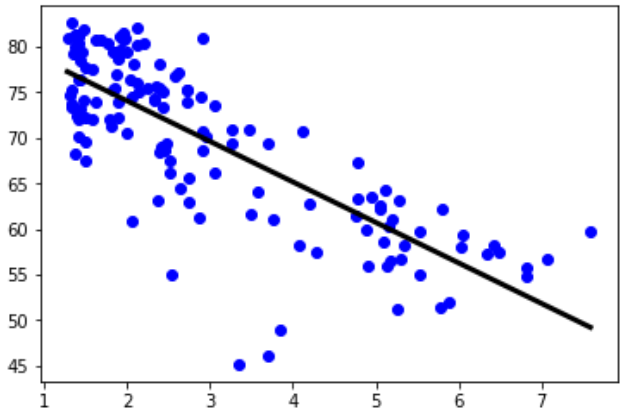
In [39]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Import LinearRegression
from sklearn.linear_model import LinearRegression

# Read the CSV file into a DataFrame: df
df = pd.read_csv('project_data_014.csv')

y = df[['target']].values
X_3 = df[['x3']].values


# Create the regressor: reg
reg = LinearRegression()

# Create the prediction space
prediction_space = np.linspace(min(X_3), max(X_3)).reshape(-1,1)

# Fit the model to the data
reg.fit(X_3, y)

# Compute predictions over the prediction space: y_pred
y_pred = reg.predict(prediction_space)
#y_pred = reg.predict(X_3)

# Print R^2
print("R^2: {}".format(reg.score(X_3, y)))

# Plot regression line
plt.scatter(X_3, y, color='blue')
plt.plot(prediction_space, y_pred, color='black', linewidth=3)
plt.show()
```

```
R^2: 0.6192442167740035
```



In [10]:
```
# Import necessary modules
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

y = df['target'].values

X = df.drop(['x1','x2'], axis=1).values


# Create training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)

# Create the regressor: reg_all
reg_all = LinearRegression()

# Fit the regressor to the training data
reg_all.fit(X_train, y_train)

# Predict on the test data: y_pred
y_pred = reg_all.predict(X_test)

# Compute and print R^2 and RMSE
print("R^2: {}".format(reg_all.score(X_test, y_test)))

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("Root Mean Squared Error: {}".format(rmse))
```

```
R^2: 0.8380468731430135
Root Mean Squared Error: 3.2476010800369455
```

In [20]:
```
#Cross-validation in scikit-learn
from sklearn.model_selection import cross_val_score

reg = LinearRegression()

cv_scores_3 = cross_val_score(reg, X_3, y, cv=3)

print(cv_scores_3)

print("Average 3-Fold CV Score: {}".format(np.mean(cv_scores_3)))
```

```
[0.75397745 0.55557583 0.57886144]
Average 3-Fold CV Score: 0.6294715754653507
```

In [22]:
```
# Compute 5-fold cross-validation scores: cv_scores
cv_scores_5 = cross_val_score(reg, X_3, y, cv=5)

# Print the 5-fold cross-validation scores
print(cv_scores_5)

print("Average 5-Fold CV Score: {}".format(np.mean(cv_scores_5)))
```

```
[0.71001079 0.75007717 0.55271526 0.547501   0.52410561]
Average 5-Fold CV Score: 0.6168819644425119
```

In [25]:
```
# Perform 10-fold CV
cv_scores_10 = cross_val_score(reg, X_3, y, cv=10)

print(cv_scores_10)

print("Average 10-Fold CV Score: {}".format(np.mean(cv_scores_10)))
```

```
[0.54471791 0.75586083 0.83921958 0.6900756  0.33991801 0.53042913
 0.44784016 0.66619475 0.4439451  0.62573667]
Average 10-Fold CV Score: 0.5883937741571185
```

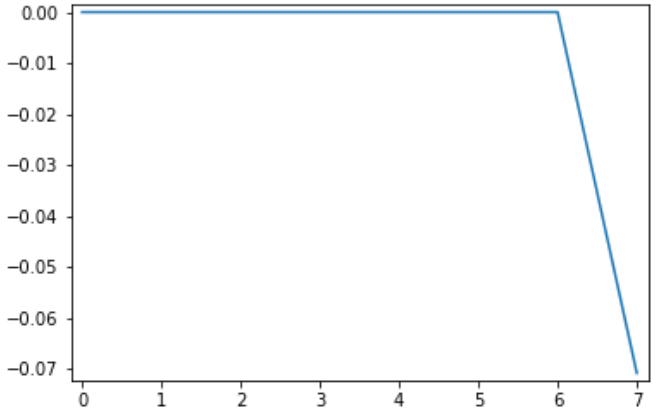
In [29]:
```
#Ridge regression in scikit-learn
from sklearn.linear_model import Ridge

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)

ridge = Ridge(alpha=0.1, normalize=True)

ridge.fit(X_train, y_train)

ridge_pred = ridge.predict(X_test)

print("The score of Ridge Regression: {}".format(ridge.score(X_test, y_test)))
```

```
The score of Ridge Regression: 0.8442469959975749
```

In [37]:

```
#Lasso regression in scikit-learn
from sklearn.linear_model import Lasso

df_columns = df.drop(['x1','x2'], axis=1).columns

# Instantiate a lasso regressor: lasso
lasso = Lasso(alpha=0.4, normalize=True)

# Fit the regressor to the data
lasso = lasso.fit(X, y)

# Compute and print the coefficients
lasso_coef = lasso.coef_


print("The score of Lasso Regression: {}".format(lasso.score(X_test, y_test)))

# Plot the coefficients
plt.plot(range(len(df_columns)), lasso_coef)

#plt.xticks(range(len(df_columns)), df_columns.values, rotation=60)
plt.margins(0.02)

plt.show()
```

```
The score of Lasso Regression: 0.5497472771446612
```



In [40]:

```
def display_plot(cv_scores, cv_scores_std):
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.plot(alpha_space, cv_scores)

    std_error = cv_scores_std / np.sqrt(10)

    ax.fill_between(alpha_space, cv_scores + std_error, cv_scores - std_error, alpha=0.2)
    ax.set_ylabel('CV Score +/- Std Error')
    ax.set_xlabel('Alpha')
    ax.axhline(np.max(cv_scores), linestyle='--', color='.5')
    ax.set_xlim([alpha_space[0], alpha_space[-1]])
    ax.set_xscale('log')
    plt.show()


# Setup the array of alphas and lists to store scores
alpha_space = np.logspace(-4, 0, 50)
ridge_scores = []
ridge_scores_std = []

# Create a ridge regressor: ridge
ridge = Ridge(normalize=True)

# Compute scores over range of alphas
for alpha in alpha_space:

    # Specify the alpha value to use: ridge.alpha
    ridge.alpha = alpha

    # Perform 10-fold CV: ridge_cv_scores
    ridge_cv_scores = cross_val_score(ridge, X, y, cv=10)

    # Append the mean of ridge_cv_scores to ridge_scores
    ridge_scores.append(np.mean(ridge_cv_scores))

    # Append the std of ridge_cv_scores to ridge_scores_std
    ridge_scores_std.append(np.std(ridge_cv_scores))

# Display the plot
display_plot(ridge_scores, ridge_scores_std)
```