

How Tall Are You?

Introducing Functions for Alice 3



By Jenna Hayes
under the direction of Professor Susan Rodger
Duke University
July 2008
Updates made June 2014 by Yossra Hamid

Step 1: Getting Started

In this tutorial you will be learning to use functions to ask how tall a character is. Using this information two characters will compare their height and give a specific response depending on who is the tallest.

A **function** in Alice is basically a question about information in your Alice world that Alice answers. You can ask how tall or wide an object is, how far away it is from another object, and many other questions.

Step 1: Getting Started

Click on the **Setup Scene** button. Go into your **Biped Class** and click on **new Tortoise ()**. Now go back up to the **All Classes**, and find the **Flyers** folder. Click on **new penguin ()**, and add **new Penguin (Adult)** to your world.

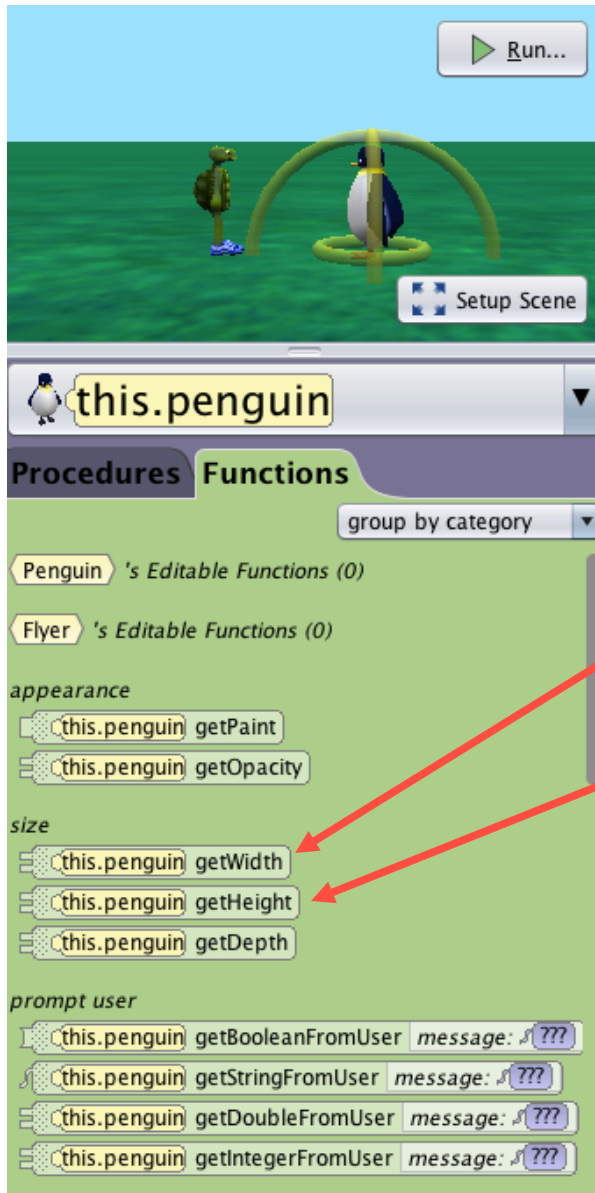


Step 1: Getting Started

Using your object moving buttons, move the tortoise to the left of your viewing screen, and move the penguin to the right of your viewing screen. **Resize** so they are approximately the same size. **Rotate** them so they are facing each other. Your world should look like the picture below.



Step 2: Understanding Functions



Click on **Edit Code**. Click on the **this.penguin** object. Then click on the **Functions** tab. You will see a LONG list of functions. Scroll down and look at the functions under **size**.

Each of these functions asks a question about the penguin, and then keeps the answer so that you can use it in your Alice world.

How wide is the penguin?

How tall is the penguin?

These functions can be very useful in Alice. What if, for example, you want to make something move up and stand on top of the penguin's head? You don't know how exactly how tall the penguin is. But Alice does!

Step 3: Using the **True** or **False** Functions

Some functions in Alice are statements to which the answer is either **true** or **false**, like the one on the previous page that says, “penguin is taller than.” We want to know whether the penguin is taller than the tortoise. It’s almost impossible to tell just by looking at them, because their heights are so close together. So we will use a function to know for sure.

