```
/* POTENTIOMETER

  Measure the position of a potentiometer and use it to
  control the blink rate of an LED. Turn the knob to make
  it blink faster or slower!

 What's a potentiometer?

  A potentiometer, or "pot" for short, is a control knob.
  It's the same type of control you'd use to change volume,
  dim a lamp, etc. A potentiometer changes resistance as it
  is turned. By using it as a "voltage divider", the Arduino  can sense the position of the knob, and use that value to
  control whatever you wish (like the blink rate of an LED,
  as we're doing here).*/

// Welcome back! In this sketch we'll start using "variables".

// A variable is a named number. We'll often use these to store
// numbers that change, such as measurements from the outside
// world, or to make a sketch easier to understand (sometimes a
// descriptive name makes more sense than looking at a number).

// Variables can be different "data types", which is the kind of
// number we're using (can it be negative? Have a decimal point?)
// We'll introduce more data types later, but for the moment we'll
// stick with good old "integers" (called "int" in your sketch).

// Integers are whole numbers (0, 3, 5643), can be negative, and
// for reasons we won't go into right now, can range from -32768
// to 32767. (Don't worry, if you need to work with larger numbers,
// there are other data types for that.

// You must "declare" variables before you use them, so that the
// computer knows about them. Here we'll declare two integer
// variables, and at the same time, initialize them to specific
// values. We're doing this so that further down, we can refer to
// the pins by name rather than number.

// Note that variable names are case-sensitive! If you get an
// "(variable) was not declared in this scope" error, double-check
// that you typed the name correctly.

// Here we're creating a variable called "sensorPin" of type "int"
// and initializing it to have the value "0":

int sensorPin = 0;    // The potentiometer is connected to
                      // analog pin 0

int ledPin = 13;      // The LED is connected to digital pin 13

// One more thing. If you declare variables outside of a function,
// as we have here, they are called "global variables" and can be
// seen by all the functions. If you declare variables within a
// function, they can only be seen within that function. It's good
// practice to "limit the scope" of a variable whenever possible,
// but as we're getting started, global variables are just fine.

void setup() // this function runs once when the sketch starts up
{
  // We'll be using pin 13 to light a LED, so we must configure it
  // as an output.

  // Because we already created a variable called ledPin, and
  // set it equal to 13, we can use "ledPin" in place of "13".
  // This makes the sketch easier to follow.

  pinMode(ledPin, OUTPUT);

  // The above line is the same as "pinMode(13, OUTPUT);"

  // You might be wondering why we're not also configuring
  // sensorPin as an input. The reason is that this is an
  // "analog in" pin. These pins have the special ability to
  // read varying voltages from sensors like the potentiometer.
  // Since they're always used as inputs, there is no need to
  // specifically configure them.
}

void loop() // this function runs repeatedly after setup() finishes
{
  // First we'll declare another integer variable
  // to store the value of the potentiometer:
```

```
  int sensorValue;

  // The potentiometer is set up as a voltage divider, so that
  // when you turn it, the voltage on the center pin will vary
  // from 0V to 5V. We've connected the center pin on the
  // potentiometer to the Arduino's analog input 0.

  // The Arduino can read external voltages on the analog input
  // pins using a built-in function called analogRead(). This
  // function takes one input value, the analog pin we're using
  // (sensorPin, which we earlier set to 0). It returns an integer
  // number that ranges from 0 (0 Volts) to 1023 (5 Volts).
  // We're sticking this value into the sensorValue variable:

  sensorValue = analogRead(sensorPin);

  // Now we'll blink the LED like in the first example, but we'll
  // use the sensorValue variable to change the blink speed
  // (the smaller the number, the faster it will blink).

  // Note that we're using the ledPin variable here as well:

  digitalWrite(ledPin, HIGH);     // Turn the LED on

  delay(sensorValue);             // Pause for sensorValue
                                  // milliseconds

  digitalWrite(ledPin, LOW);      // Turn the LED off

  delay(sensorValue);             // Pause for sensorValue
                                  // milliseconds

  // Remember that loop() repeats forever, so we'll do all this
  // again and again.
}
```