (the value of y when all independent variables are 0).

$\beta_1,\beta_2,\ldots,\beta_n$ are the coefficients (the weights) representing the change in y for a one-unit change in each independent variable, holding other variables constant.

ϵ is the error term, representing the difference between the predicted and actual values of y.

The goal of linear regression is to estimate the coefficients (0,1,...,$\beta_0,\beta_1,\ldots,\beta_n$)that minimize the sum of squared differences between the observed and predicted values of the dependent variable.

Here's a practical example of linear regression using Python:

```
python
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Generating example data
np.random.seed(0)
X = 2 * np.random.rand(100, 1)  # Independent variable
y = 3 + 4 * X + np.random.randn(100, 1)  # Dependent variable

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Creating and fitting the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Making predictions
y_pred = model.predict(X_test)

# Calculating coefficients and intercept
coefficients = model.coef_
intercept = model.intercept_

# Calculating mean squared error
mse = mean_squared_error(y_test, y_pred)

# Plotting the data and regression line
plt.scatter(X_test, y_test, color='black')
plt.plot(X_test, y_pred, color='blue', linewidth=3)
plt.title('Linear Regression')
plt.xlabel('Independent variable')
plt.ylabel('Dependent variable')
plt.show()

# Printing coefficients, intercept, and mean squared error
print("Coefficients:", coefficients)
print("Intercept:", intercept)
print("Mean Squared Error:", mse)
```

In this example:

We generate synthetic data using NumPy, where the dependent variable (y) is a linear function of the independent variable (x) plus some random noise.

We split the data into training and testing sets using train_test_split from scikit-learn.

We create an instance of the LinearRegression class, fit the model to the training data, and make predictions on the testing data.

We calculate the coefficients, intercept, and mean squared error of the model.

Finally, we plot the data points and the regression line to visualize the relationship between the independent and dependent variables.