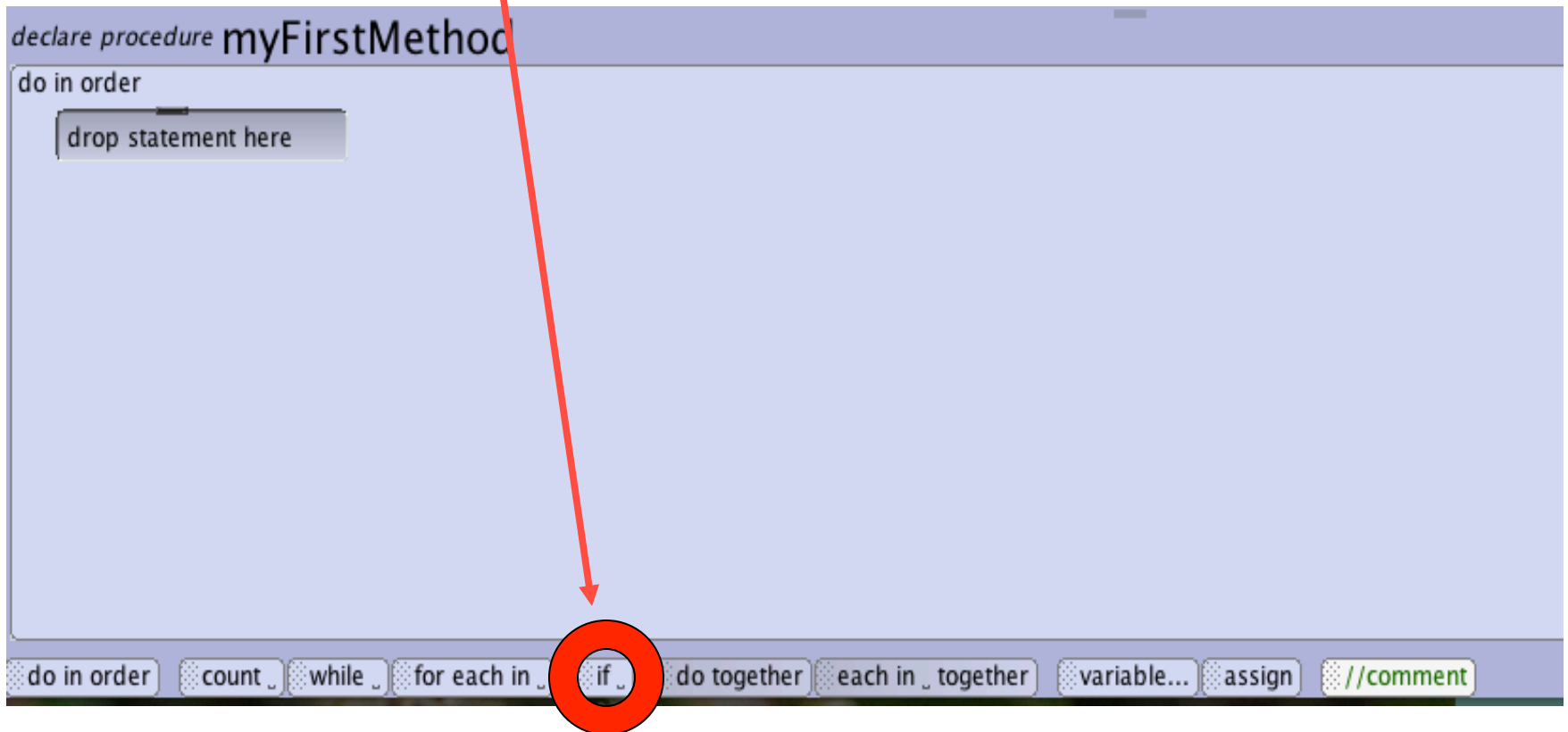
We are going to add commands so that when the Alice world starts:

- If the penguin is taller, it will say “Hah! I’m taller!”, and do a victory spin
- If the tortoise is taller, he will say, “Hah! I’m taller!”, and do a victory spin.



Step 3: True or False Continued...

Since we only want the penguin to say “Hah! I’ m taller” *if* he is taller, we need to use something called an **If Else** statement. You will find it located below your method editor:



Drag in an **If Else** statement. Select **true** when you drop it. We will replace this later.

