

CS101

# Problem Solving and Object-Oriented Programming

## L5: Flow of Control II

# Conditional Execution

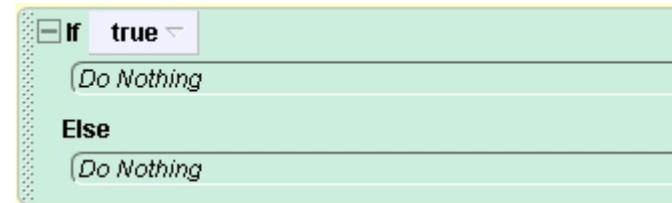
- 🌍 **Conditional execution** is where some condition is checked and a decision is made about whether a block of the program will be executed.
- 🌍 Conditional execution is extremely useful in
  - 🤖 games
  - 🤖 simulations
  - 🤖 real-time controls, e.g. robot systems



# If/Else

## Empty If/Else instruction

- Note to empty “slots” in instruction.
  - IF part
  - ELSE part



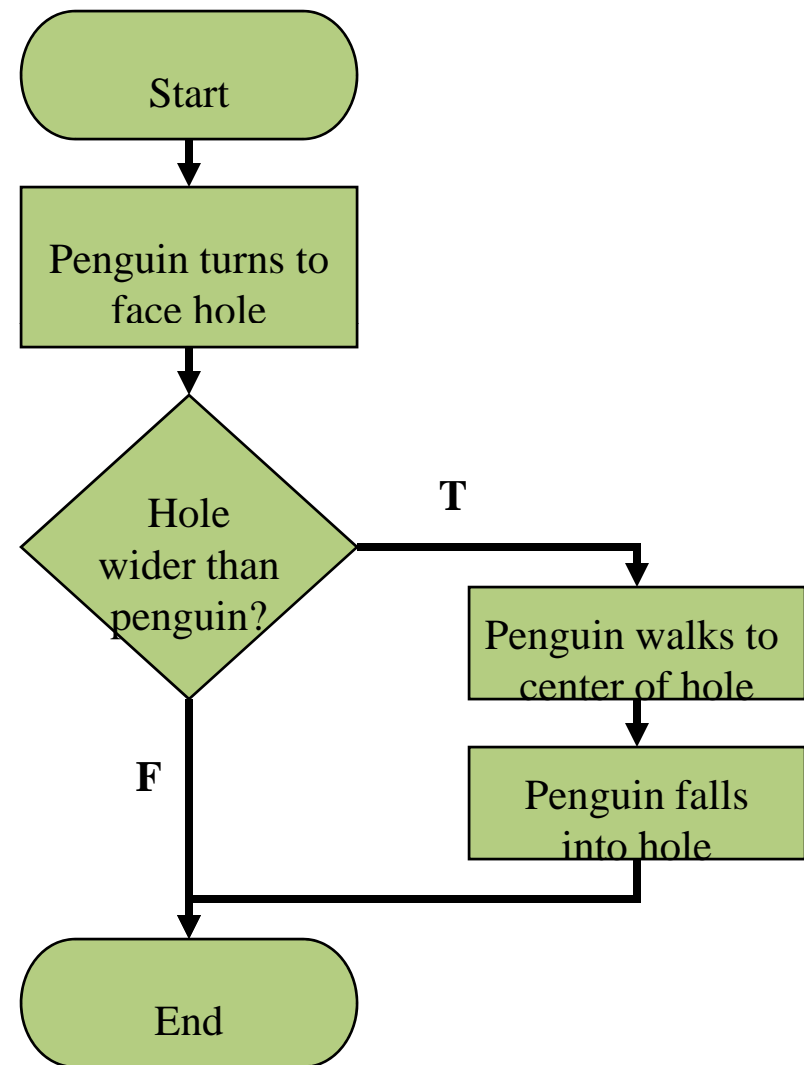
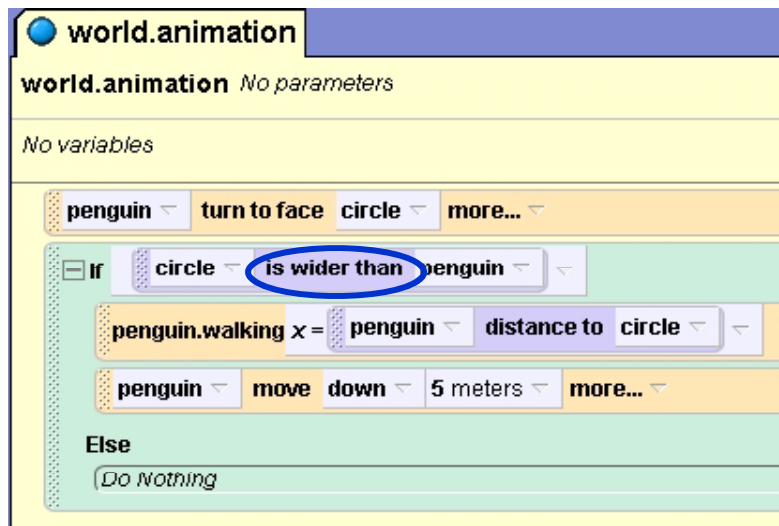
- The TRUE placeholder is replaced with the Boolean variable or a Boolean function
- Instructions tiles added to both the IF part and the ELSE part
- More than one instruction can be added to both the IF and ELSE part!

