for example, if we combine a column vector and a row vector within the indices, we get a two-dimensional result:

```
In[7]: X[row[:, np.newaxis], col]

Out[7]: array([[ 2,  1,  3],
               [ 6,  5,  7],
               [10,  9, 11]])
```

Here, each row value is matched with each column vector, exactly as we saw in broadcasting of arithmetic operations. For example:

```
In[8]: row[:, np.newaxis] * col

Out[8]: array([[0, 0, 0],
               [2, 1, 3],
               [4, 2, 6]])
```

It is always important to remember with fancy indexing that the return value reflects the *broadcasted shape of the indices*, rather than the shape of the array being indexed.

Combined Indexing

For even more powerful operations, fancy indexing can be combined with the other indexing schemes we've seen:

```
In[9]: print(X)

[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

We can combine fancy and simple indices:

```
In[10]: X[2, [2, 0, 1]]

Out[10]: array([10,  8,  9])
```

We can also combine fancy indexing with slicing:

```
In[11]: X[1:, [2, 0, 1]]

Out[11]: array([[ 6,  4,  5],
                [10,  8,  9]])
```

And we can combine fancy indexing with masking:

```
In[12]: mask = np.array([1, 0, 1, 0], dtype=bool)
        X[row[:, np.newaxis], mask]

Out[12]: array([[ 0,  2],
                [ 4,  6],
                [ 8, 10]])
```

All of these indexing options combined lead to a very flexible set of operations for accessing and modifying array values.

Example: Selecting Random Points

One common use of fancy indexing is the selection of subsets of rows from a matrix. For example, we might have an N by D matrix representing N points in D dimensions, such as the following points drawn from a two-dimensional normal distribution:

```
In[13]: mean = [0, 0]
        cov = [[1, 2],
               [2, 5]]
        X = rand.multivariate_normal(mean, cov, 100)
        X.shape

Out[13]: (100, 2)
```

Using the plotting tools we will discuss in Chapter 4, we can visualize these points as a scatter plot (Figure 2-7):

```
In[14]: %matplotlib inline
        import matplotlib.pyplot as plt
        import seaborn; seaborn.set()  # for plot styling

        plt.scatter(X[:, 0], X[:, 1]);
```

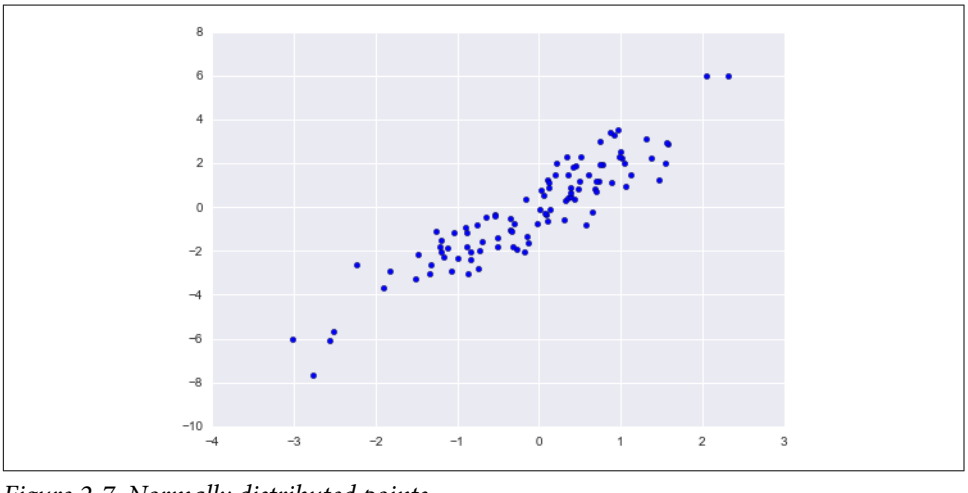

Figure 2-7. Normally distributed points

Let's use fancy indexing to select 20 random points. We'll do this by first choosing 20 random indices with no repeats, and use these indices to select a portion of the original array:

```
In[15]: indices = np.random.choice(X.shape[0], 20, replace=False)
        indices

Out[15]: array([93, 45, 73, 81, 50, 10, 98, 94,  4, 64, 65, 89, 47, 84, 82,
                80, 25, 90, 63, 20])
```