Step 3: True or False Continued...

```
declare procedure myFirstMethod  
do in order
```

```
if true is true then  
  drop statement here  
else  
  drop statement here
```

Here is where you put the question that is either true or false. For us, that will be **penguin is taller than the tortoise**. Since it is currently set at **true**, that means this If Else says, “If this statement right here is true, do whatever commands are right under it.”

This is where you put whatever you want to happen if the answer to your question is true. This is where we’ll tell the penguin to say “Hah! I’ m taller!”

This is where you put whatever you want to happen if the answer to your question is NOT true. If it is not true, it will skip everything above the **Else**, and go straight to whatever is here. This is where we’ll tell the tortoise to say “Hah! I’ m taller!”, because if our statement is false, and the penguin is NOT taller, that means the tortoise is taller!

Step 3: True or False Continued...

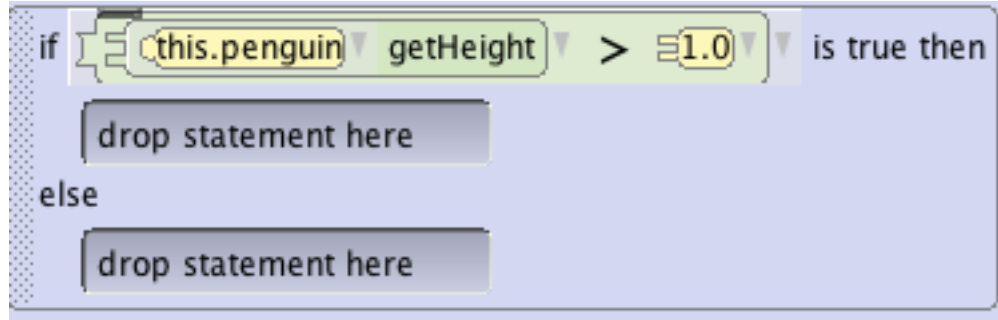
So let's construct our question. Click on **true** in the If Else statement. Click on **Relational (Decimal Number)**. Click on the greater than **>** symbol, and pick **2.0** for the first value and **1.0** for the second value.

The screenshot shows a Scratch-style programming environment. The 'if true then' block is selected, and the 'true' block is highlighted. The 'Relational (Decimal Number)' menu is open, showing the '>' operator selected. The values '2.0' and '1.0' are being chosen for the comparison.

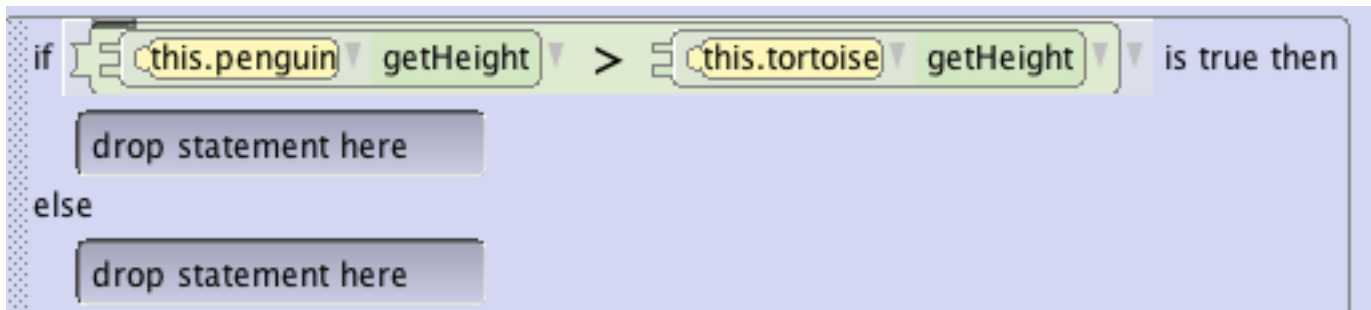
Menu Item	Sub-Item	Value
Relational (DecimalNumber) { ==, !=, <, <=, >=, > }	<=	
	<	
	>=	
	>	
	>	0.25
	>	0.5
Relational (WholeNumber) { ==, !=, <, <=, >=, > }	<=	
	<	
	>=	
Relational (SThing) { ==, != }	==	1.0
	!=	
TextString Comparison	>=	2.0
	>	
	>	10.0
Custom DecimalNumber...		1.0
		0.25
		0.5
		10.0
		Custom DecimalNumber...