# Single-Alternative Decision Structures

- If/Else is a *dual-alternative* decision structure
  - Two paths of execution: one following TRUE and the other following FALSE
- *Single-alternative* decision structures are similar to the dual-alternative
  - The ELSE part is empty!

# Single-Alternative Decision Structures

- The condition of an If/Else statement is constructed by using a function that returns a Boolean value (true or false).

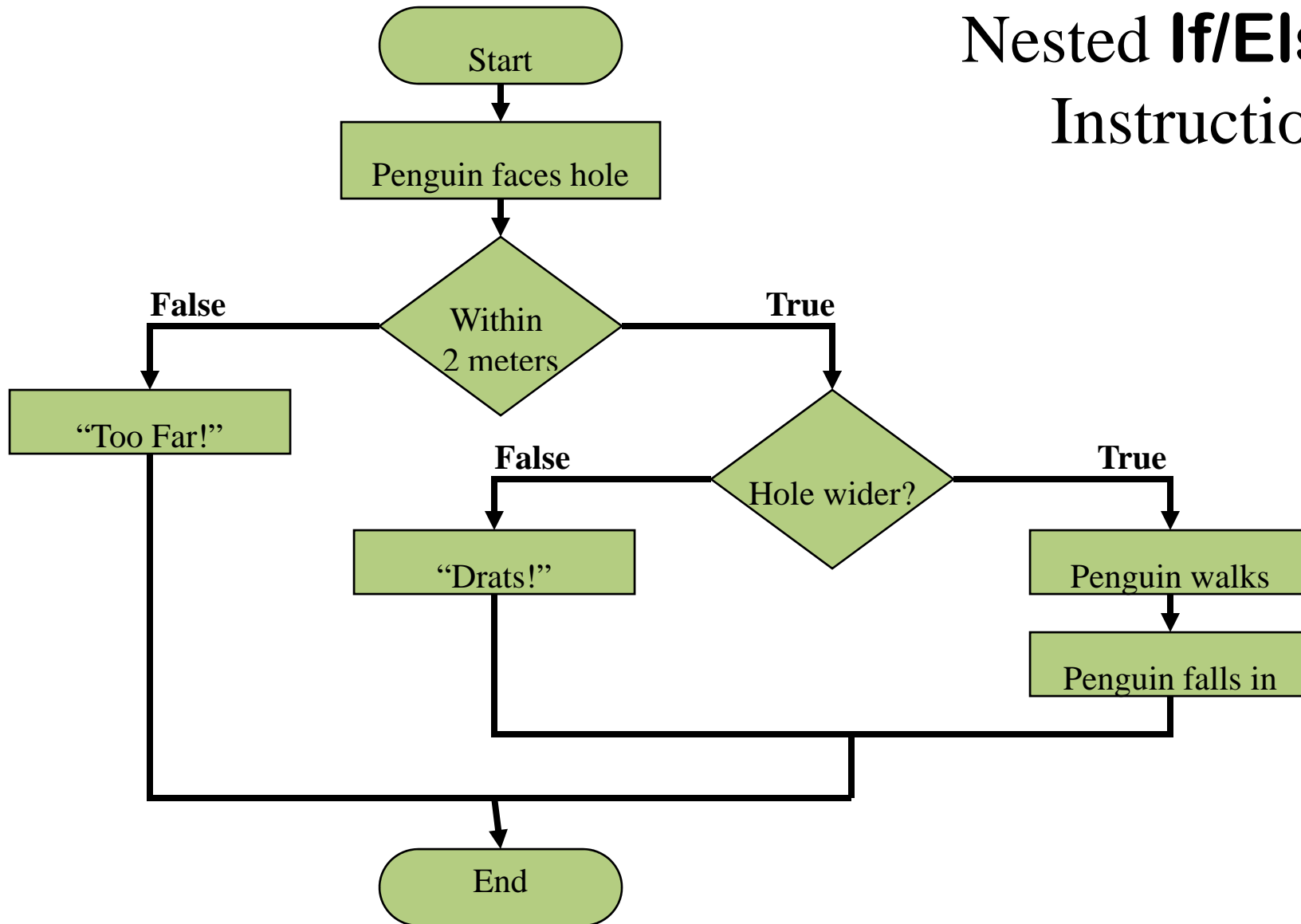


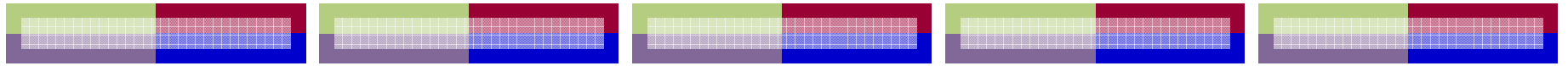
## Nested If/Else Instructions

- An **If/Else** instruction is placed inside another **If/Else** instruction or *nested*.
- The inner **If/Else** executes only if the outer **If/Else** is true.

The screenshot shows a Scratch script for a character named 'world.animation'. The script starts with a 'turn to face circle' block. This is followed by a large 'if' block with the condition 'penguin is within 2 meters of circle'. Inside this 'if' block, there is another 'if' block