Chapter 3
Do The Robot

Programs in review

Last week we learned all about how to make our very first program. We learned that the computer will do exactly what you tell it to do, and we learned that you must be very specific and focus on all the tiny details. Humans are very good at doing tasks that aren't perfectly clear, but computers are not!

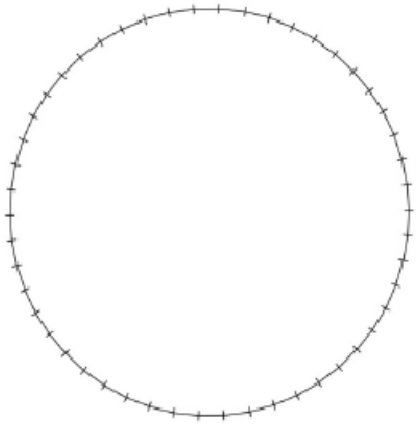
New topic: Loops!

Last week we learned how to program our robots to draw a triangle:

```
from myro import *
def triangle():
        forward(1,1)
        turrnLeft(1,.4)
        forward(1,1)
        turrnLeft(1,.4)
        forward(1,1)
```

Do you notice anything interesting about the program on the left? You may notice that the program repeats itself 3 times. This is actually problematic. Can you guess why?

In this program, the repeated code only takes up 6 lines. That's because triangles are a very simple shape, but what if you were trying to program your robot to draw a pentacontagon (50 sided polygon)?



If you tried to write a program to draw a pentacontagon like we did for the triangle, your program would be 100 lines long! That would take a while to write! How can we fix this? Using *loops*!

Loops allow us to tell the computer to do the same thing as many times as we want. We use loops all the time in our daily lives, whether you realize it or not. Let's say we have to send out 100 wedding invitations. We use a loop when we lick the stamps the same way over and over!

There are two kinds of loops, but today we will focus on *For loops*.

For loops and a group exercise

You use For loops when you know exactly how many times you want to repeat something. Let's take the triangle program:

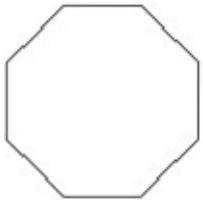
```
from myro import *
def triangle():
        forward(1,1)
        turrnLeft(1,.4)
        forward(1,1)
        turrnLeft(1,.4)
        forward(1,1)
```

```
from myro import *
def triangle():
        for i in range(3):
            forward(1,1)
            turnLeft(1,.4)
```

See the difference?

The first line says `for i in range(3)`. The word range, followed by number 3, states how many times the loop is going to run, in this case 3. The words `for i in` are a little trickier.

The letter i is a *variable*. You have heard of variables in math; they always hold a number. When you say `for i in range(3)`, you are telling the computer to make the variable i into every single number in the range of 3: First 0, then 1, then 2, because computers start counting from 0. The range of 5 would be 0, 1, 2, 3, and 4.

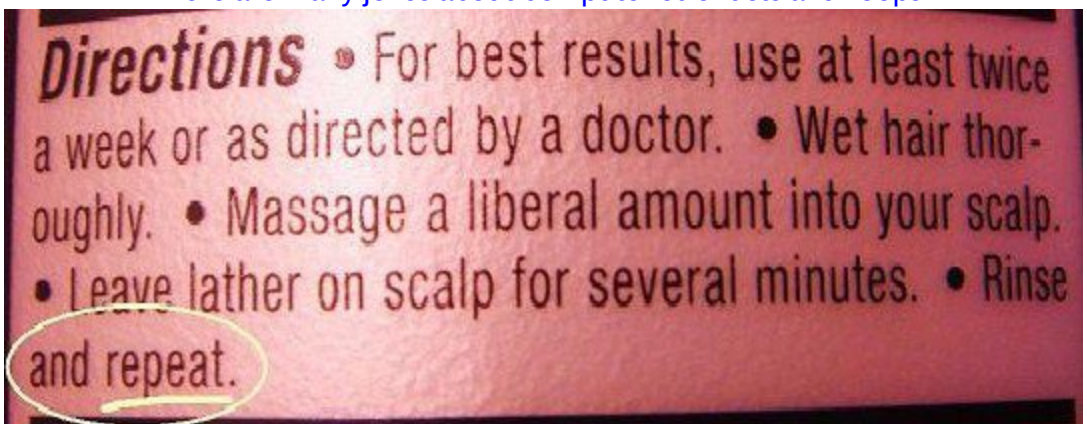


Imagine that you want to program your robot to draw an octagon using loops. Let's say that in order to create the correct angle for the octagon, you need to turn at speed 1 for .15 seconds. How would you write this program?

To Loop or not to loop, that is the question!

When do you want to use a loop? That's something you have to decide, but if you find you are doing the same thing over and over again, it might be time to use a loop! The computer figures out what to loop based on indentation. You may have noticed that all the commands underneath `for i in range(3)`were indented. This is the only way for the computer to tell whether or not a command is supposed to be repeated! So make sure that you indent at the right times! Want to write something else that shouldn't be repeated? Just unindent it!

Fun Fact!
There are many jokes about Computer Scientists and Loops!



How did the Computer Scientist die in the shower?
He followed the directions on the shampoo bottle!

Everybody dance now!

As it turns out, the robot has become one of the most popular dances in America. The only thing cooler than *doing* the robot is *being* a robot, and *dancing* robots are all the rage. Fortunately for you, your Scribbler can really boogie, but only with your help. Most dances repeat themselves over and over. Use loops to create a series of awesome dance moves for your robot to perform. Go on youtube and find a song for robot dance!

Here are some commands that you can use to make your robot get down:

```
forward(speed, seconds)
turnLeft(speed,seconds)
turnRight(speed, seconds)
beep()
speak(message) e.g. speak("Do the Sprinkler!")
stop()
wait(seconds) i.e. waits so many seconds before doing the next command
```

Discussion Topics

Computer scientists are creative. Computing requires creativity. Do you agree with these statements? Do you think most people believe computing is creative?