Step 3: True or False Continued...

So let's construct our question. Go to the penguin's functions and drag **this.penguin getHeight** over the 2.0 in the if else statement.



Next, go to the tortoise's functions and drag **this.tortoise getHeight** over the 1.0 in the if else statement.



Now, the if else statement will ask the penguin's height is greater than the tortoise's height.

Step 3: True or False Continued...

Get a **Do together** from the bottom of your method editor and drag and drop it into the **If/Else** right below **If**. Then, click on **penguin** in the object tree, and go to his **procedures** list. Drag and drop **penguin turn** into the **Do together**. When you drop it, select **right**, and then **1**. Your code will look like this.

The image shows a code editor interface with a light blue background. The code is as follows:

```
if [this.penguin] getHeight > [this.tortoise] getHeight is true then
  do together
    [this.penguin] turn [RIGHT], [1.0] add detail
    drop statement here
else
  drop statement here
```

The 'do together' block is highlighted with a light blue background. Inside it, the 'turn' block is highlighted with a light blue background. The 'turn' block has a dropdown menu set to 'RIGHT' and a value of '1.0'. The 'add detail' block is also highlighted with a light blue background. The 'drop statement here' blocks are highlighted with a light blue background.

Step 3: True or False Continued...

Then, find **penguin say**, and drag it into the **Do together**. When the menu pops up, click on **Custom TextString**, and type in “**Hah! I’ m taller!**”.

```
if this.penguin.getHeight > this.tortoise.getHeight is true then
  do together
    this.penguin turn RIGHT, 1.0 add detail
    this.penguin say "Hah! I'm taller!" add detail
else
  drop statement here
```

Step 3: True or False Continued...

Drag and drop another **Do together** into the **If/Else**, this time right under the **Else**. Click on **tortoise** in the object tree and look at his methods list. Find **tortoise turn**, and then drag and drop that into the newest **Do together**. When you drop it, select **right**, and then **1**. Next, find **tortoise say** and put it in the **Do together**. Tell him to say “**Hah! I’ m taller!**”
Your code will look like this.

```
if [this.penguin] getHeight > [this.tortoise] getHeight is true then
  do together
    [this.penguin] turn [RIGHT], [1.0] add detail
    [this.penguin] say ["Hah! I'm taller!"] add detail
else
  do together
    [this.tortoise] turn [RIGHT], [1.0] add detail
    [this.tortoise] say ["Hah! I'm taller!"] add detail
```

The image shows a Scratch script with an if/else statement. The if condition is `[this.penguin] getHeight > [this.tortoise] getHeight`. The if block contains a **do together** block with two actions: `[this.penguin] turn [RIGHT], [1.0] add detail` and `[this.penguin] say ["Hah! I'm taller!"] add detail`. The else block also contains a **do together** block with two actions: `[this.tortoise] turn [RIGHT], [1.0] add detail` and `[this.tortoise] say ["Hah! I'm taller!"] add detail`.

Step 3: True or False Continued...

It seems that the penguin is taller. To test and make sure your function is working correctly, go to the **Set Up** screen and use your object buttons to resize the **tortoise** and make him clearly taller than the penguin. Now, play your world again. This time, the tortoise will say that he is taller! You can change him back to his normal size after you test this out.



Step 4: Using Number Functions

Now we're going to use one of the functions that is a question whose answer is a number. We're going to make the penguin move right up to the tortoise and give him a hug. The only problem is, we don't know how far to tell him to move! That's why we'll use a function!

```
if (this.penguin.getHeight > this.tortoise.getHeight)
  do together
    this.penguin turn RIGHT 1.0
    this.penguin say "Hah! I'm taller!"
else
  do together
    this.tortoise turn RIGHT 1.0
    this.tortoise say "Hah! I'm taller!"
this.tortoise say "Let's be friends. Give me a hug, penguin!"
```

First we'll tell the tortoise to say something. He's tired of competing with the penguin about their height, and he wants to be friends.

Click on **this.tortoise**, then click on **Procedures**. Find **tortoise say**, and drag it under your If Else statement.

Make him say “**Let's be friends. Give me a hug, penguin!**”

Step 4: Number Functions Continued...

Now we want the penguin to move right up to the guy to hug him.