with the condition 'circle is wider than penguin'. This inner 'if' block contains a 'penguin.walking x = penguin distance to circle' block and a 'penguin move down 5 meters' block. Below the inner 'if' block is an 'Else' block with a 'penguin say Drats!' block. Below the outer 'if' block is another 'Else' block with a 'penguin say Too far away.' block.

# Nested If/Else Instructions





- In some cases, the built-in functions are not sufficient for a condition that we want to check.
  - For example, the built-in function *is wider than* compares the width of two objects (e.g. penguin and hole)
  - Suppose, however, that we wanted to compare the hole's width to 2 meters.
  - How can we compare the width of the hole to a specific measurement (2 meters)?



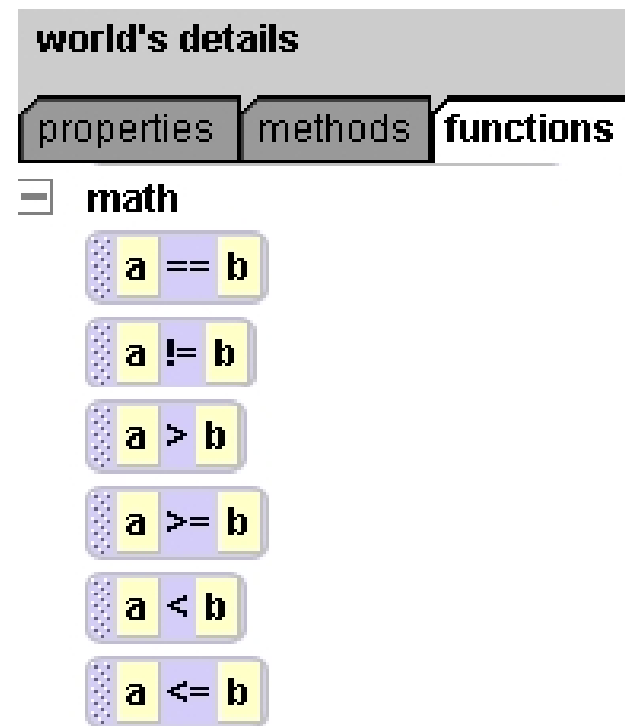
# Relational Operators

- Compare values and determine whether relationships such as greater than, less than, or equal to exist
- Compare two values and determine how they *relate* to each other

