Your Code Often Wants to Test Something and Make a Decision Based On It

```c
if( condition )
{
    do this;
    do that;
}
```

These Operators Make up the Possible Conditions:
- `<`  Is less than
- `<=`  Is less than or equal to
- `>`  Is greater than
- `>=`  Is greater than or equal to
- `==`  Is equal to
- `!=`  Is not equal to
- `&&`  And
- `||`  Or

Example #1

```c
int x = 100;
fill( 0, 255, 0 );
for( int y = 0; y < 800; y = y + 100 )
{
    if( y >= 200  )
    {
        fill( 255, 0, 0 );
    }
    rect( x, y, 200, 100 );
}
```

Example #2

```c
fill( 0, 255, 0 );
for( int y = 0; y < 800; y = y + 100 )
{
    int x = y / 5;
    if( x < 100   &&   y >= 200  )
    {
        fill( 255, 0, 0 );
    }
    rect( x, y, 200, 100 );
}
Your Code Often Wants to Test Something and Make a Decision Based On It or the Opposite Condition

```java
if (condition)
{
    do this;
}
else
{
    do that;
}
```

Your Code Often Wants to Test Something and Make a Decision Based On It or Other Conditions

```java
if (condition)
{
    do this;
} else if (another_condition)
{
    do it;
} else
{
    do that;
}
```

Your Code Often Wants to Test Something and Make a Decision Based On It or Lots of Alternatives

```java
if (key == 'r')
{
    fill(255, 50, 50);
} else if (key == 'g')
{
    fill(50, 255, 50);
} else if (key == 'b')
{
    fill(50, 50, 255);
} else
{
    fill(100, 100, 100);
}
```

Your Code Often Wants to Test Something and Make a Decision Based On It or Lots of Alternatives -- a Better Way

```java
switch(key)
{
    case 'r':
        fill(255, 50, 50);
        break;
    case 'g':
        fill(50, 255, 50);
        break;
    case 'b':
        fill(50, 50, 255);
        break;
    default:
        fill(100, 100, 100);
}
```
Some of Processing’s Variables Already Have the Condition Built-In

```java
void draw() {
    stroke(0, 0, 0);
    fill(255, 50, 50);
    if (mousePressed) {
        rect(mouseX, mouseY, 50, 20);
    }
}
```

`mousePressed` is a built-in variable that is always telling you if a mouse button is currently pressed.

`mouseX` and `mouseY` are built-in variables that are always telling you where the mouse cursor is.