PRACTICE 8: SUBROUTINES

8

Reusing Code with Subroutines

Learn how to create a "terribly inaccurate truth detector" that is easy to maintain and understand by using subroutines! In other words, you're going to make a magic 8-ball.

What's a subroutine?

A subroutine is a collection of instructions that can be executed multiple times within your program, without needing to copy every instruction. Subroutines can help simplify your program because they allow you to name that set of instructions, and then invoke it from multiple places. Furthermore, subroutines can help your program be updated more easily, because they centralize repeated instructions to a single location. This means they need to be modified only once when changed.

Imagine that you ask your friend to mail a letter. You could tell them that they need to: 1) walk across the street to the mailbox, 2) open the mailbox door, 3) insert the letter, 4) close the mailbox door, and 5) put the red flag on the mailbox into the up position so that the mail carrier will know to pick it up. If you had to mail many letters, that would get exhausting! Instead, you can explain to your friend what steps are needed to mail a letter once, and thereafter, any time you asked for a letter to be mailed, they would simply perform those steps. That's a subroutine!

Let's say that we want to write a program that will determine the truth of the answers to the questions: 1) Do you clean your room regularly and 2) Have you ever peeked at presents early. We could do that with the following:

```
TextWindow.WriteLine("Do you clean your room regularly?")
TextWindow.Read()

TextWindow.Write("Truth detector declares: ")
randomNumber = Math.GetRandomNumber(2)
If (randomNumber = 1) Then
  TextWindow.WriteLine("Truth")
Else
  TextWindow.WriteLine("False")
EndIf
```

```
TextWindow.WriteLine("")
TextWindow.WriteLine("Have you ever peeked at
presents early?")
TextWindow.Read()

TextWindow.Write("Truth detector declares: ")
randomNumber = Math.GetRandomNumber(2)
If (randomNumber = 1) Then
  TextWindow.WriteLine("Truth")
Else
  TextWindow.WriteLine("False")
EndIf
```

The code on line #5 may be unfamiliar. `Math.GetRandomNumber(2)` is a function available in Small Basic that calculates a random number, either 1 or 2, and assigns the result to the variable **randomNumber**. We then test **randomNumber** to decide whether the program will respond with the truth or not. Our truth detector isn't very accurate since it is a random answer!

Let's suppose that we wanted to ask another question. It would be easy to do; we just need to copy lines #1-11 and make a few small changes. But what if we wanted to change the output of the truth detector, so that it printed "Lie" instead of "False"? We could do that by updating the three places that the program outputs "False", but we might miss one, and it seems like a lot of work. Maybe there is a better way?

As it turns out, that's exactly what we could use a subroutine for! Let's rewrite the program above with a subroutine.

```
1     TextWindow.WriteLine("Do you clean your room
      regularly?")
2     TextWindow.Read()
3
4     TruthDetectorResponse ()
5
6     TextWindow.WriteLine("")
7     TextWindow.WriteLine("Have you ever peeked
      at presents early?")
8     TextWindow.Read()
9
10    TruthDetectorResponse ()
11
```

```
12  Sub  TruthDetectorResponse
13    TextWindow.Write("Truth detector declares: ")
14    randomNumber = Math.GetRandomNumber(2)
15    If (randomNumber = 1) Then
16      TextWindow.WriteLine("Truth")
17    Else
18      TextWindow.WriteLine("False")
19    EndIf
20  EndSub
```

Note: You can create as many subroutines as you want within a program!

This program executes identically to the previous program. The difference is in how it's written. You may notice that line 12 now uses the **Sub** keyword. This indicates a subroutine. We then name that subroutine, and in this case, our name is TruthDetectorResponse. We then place the code that randomly writes out Truth or Falsehood below the subroutine name, and we close that section with the keyword EndSub. After we have defined a subroutine, we can 'call' it, by typing its name followed by open and close parentheses (see lines 4 and 10). Calling a subroutine simply means that we want to execute those instructions, and after they have been executed, to continue executing our program at the next line following the call. Notice that we can call the subroutine any number of times within our program after it's defined.

At this point, if we wanted to change the truth detector's output to be "Lie" instead of "Falsehood", we only need to change line 17 because that's the only place in the program that writes it out. Further, if we want to ask an additional question or even 10 additional questions, we only have to call the subroutine after each answer is input – by entering TruthDetectorResponse (). Finally, the code above is more readable than the original because it's been split into smaller sections and we named the subroutine.

Challenge: Magic 8-ball

Using your knowledge of variables, conditionals, text input, and now subroutines, create a virtual Magic 8-ball!

A Magic 8-ball is a plastic toy, in the shape of an 8-ball, that has the user ask a question, shake the ball, and read a 'prediction' from a clear pane on the side. The answer might be any one of the following: "It is certain", "It is decidedly so", "Without a doubt", "Yes, definitely", "You may rely on it", "As I see it, yes", "Most likely", "Outlook good", "Yes", "Signs point to yes", "Reply hazy, try again", "Ask again later", "Better not tell you now", "Cannot predict now", "Concentrate and ask again", "Don't count on it", "My reply is no", "My sources say no", "Outlook not so good", "Very doubtful".

A couple of tips that may help you get started:

- You can use a while loop to allow the user to keep asking questions, so that your program doesn't end after the first prediction!
- A subroutine that uses a random number to calculate the prediction would be a great way to start!

Discussion Questions

- How can the application of subroutines make a program easier to read and understand?
- What are other subroutines that you can think of in everyday life problems?
- Can you think of cases where you may want to have multiple subroutines?
- Do you think it's possible for a subroutine to call another subroutine?
- Can you think of a case where you might want a subroutine that calls itself?
- What guidelines would you suggest on how to use subroutines?

Additional Resources

- Small Basic Subroutines - Input and Output blog
    - http://aka.ms/SmallBasic/IOBlog
- Calling subroutines from within a subroutine - blog
    - http://aka.ms/SmallBasic/SubRoutineNesting
- Small Basic Getting Started Guide: Chapter 9: Subroutines
    - http://aka.ms/SmallBasic/Sub
- Wikipedia: Subroutines
    - https://en.wikipedia.org/wiki/Subroutine
- Definition of Routine or Subroutine
    - http://aka.ms/SmallBasic/SubDefn