Operator	Meaning	Example
<code>==</code>	Equal to	<code>5 == 5</code>
<code>!=</code>	Not equal to	<code>5 != 4</code>
<code>&gt;</code>	Greater than	<code>9 &gt; 5</code>
<code>&gt;=</code>	Greater than or equal to	<code>9 &gt;= 9;</code> <code>9 &gt;= 5</code>
<code>&lt;</code>	Less than	<code>4 &lt; 11</code>
<code>&lt;=</code>	Less than or equal to	<code>4 &lt;= 4;</code> <code>4 &lt;= 11</code>

# Relational Operators

- Binary...operate on two pieces of data
- Can use to create conditions in an **If/Else** instruction
- Alice uses “a” and “b” for placeholders
  - Drag the desired tile and replace either the “a” or “b” with a value...such as an object

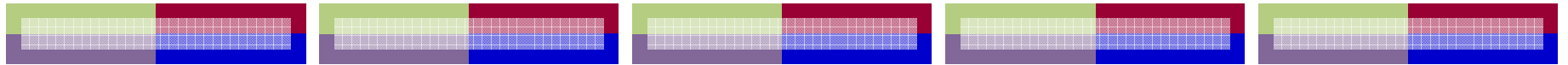


# Logical Operators

- Tests more than true/false...can do complex testing!
  - Test two conditions to see if they BOTH are true!
  - Or ONLY one condition is true!
  - Or NEITHER condition is true!

The image shows a Scratch 'world's details' menu with three tabs: 'properties', 'methods', and 'functions'. The 'functions' tab is selected, showing a 'create new function' button. Below this, a 'boolean logic' section is expanded, displaying three options: 'not a', 'both a and b', and 'either a or b, or both'. Each option is represented by a purple block with a dotted left edge and a yellow highlight on the variable 'a'.

The image shows a Scratch 'if' block with a complex logical condition. The condition is: 'both penguin is within 2 meters of circle and circle is wider than penguin'. The 'both' operator is highlighted in yellow, and the 'and' operator is also highlighted in yellow. The variables 'penguin' and 'circle' are shown in dropdown menus.



Write an if-else statement for the following scenarios:

1. If *angle* is equal to 90 degrees, print the message “The angle is a right angle”, otherwise print the message “The angle is not a right angle”
2. If *hours* is less than or equal to 20, print “ part-time pay”. If *hours* is greater than 20 and less than or equal to 40, print “full-time pay”. If *hours* is greater than 40, print “over-time pay”.

```
if hours <= 20
    “part-time pay”
else
    if both hours >20 and hours <= 40
        “full-time pay”
    else
        “over-time pay”
```



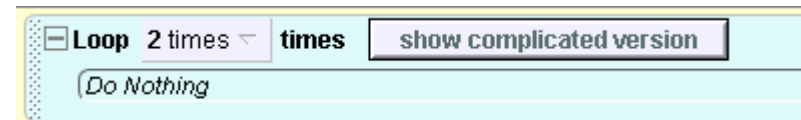
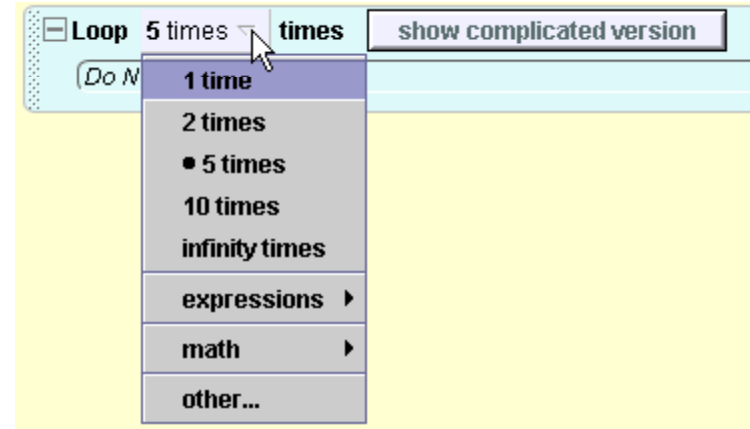
# The **Loop** Instruction

- **Loop** instructions cause one or more other instructions to *repeat* or *loop* a certain number of times.