Click on **penguin** in your object tree and then click on **procedures**. Find the **penguin move** and drag it into your method editor under everything else. We don't know exactly how far yet, so just put 1 meter for now.

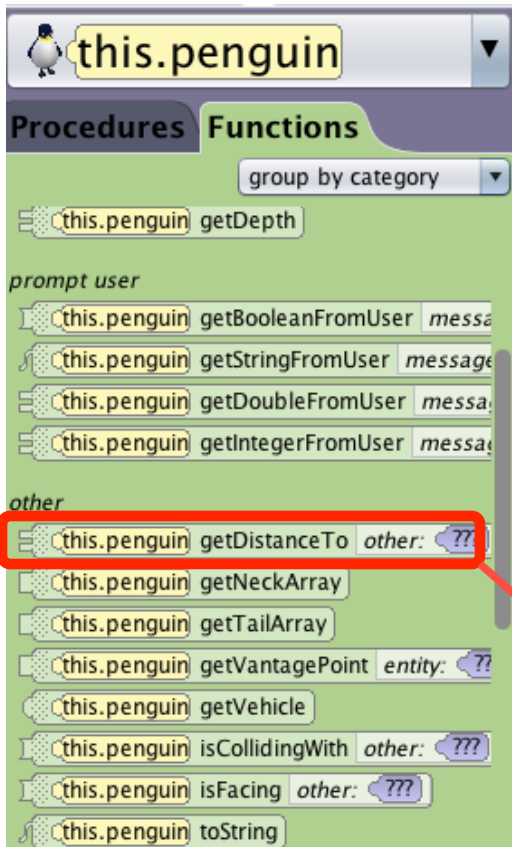
```
if this.penguin > this.tortoise
  do together
    this.penguin turn RIGHT, 1.0 add detail
    this.penguin say "Hah! I'm taller!" add detail
else
  do together
    this.tortoise turn RIGHT, 1.0 add detail
    this.tortoise say "Hah! I'm taller!" add detail
this.tortoise say "Let's be friends. Give me a hug, penguin!" a
this.penguin move FORWARD, 1.0 add detail
```

Play your world just to see what it looks like.

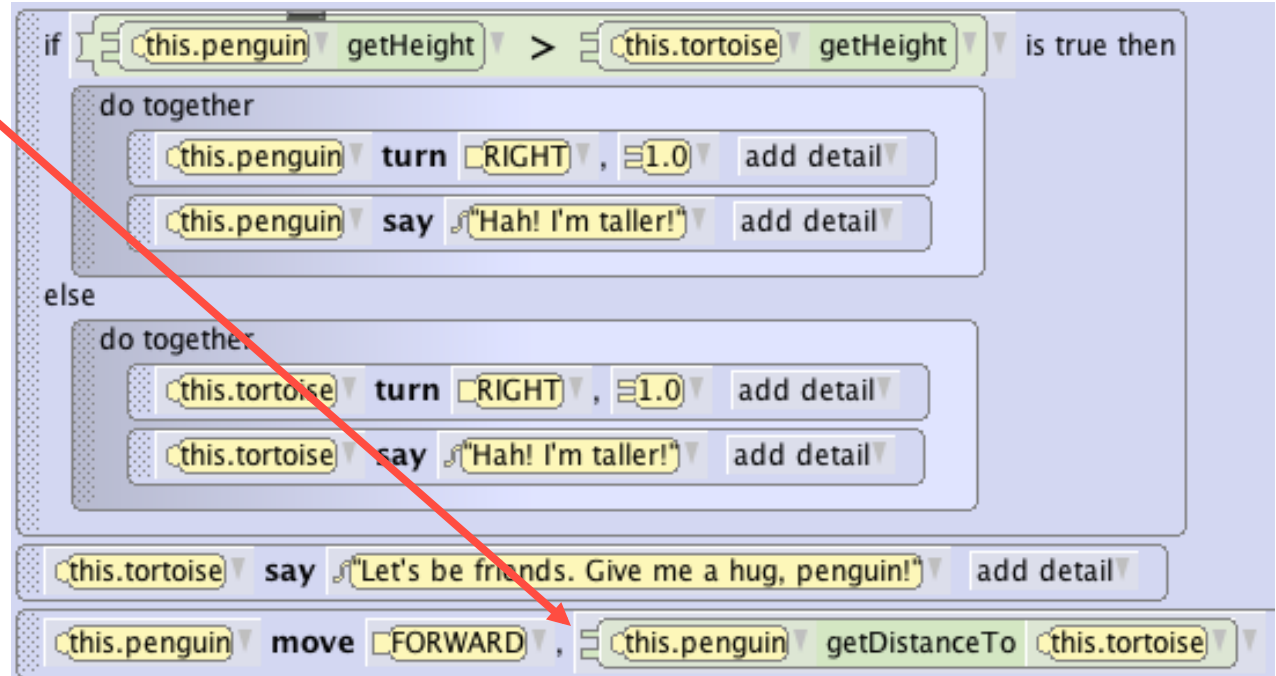
Step 4: Number Functions Continued...

