Session 6:
WHILE LOOPS

6

Loop and Conquer

Create your own adventure with variables, conditionals, and loops.

What's a Loop?

A loop is a way of repeating something, like a command or action, over and over.

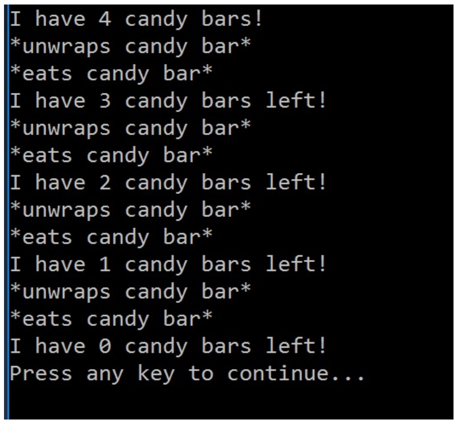
Let's imagine that you've been given the important task of eating all the candy bars in your house! You take one candy bar, unwrap it, and start eating it. When you're done with the first one, you grab another candy bar and do the same thing over again. As long as there are still any candy bars left, you have to keep going. Once you're out of candy bars, you've completed your task!

In computer science, we call this kind of loop a `While` loop. While you have candy bars to eat, you eat them. When you don't have any to eat anymore, you're done with your task. We can write this exact process in Small Basic like this:

```
candy_bars = 4

TextWindow.WriteLine("I have " + candy_bars + " candy
bars!")
While (candy_bars > 0)
     TextWindow.WriteLine("*unwraps candy bar*")
     TextWindow.WriteLine("*eats candy bar*")
     candy_bars = candy_bars - 1
     TextWindow.WriteLine("I have " + candy_bars + " candy
     bars left!")
EndWhile
```

And this is what happens when you run the code from above:



```
I have 4 candy bars!
*unwraps candy bar*
*eats candy bar*
I have 3 candy bars left!
*unwraps candy bar*
*eats candy bar*
I have 2 candy bars left!
*unwraps candy bar*
*eats candy bar*
I have 1 candy bars left!
*unwraps candy bar*
*eats candy bar*
I have 0 candy bars left!
Press any key to continue...
```

The code above shows a variable called **`candy_bars`** being declared and given a value of 4. After, we write "I have 4 candy bars!". This is being stated outside of the loop, because everything that's inside the while loop is being run from top to bottom each time the loop runs. You can see this in action in the picture of the code after it runs.

If we said, "I have 4 candy bars!" Inside the while loop, the program would say it has 4 candy bars each time the loop ran.

Once the while loop has been started, we give the loop a condition in parentheses. While this condition is true (in this case, while the amount of candy bars is not equal to 0), the loop will run.

Each time the loop runs, we write out the steps of consuming the candy bars, and then we write how many candy bars we have left. We can calculate this by subtracting 1 from `candy_bars` each time we get to the end of the loop. This tracks how many candy bars we have left to eat.

Note: Don't forget the **`EndWhile`** *keyword!*

Once the loop reaches its end, we use the keyword **`EndWhile`** to signal that the while loop will end. That way the loop doesn't continue forever, creating an infinite loop.

Challenge: Choose Your Own Adventure

Using your knowledge of variables, conditionals, and now loops, you'll be creating a multi-choice choose your own adventure story!

Now let's look at some tips and reminders to help you start off your program!

You're going to need to keep your story in a loop. That way if the user answers a question in a way your program can't understand, the program doesn't just stop; it gives the user another chance to answer the question with one of the choices you give it, and then your program continues on with the story.

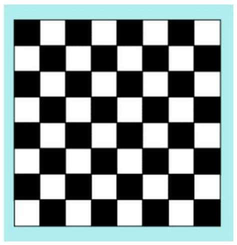
With that in mind, you need to prompt the user to answer questions. But, you also need to save their answers. You can prompt a user to answer by using `TextWindow.Write()` to ask a question, and then collect the users answer with:

```
TextWindow.Read()
```

You're also going to need to use conditionals to allow users to make multiple choices.

Challenge: Draw a Chess Board

Using a while loop, draw a chess board that looks like the image below. The chess board is an 8 by 8 grid of squares with alternating colors (black and white or any two colors you want) in each adjacent square.



You will need some drawing skills that you learned from previous lessons to get started. Do you remember how to draw a rectangle using `GraphicsWindow.DrawRectangle()`? You will use that to draw each block of the grid. The position of the squares can be calculated using the coordinates (0,0), so that the top left corner of the window is the beginning point of reference.

You will also need to use `GraphicsWindow.FillRectangle()` to fill in the color for each block. It works similarly to `GraphicsWindow.DrawRectangle():`

```
GraphicsWindow.FillRectangle(x_pos, y_pos, width, height)
```

Do you remember how to set the color using `GraphicsWindow.BrushColor="ColorName"`? You can set the color before calling `GraphicsWindow.FillRectangle()`. Note that you could just use `FillRectangle` (without `DrawRectangle`), but this way you also get the black lines in between the colored squares (which is even more important if one of your colors is white).

Have some fun picking your favorite color to be used in the chess board!

Discussion Questions

What can you use to save the input from the `TextWindow.Read()`?

How are you going to keep track of the user's choices?

Where might you use a loop in real life?

Knowing what you know now, what are some examples of how loops could simplify everyday tasks?

Additional Resources

- Small Basic Getting Started Guide: Chapter 5: Loops
    -
- Small Basic Curriculum: Lesson 1.4: Conditions and Loops
- Small Basic Example: Loop
- Video: Small Basic Tutorial 8 - While Loops
- Video: Small Basic 1.07 while loop