Scatter Plots

- Another commonly used plot type is the scatter plot, a close cousin of the line plot.
- Instead of points being joined by line segments, here the points are represented individually with a dot, circle, or other shape.

Necessary Settings

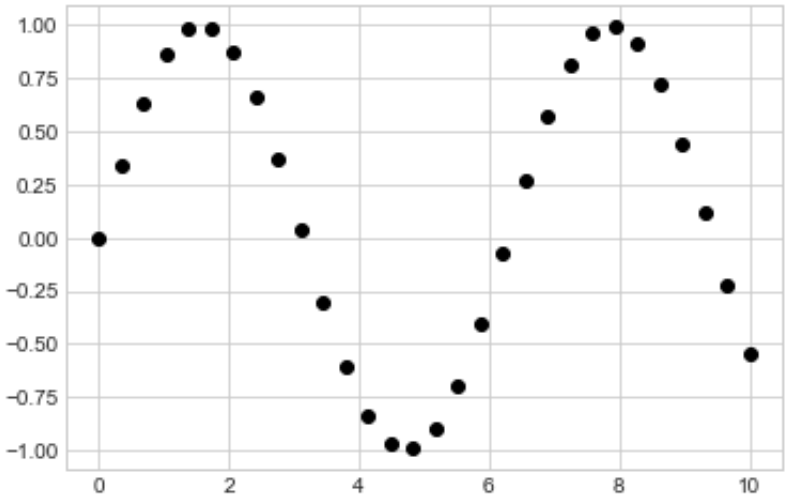
In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
```

Scatter Plots with `plt.plot`

- Previously we have seen that `plt.plot` / `ax.plot` is used to produce line plots.
- The same function can produce scatter plots as well.

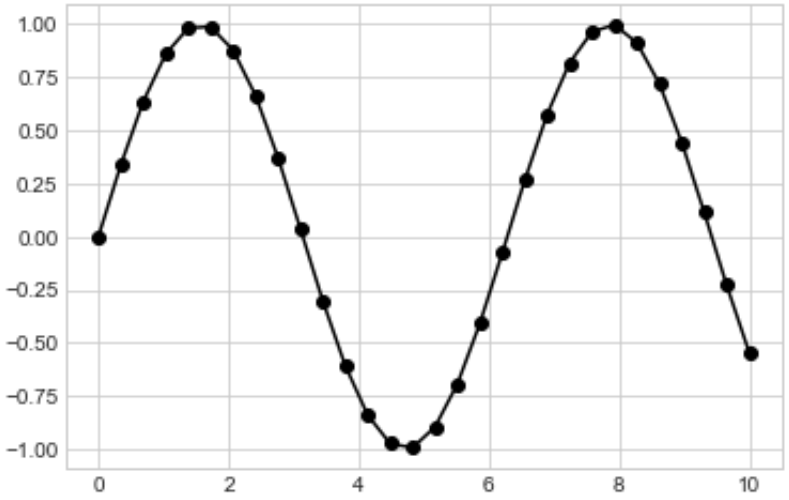
In [3]:
```
x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x, y, 'o', color='black');
```



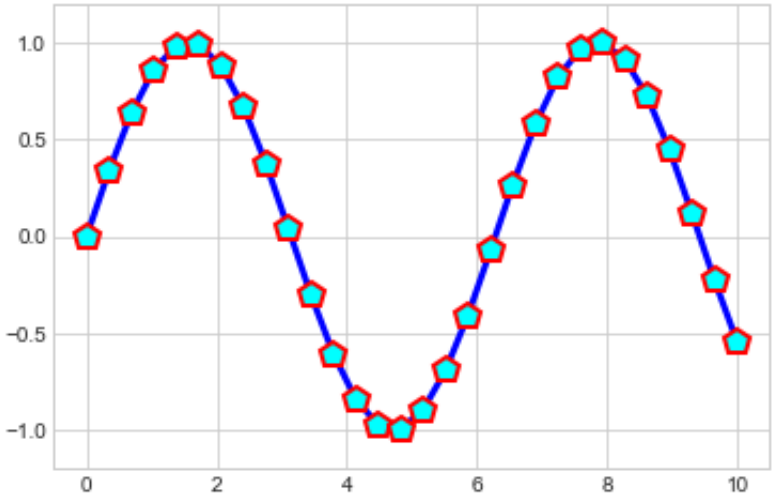
- The third argument in the function call is a character that represents the type of symbol used for the plotting (marker style).
- The marker style character codes can be used together with line and color codes to plot points along with a line connecting them.

In [8]:
```
plt.plot(x, y, '-ok');
```



- There are additional keyword arguments to `plt.plot` which specify a wide range of properties of the lines and markers.

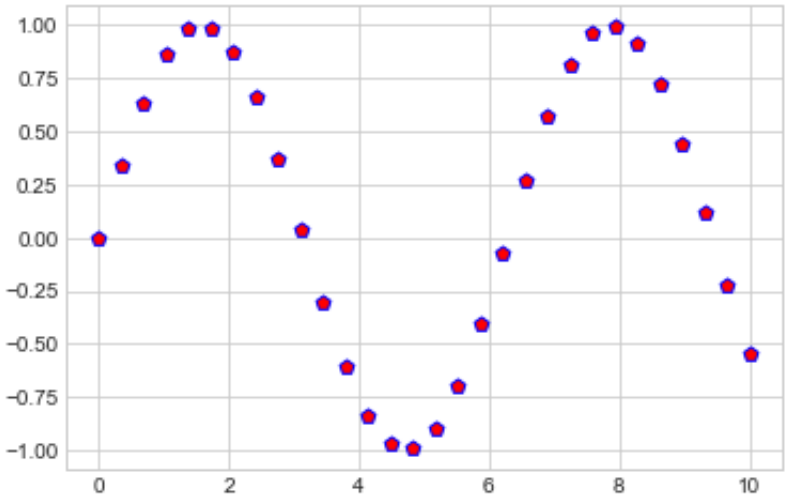
In [27]:
```
plt.plot(x, y, '-p', color='blue',
         linewidth=3,
         markersize=13,
         markerfacecolor='cyan',
         markeredgecolor='red',
         markeredgewidth=2)
plt.ylim(-1.2, 1.2);
```



Scatter Plots with `plt.scatter`

- A second, more powerful method of creating scatter plots is the `plt.scatter` function, which can be used very similarly to the `plt.plot` function.
- The primary difference of `plt.scatter` from `plt.plot` is that it can be used to create scatter plots where the properties of each individual point (size, face color, edge color, etc.) can be individually controlled or mapped to data.
- However, `plt.plot` should be preferred over `plt.scatter` for better performance in the case of large datasets.

In [47]:
```
plt.scatter(x, y, marker='p', c='red', s=50, edgecolors='blue');
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10