Now we're going to use a function to tell the penguin how far forward to move. Click on **penguin** in your object tree, and then click on the **functions** tab. Find the function under "other" called **get distance to**.

Drag that function over the **1 meter** on your **penguin move forward** command and drop it there. On the menu that pops up, select **tortoise**.



The screenshot shows the Scratch Functions palette for the 'this.penguin' object. The 'Functions' tab is selected, and the 'other' category is expanded. The 'getDistanceTo' function is highlighted with a red box. The function's parameters are 'other: ???'. Other functions visible include 'getDepth', 'getBooleanFromUser', 'getStringFromUser', 'getDoubleFromUser', 'getIntegerFromUser', 'getNeckArray', 'getTailArray', 'getVantagePoint', 'getVehicle', 'isCollidingWith', 'isFacing', and 'toString'.



The screenshot shows a Scratch script for a 'this.tortoise' object. The script is as follows:

```
if [this.penguin > this.tortoise] is true then
  do together
    this.penguin turn RIGHT, 1.0 add detail
    this.penguin say "Hah! I'm taller!" add detail
  else
    do together
      this.tortoise turn RIGHT, 1.0 add detail
      this.tortoise say "Hah! I'm taller!" add detail
  this.tortoise say "Let's be friends. Give me a hug, penguin!" add detail
  this.penguin move FORWARD, [this.penguin getDistanceTo this.tortoise]
```

A red arrow points from the 'getDistanceTo' function in the palette to the '1 meter' value in the 'move FORWARD' block.

Step 4: Number Functions Continued...

Try playing your world. What happens? The penguin moves too far, into the body of the tortoise. It would be nice if the penguin would stop about 0.5 meters in front of tortoise. We can select math, followed by “-”, followed by a number. [Click here](#) to apply math.

The screenshot shows the Scratch code editor interface. A 'move FORWARD' block is selected, and the 'add detail' menu is open. The 'amount' field is currently empty. The 'Math' option is selected in the menu, and the '-' operator is chosen with '0.5' as the distance. A red arrow points from the text 'Click here' to the '-' operator in the menu.

Operator	Value
+	0.25
-	0.5
*	1.0
/	2.0
	10.0
	Custom DecimalNumber...

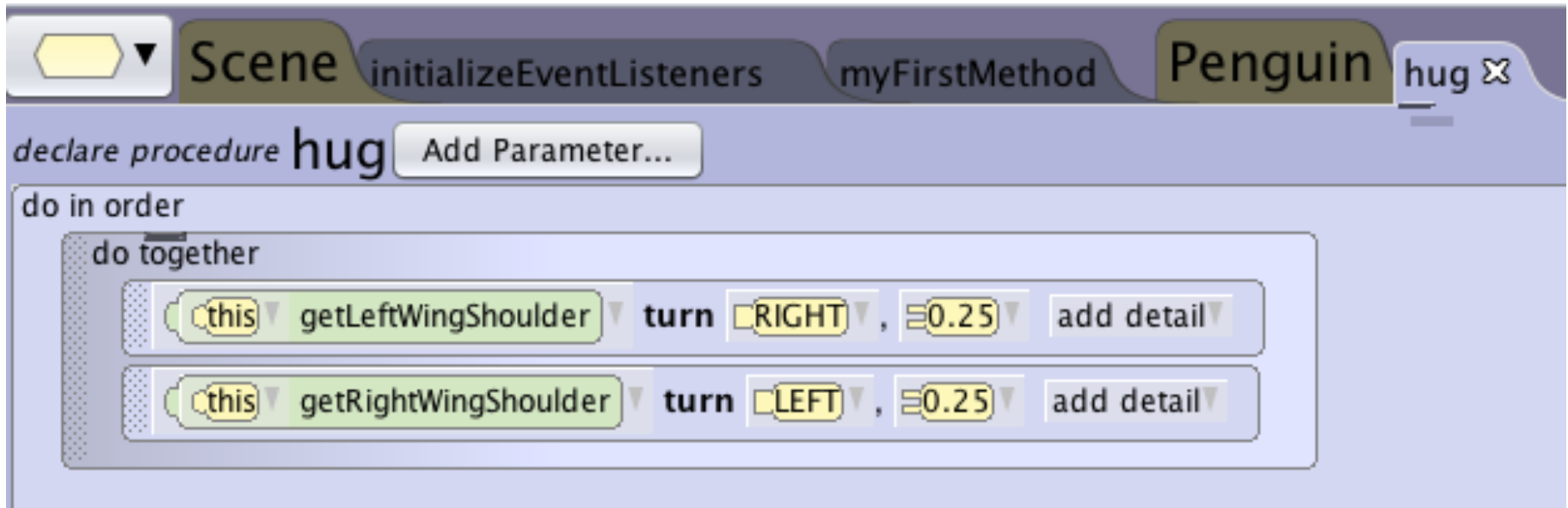
Your code for this line will then look like this:

