In [ ]:

```
import sys
import sklearn
import numpy as np
import os

# Checking versions
# Python ≥ 3.5 required else it will raise assertionError
# Scikit-Learn ≥0.20 is required
assert sys.version_info >= (3, 5)
assert sklearn.__version__ >= "0.20"

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
# The "%matplotlib inline" magic command ensures that any Matplotlib plots will be displayed directly in the output cells of the notebook.
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# Where to save the figures
# This code defines a function for saving Matplotlib figures to a specified directory.
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "training_linear_models"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

In [ ]:

```
# Code is generating 100 random samples from a uniform distribution between 0 and 1
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
```

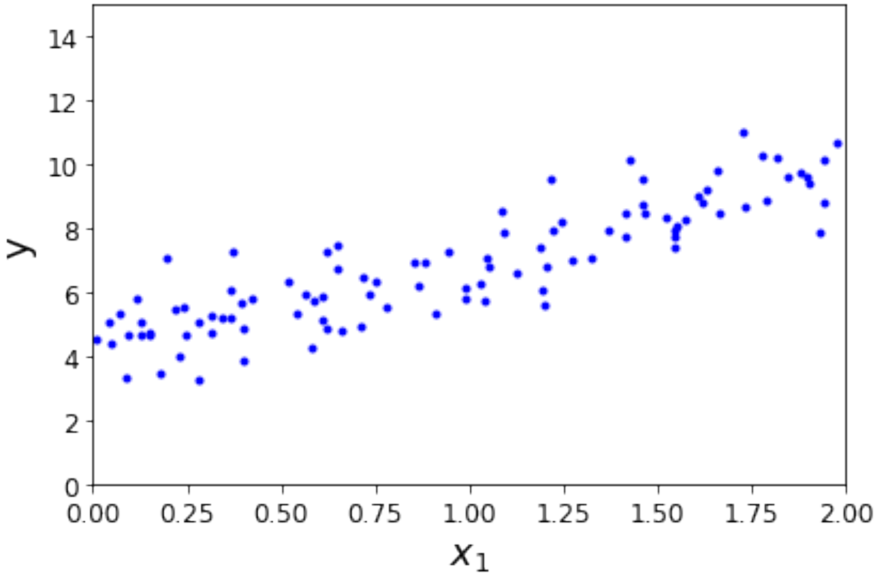
In [ ]:
```
# plotting x and y with blue dot
plt.plot(X, y, "b.")

# the label is "$x_1$" , which is typeset in LaTeX font
plt.xlabel("$x_1$", fontsize=18)

# rotation representing how much the label is rotated degrees
plt.ylabel("y", rotation=0, fontsize=18)

# assigning x-axis and y-axis limit
plt.axis([0, 2, 0, 15])
save_fig("generated_data_plot")
plt.show()
```

```
Saving figure generated_data_plot
```



In [ ]:
```
X_b = np.c_[np.ones((100, 1)), X]  # add x0 = 1 to each instance
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

In [ ]:
```
theta_best
```

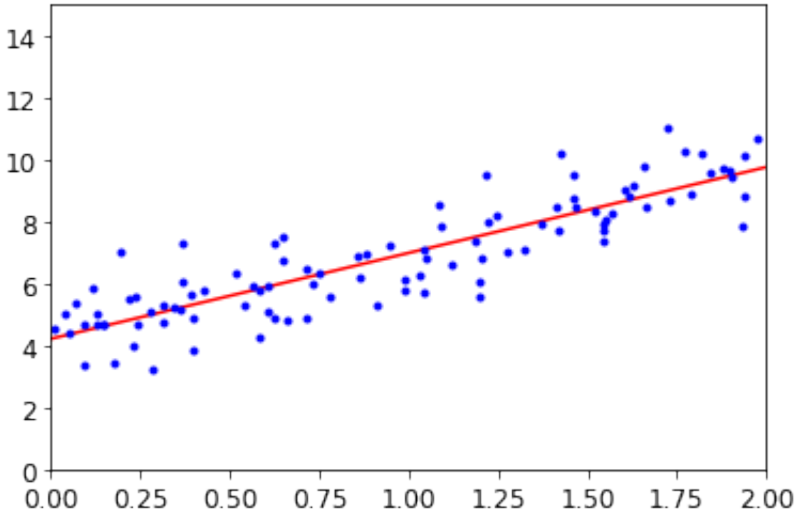
Out[ ]:
```
array([[4.21509616],
       [2.77011339]])
```

In [ ]:
```
X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]  # add x0 = 1 to each instance
y_predict = X_new_b.dot(theta_best)
y_predict
```

Out[ ]:
```
array([[4.21509616],
       [9.75532293]])
```

In [ ]:

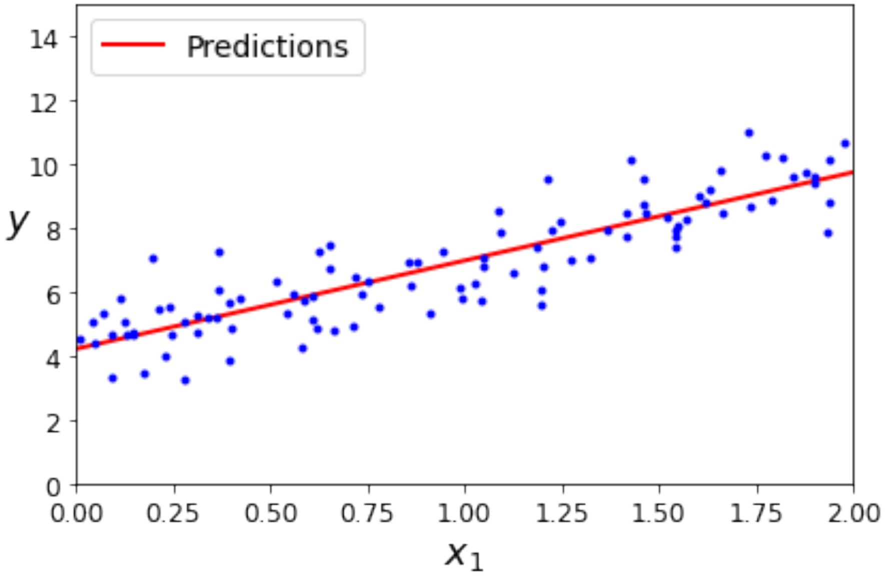
```
plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 2, 0, 15])
plt.show()
```



In [ ]:

```
plt.plot(X_new, y_predict, "r-", linewidth=2, label="Predictions")
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14)
plt.axis([0, 2, 0, 15])
save_fig("linear_model_predictions_plot")
plt.show()
```

```
Saving figure linear_model_predictions_plot
```



```
In [ ]: from sklearn.linear_model import LinearRegression

        lin_reg = LinearRegression()
        lin_reg.fit(X, y)
        lin_reg.intercept_, lin_reg.coef_
```

```
Out[ ]: (array([4.21509616]), array([[2.77011339]]))
```

```
In [ ]: lin_reg.predict(X_new)
```

```
Out[ ]: array([[4.21509616],
               [9.75532293]])
```

```
In [ ]: theta_best_svd, residuals, rank, s = np.linalg.lstsq(X_b, y, rcond=1e-
        6)
        theta_best_svd
```

```
Out[ ]: array([[4.21509616],
               [2.77011339]])
```

```
In [ ]: np.linalg.pinv(X_b).dot(y)
```

```
Out[ ]: array([[4.21509616],
               [2.77011339]])
```