Selection & If Statements

If statements are used in Javascript to check for certain conditions to be met, in order to make choices. This ranges from checking number values, to comparing lists and words

Structure

If statements in javascript all have the same basic structure that looks something like this:

if (**check condition**){

**//Code that runs if condition is true**

}

Essentially, in the brackets after the if, you ask the computer to check something and see if it is true or not. If it is true,it runs all the code within the brackets following. Otherwise, it skips the bracketed code.

else and else if are added beneath the statements to check for further conditions if the initial conditions aren’t met. Sounding it out, a statement like this would read *“if this condition is met,do this; if not, do this. ”*. It would look something like this:

if (**check condition**){

**//Code that runs code if condition is true**

alert(“the first condition is true!”);

}

**else if** (**check a different condition**){

**//Code that runs code if second condition is true but not the first**

alert(“the first condition is false, but the second condition is true!”);

}

**else**{

**//Code that runs code if all conditions above are false**

alert(“both the first and second conditions are false”);

}

Operators

Operators are the methods for comparing values and checking conditions. Below is a list of operators for comparing various values:

* if (value1 **==** value2) - Checks if the first value is exactly equal to the second value (works with numerical values *and* strings)
* if (value1 **<** value2) or if (value1 **>** value2) - Checks if the first value is greater than or less than the second (only works with numerical values)
* if (value1 **<=** value2) or if (value1 **>=** value2) - Checks if values are less/greater than *or* equal to each other (only works with numerical values)
* if (value1 **!=** value2) - Will be true if the two values *do not* equal each other (works with numerical values *and* strings)

ORs, ANDs and NOTs

You can also add the character **||(or)**, **&&(and)**, and **!(not)** to be more specific with what conditions you are looking for. These words let you check multiple conditions at once, each in a different way:

* || will allow the indented code to run if *at least one* of the conditions it separates are true
* && will allow the indented code to run only if *all* the conditions it separates are true
* ! will *reverse* the output of the condition (true becomes false, and vice versa)

You have to be careful whenever using || and && within the same if statement, because Python prioritizes || and && in a very specific way. A good way to understand the way it does this is to put brackets around a set of conditions for every or. Here’s some examples of how it would look:

if (**(**Condition1 **&&** Condition2**)** **|| (**Condition3 **&&** Condition4 **&&** Condition5**)** **||** **(**Condition6**))**{}

if **(**Condition1**)** **||** **(**Condition2 **&&** Condition3 **&&** Condition4 **&&** Condition5**)**{}

Basically, as long as *all* the conditions *within a single set of brackets* are true, then javascript will consider the statement to be true and run the indented code. Keep this in mind whenever checking multiple conditions, and be sure to practice using the brackets.

Applications in Processing

In drawing, there are a couple uses for if statements, such as:

* Making a shape a different colour based on certain conditions
  + Some examples are if the mouse is inside the shape, if one of the mouse buttons was clicked, based on the shapes position or size, etc.
* They also help a lot with making your program interactive. For example, if the user clicks on a certain spot in your program, you can make an image appear or have some sort of event occur. Think of it as making a button that can make something happen when clicked

if statements also have a lot of use for animations, such as checking boundaries and the positions of objects being drawn, and making changes based on them. We’ll get into animation another time.

**Practice**

1. Create a variable called myCondition. Then create a program so that when myCondition = true, the program will draw a rectangle, and when myCondition = false, it will draw an ellipse.
2. Using a mouseClicked function, have a program draw a dark red square if the mouse is clicked on the left half of the canvas and draw a bright red square in the same spot if the mouse is clicked on the right half of the canvas.
3. Draw a rectangle in the bottom right corner of the canvas. Using a mouseClicked function, have a program draw a big triangle in the middle of the screen if the mouse is clicked inside the rectangle (like a button). Once you do this, see if you can do this with a circular button (this part is optional).
4. Using a mousePressed function, create a program that draws a triangle on the left side of the screen while the left mouse button is held down and a triangle on the right side of the screen while the right mouse button is held down. When the mouse buttons are released, reset the program using a mouseReleased function that calls the background command.