The final code block in the Scratch editor is: `this.penguin move FORWARD, this.penguin getDistanceTo this.tortoise - 0.5 add detail`

Step 5: Finishing Up

The last thing you need to do is make the penguin hug the guy.

Go to classes, then penguins, and click on **add Penguin procedure**. Call it **hug**. In this method, experiment until you find code that makes the penguin hug the guy. My code for **penguin.hug** looks like this.



The screenshot shows the Scratch code editor interface. The top navigation bar includes a yellow hexagon icon, a 'Scene' tab, and several sub-tabs: 'initializeEventListeners', 'myFirstMethod', 'Penguin', and 'hug' (which is currently selected and has a close button 'X'). Below the navigation bar, the code editor displays the following code:

```
declare procedure hug Add Parameter...  
do in order  
  do together  
    this getLeftWingShoulder turn RIGHT, 0.25 add detail  
    this getRightWingShoulder turn LEFT, 0.25 add detail
```

Step 5: Finishing Up

Now click on the **my first method** tab of your method editor. Make sure you have clicked on **penguin** in the object tree, and then look at the penguin's **procedures** list. Find **penguin.hug**, and drag it into the bottom of your method editor. Your final code will look like this.

```
declare procedure myFirstMethod
do in order
  if [this.penguin] getHeight > [this.tortoise] getHeight is true then
    do together
      [this.penguin] turn RIGHT, 1.0 add detail
      [this.penguin] say "Hah! I'm taller!" add detail
    else
      do together
        [this.tortoise] turn RIGHT, 1.0 add detail
        [this.tortoise] say "Hah! I'm taller!" add detail
      [this.tortoise] say "Let's be friends. Give me a hug, penguin!" add detail
      [this.penguin] move FORWARD, [this.penguin] getDistanceTo [this.tortoise] - 0.5
      [this.penguin] hug
```

The image shows a Scratch code editor window titled "myFirstMethod". The code is written in a block-based style. It starts with a "do in order" block. Inside, there is an "if" block with the condition "[this.penguin] getHeight > [this.tortoise] getHeight is true then". The "if" block has two "do together" blocks. The first "do together" block contains two blocks: "[this.penguin] turn RIGHT, 1.0 add detail" and "[this.penguin] say 'Hah! I'm taller!' add detail". The second "do together" block contains two blocks: "[this.tortoise] turn RIGHT, 1.0 add detail" and "[this.tortoise] say 'Hah! I'm taller!' add detail". Below the "if" block, there are three more blocks: "[this.tortoise] say 'Let's be friends. Give me a hug, penguin!' add detail", "[this.penguin] move FORWARD, [this.penguin] getDistanceTo [this.tortoise] - 0.5", and "[this.penguin] hug".

Step 5: Finishing Up Continued...

Play your world, and see the plot unfold! Now you know the basics of using functions. We only used two functions in this tutorial, but there are MANY functions in the Alice world that can be useful. Try some more functions on your own and explore how they work!

