# PRACTICE 5: CONDITIONALS

# Take Control with Conditionals

In this lesson, you will learn how to create your very own calculator using conditional statements!

## What are Conditionals and How Do You Use Them?

We make decisions every day. What should I wear today? What should I have for lunch? Should I watch the new episode of that TV show or do my homework? When we make these sorts of choices, we consider a lot of information. If it's sunny outside, I'll wear shorts. If my homework isn't due for a few days, I'll watch that new episode. You might be surprised to know that computers make decisions in a similar way.

At this point we can give a computer a series of instructions and it will carry them out for us. But what if we don't want to perform the same instructions every time we run a program? For instance, say I want to display a special message on my birthday. The following program will do just that:

```
If Clock.Date = "2/10/2017" Then
  TextWindow.WriteLine("Happy Birthday To You!!!")
EndIf
```

*Note:* `Clock` *will tell you today's date*

This program will only display the message if it's my birthday. But how does the program know how to do this?

Conditions are statements that either evaluate to True or False. For example, "I have a twin sister" is a condition. If I do have a twin sister, then this condition is True, otherwise it is False.

In the program above the condition is "The current date is 2/10/2017" or in Small Basic:

```
Clock.Date = "2/10/2017"
```

If the current date is 2/10/2017 Then we want the program to say Happy Birthday. We use the If/Then keywords to accomplish this.

The keyword If will take a condition and evaluate it to True or False. You use Then to specify the instructions you want executed if the condition is True. If the condition is False, then the program will skip to the next line and execute those instructions. EndIf tells the program to perform the following instruction regardless of whether the condition was True or False.

## Creating Complex Conditions with Logical Operators

The above program will only print out my birthday if it's 2017, but if it isn't 2017 then that means I'm never going to get my special birthday message! Let's fix that:

```
If Clock.Month = 2 And Clock.Day = 10 Then
  TextWindow.WriteLine("Happy Birthday To You!!!")
EndIf
```

What is the condition in this program? "If the current month is February (2) AND the current day is the 10th." Or, in Small Basic:

```
Clock.Month = 2 And Clock.Day = 10
```

And is what is known as a logical operator. Logical operators allow us to combine multiple conditions into a single condition (i.e. "The current month is February", "The current day is 10"). For a condition created using And, both sub-conditions need to be True for the entire condition to be True. If it isn't February it isn't my birthday and likewise if it isn't the 10th it isn't my birthday.

Compare this with the following program that will let me know if it's the weekend:

```
If Clock.WeekDay="Saturday" Or Clock.WeekDay="Sunday" Then
  TextWindow.WriteLine("It's the weekend!")
EndIf
```

Notice that here the two sub-conditions are combined using a different logical operator: Or. If we use the Or keyword, then as long as at least one of the sub-conditions is True, then the entire condition is True. If it's Saturday, then it's the weekend and likewise if it's Sunday, it's the weekend.

Let's look at a more complicated example where you can specify an action if a condition is false:

```
TextWindow.Write("Enter a number: ")
input = TextWindow.ReadNumber()
If input > 0 Then
  TextWindow.WriteLine("That's a positive number!")
EndIf
If input < 0 Then
  TextWindow.WriteLine("That's a negative number!")
EndIf
```

In this example we specify an action to take if the condition, `input > 0`, is satisfied and a different action to take if another condition, `input < 0`, is satisfied. This code will work just fine, but there's a better way to write this program using a new keyword, `Else`.

```
TextWindow.Write("Enter a number: ")
input = TextWindow.ReadNumber()
If input > 0 Then
  TextWindow.WriteLine("That's a positive number!")
Else
  TextWindow.WriteLine("That's a negative number!")
Endif
```

This keyword allows us to specify an action to take if the condition is False. In this program the user will input a number and the condition "input is greater than zero" will be evaluated. What do you think will happen if you input zero?

But what if we want to be a bit more selective with our program? Here's one more example that's even more involved:

```
TextWindow.Write("What is the temperature today in F? ")
temperature = TextWindow.ReadNumber()
If temperature < 0 Then
  TextWindow.WriteLine("Wow it's really cold today!")
ElseIf temperature < 32 Then
  TextWindow.WriteLine ("It's freezing today!")
```

```
ElseIf temperature < 50 Then
  TextWindow.WriteLine ("It's chilly today")
ElseIf temperature < 80 Then
  TextWindow.WriteLine ("It's nice outside today")
Else
  TextWindow.WriteLine ("It's really hot today")
EndIf
```

What this program does is print out different messages for different ranges of temperatures. For a temperature between 32 and 50 degrees, it will print "It's chilly today" or for temperatures between 50 and 80 it will print "It's nice outside today." We can accomplish this with the introduction of the keyword **ElseIf**. This will allow us to nest if statements together. For example, the first few conditions above could also be written as:

```
If temperature < 0 Then
  TextWindow.WriteLine("Wow it's really cold today!")
EndIf
If temperature < 32 And temperature >= 0 Then
  TextWindow.WriteLine ("It's cold enough to freeze water
today!")
EndIf
If temperature < 50 And temperature >= 32 Then
 TextWindow.WriteLine("It's chilly today")
EndIf
```

Notice that with this approach we end up with multiple **If/EndIf** blocks. With the first approach our code has less repetition and is much cleaner.

## Challenge: Build Your Own Basic Calculator

Using what we now know about conditional statements, let's make something useful: a calculator! The basic interface is as follows:

*Remember, you can use comments to help you write the code!*

- Greet the user and ask them what operation they want to perform (addition, subtraction, multiplication, or division)
- Save this operation as a variable
- Ask them what the first operand (number) is
- Ask them what the second operand is

- Set your conditional statements for how the system should address each type of operation (which is now a variable)
- Give the user the answer

Here are some important things to think about:

- How will you keep track of what the user entered?
- How do you want the user to choose their operation?
- What happens if the user doesn't enter a valid operation? How can you let them know if they didn't enter valid input?
- How can you write your code to reduce repetition?

**Hint**: Make sure you are using `TextWindow.ReadNumber()` to read in the operands, that way the computer knows the input is a number instead of text, and you can add them. The difference is 2 + 2 = 4 (the computer thinks they are numbers and computes the mathematical result) vs. "2" + "2" = "22" (the computer thinks they are strings and combines them into one larger string).

## Discussion Questions

- Can you think of some more examples of using conditional logic in real life?
- What are some limitations associated with conditional logic? Can you think of something that would involve long or complicated conditionals?
- What can you do if you want to have a set of conditions be analyzed only if another condition is true?
- If the first condition of an `And` logical operator is False, do you need to analyze the other parts of a complex conditional?
- If the first condition of an `Or` logical operator is True, do you need to analyze the other parts of a complex conditional?

# Additional Resources

- Small Basic Tutorial: Conditions and Loops
  - https://aka.ms/sbcurriculum1.4
- Video: If then, elseif, and else
  - https://youtu.be/q80byQV_qww
- Logical operator short circuiting
  - https://aka.ms/logicaloperator