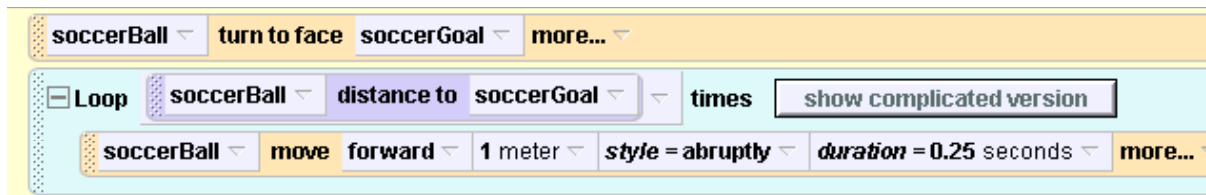
# The **Loop** Instruction

- The Loop instruction has an empty slot where instructions can be inserted.



## Computing the Number of Times to Repeat

- What if you want the something to loop and NOT have to specify a particular number of repetitions?
  - For example...getting a ball to roll across the screen, regardless of where the ball is initially placed



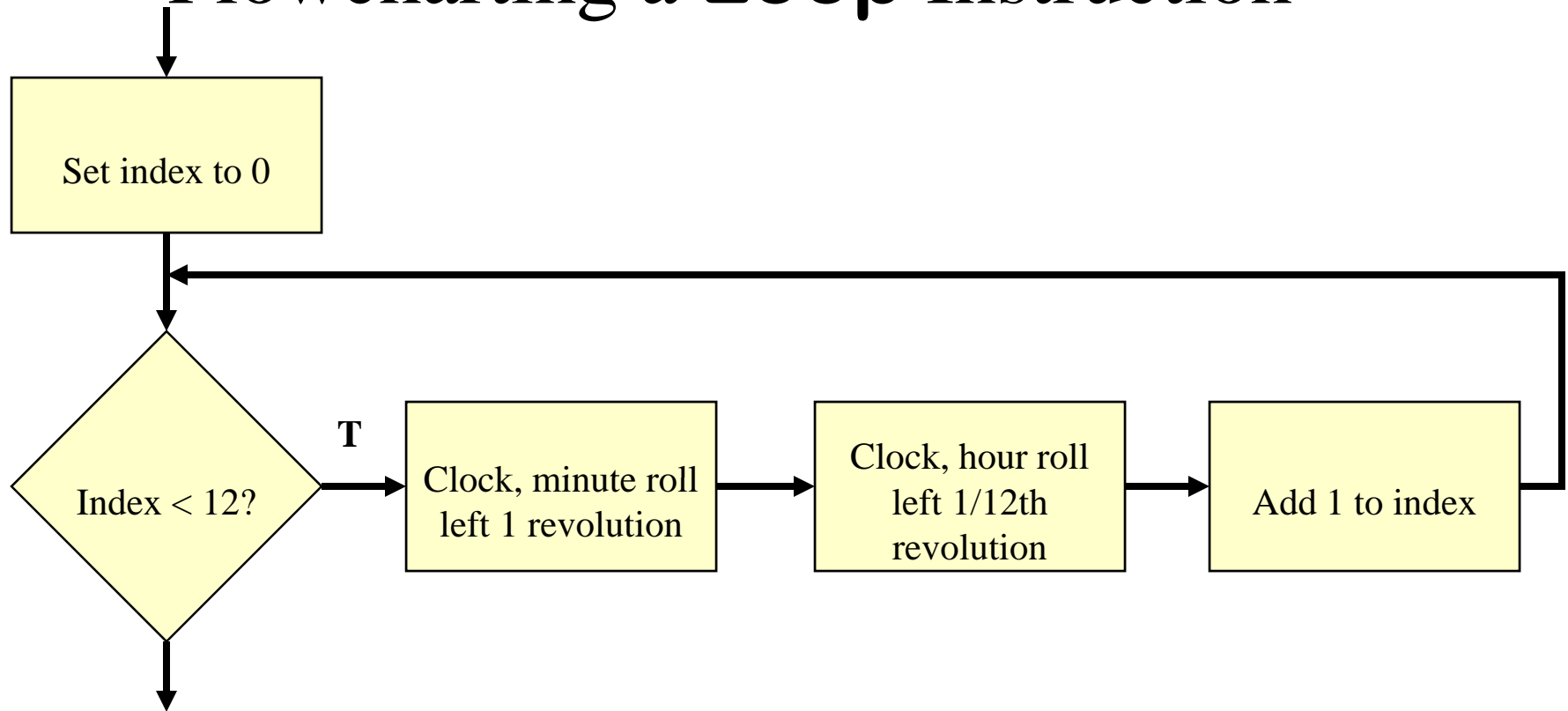
- Use the soccerBall's distance to function to calculate the distance
- Loop uses integers (whole numbers) and decimal portions are discarded
- Objects loop 4 times, not 4.8; 3 times, not 3.4

