In [1]:

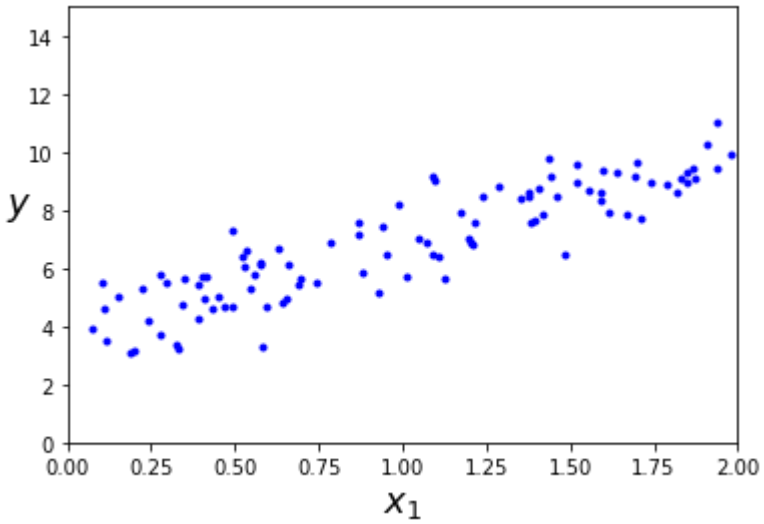
```
#%matplotlib inline
#import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:

```
#テスト用の式の定義
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
```

In [3]:

```
#x,yの描画
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 2, 0, 15])
plt.show()
```



In [ ]:

```
```

In [4]:

```
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 [5]:

```
theta_best
```

Out[5]:

```
array([[3.82373444],
       [3.05809045]])
```

In [6]:

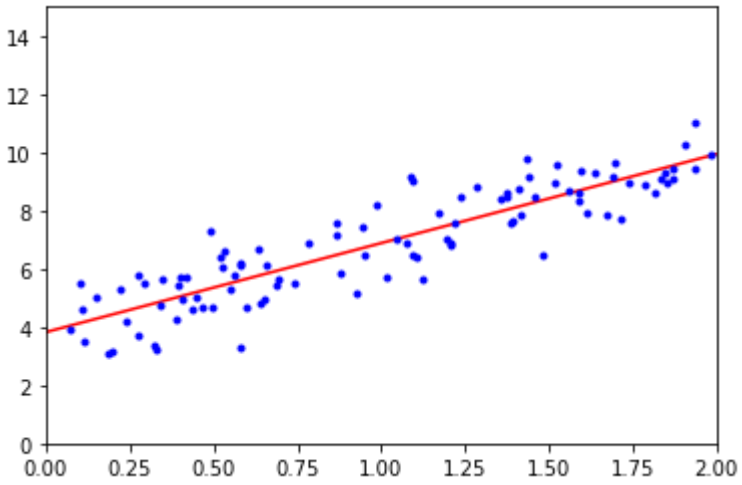
```
X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]  # add x0 = 1 to each instance
y_predict = X_new_b.dot(theta_best)
y_predict
```

Out[6]:

```
array([[3.82373444],
       [9.93991533]])
```

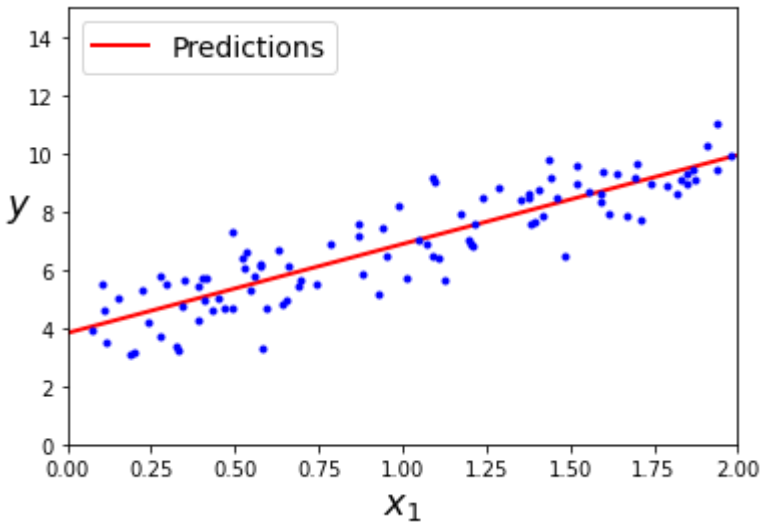
In [7]:

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



In [8]:

```
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])
plt.show()
```



In [9]:

```
from sklearn.linear_model import LinearRegression

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

Out[9]:

```
(array([3.82373444]), array([[3.05809045]]))
```

In [10]:

```
lin_reg.predict(X_new)
```

Out[10]:

```
array([[3.82373444],
       [9.93991533]])
```

The `LinearRegression` class is based on the `scipy.linalg.lstsq()` function (the name stands for "least squares"), which you could call directly:

In [11]:

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

Out[11]:

```
array([[3.82373444],
       [3.05809045]])
```

In [ ]:

```

```

In [ ]:

```

```