# Infinite Loops



- Infinite Loops...the loop **NEVER** stops!
  - Use for objects that shouldn't stop!
- If placed in consecutive order...the next instructions will **NEVER** occur, since the loop **NEVER** ends!
- Place an infinite loop into a *Do Together* structure with other items.

# Flowcharting a **Loop** Instruction

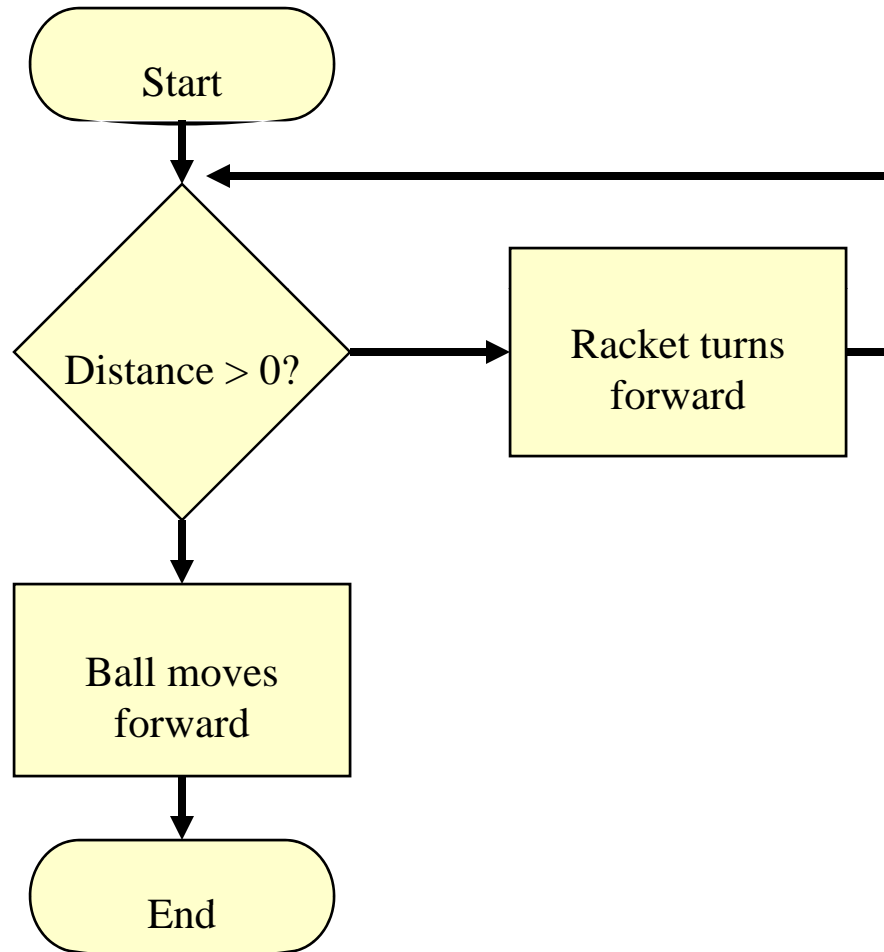


# The **While** Instruction

- **While** instruction is a loop that repeats as long as a Boolean condition is *true*
- Second repetition structure
- Called conditional loop...since the loop is controlled by a condition



# Flowcharting a **While** Instruction



- Loop's condition is tested before each repetition of the loop
- First, it tests condition
  - If true...performs a repetition and starts over
  - If false...the loop terminates