TINKER ACADEMY

SCRATCH Computer Programming Adventure (Beginner)

Handout 3: Statements Expressions Control Flow and Variables (Part 1)

Note your Student ID. You will need to use it throughout the Course.

Setup Instructions In Classroom

Connect to the Local Class Network

1. Select WiFi “TINKER ACADEMY”
2. This network has only LOCAL access and does NOT connect to the internet

Update the Course

1. Ensure you are connected to “TINKER ACADEMY”
2. Restart the VM. Login into the VM.
3. Open Firefox in the VM
4. Your Instructor would tell you what to type in the browser. (Typically it is 192.168.1.5)
5. You should see a page with a list of entries.
6. Click on CourseUpdate<Date>.zip. This will download CourseUpdate<Date>.zip onto your VM
7. Open Nautilus. Click on Downloads. You should see the file CourseUpdate<Date>.zip
8. Right Click on CourseUpdate<Date>.zip. Select Extract Here.
9. Open the extracted folder
10. Double click Course Update. Select “Run” in the window.

Update the Course (Alternate Approach In Class Using USB)

1. Borrow a USB drive from the Instructor
2. If you are on VirtualBox
    a. Click on Devices in the Top level Menu
    b. Select Drag ‘n’ Drop
    c. Select Bidirectional
3. If you are on VirtualBox (Another Way)
    a. Shutdown Virtual Machine
    b. Click on VM in the VirtualBox Manager
    c. Click on the Settings
    d. Click General

    e. Click Advanced Tab
    f. Select “Bidirectional” under Drag ‘n’ Drop
    g. Click OK
    h. Start Virtual Machine
4. If you are on VMWare
    a. Open the virtual machine settings editor (VM > Settings),
    b. Click the Options tab
    c. Select Guest isolation.
    d. Deselect Disable drag and drop to and from this virtual machine
5. Open Nautilus, Click on Desktop
6. Drag the file CourseUpdate<Date>.zip from Windows or Mac onto Desktop in your Virtual Machine
7. Right Click on CourseUpdate<Date>.zip. Select Extract Here.
8. Open the extracted folder
9. Double click Course Update. Select “Run” in the window.
10. Eject the USB Drive and hand it back to the Tinker Academy instructor

Setup Instructions At Home

Connect to your Home WiFi Network

Updating the Course (Using Wifi)

1. Make sure you are on the Home WiFi Network.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Update”. Choose “Run”.
   If you see a window popup with the message “update course failed”.
   Hop onto Skype, and request help in the class chat group.
   And send an email to classes@tinkeracademy.com with your name and student ID.
4. Follow the instructions in this handout (last 2 pages) on the quiz and homework steps.

Submitting Quiz and Homework

1. Make sure you are on the Home WiFi Network.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Submit”. Choose “Run”.
   If you see a window popup with the message “submit course failed”.
   Hop onto Skype, and request help in the class chat group.
   And send an email to classes@tinkeracademy.com with your name and student ID.

Virtual Machine Installation

Installing the Virtual Machine (VM)

1. Borrow the USB drive from your Tinker Academy instructor

2. Create the folder “tinkeracademy” (without the quotes) under Documents using Finder or Windows Explorer. Type it in *exactly* as indicated.
3. Copy the folder “installers” from the USB drive to under “tinkeracademy” using Finder or Windows Explorer
4. Eject the USB Drive and hand it back to the Tinker Academy instructor
5. Locate the VirtualBox installer under “tinkeracademy” using Finder or Windows Explorer

If your Laptop is	Double click on
Windows 7	VirtualBox-4.3.12-93733-Win.exe
Windows 8	VirtualBox-4.3.14-95030-Win.exe
Mac	VirtualBox-4.2.26-95022-OSX.dmg

6. Install the VirtualBox application
7. Congratulations, You completed a major milestone. Give yourself a pat on the back :)

Importing the Virtual Machine (VM)

1. Locate the Virtual Machine “tinkeracademy.ova” under “tinkeracademy”
2. Double click on “tinkeracademy.ova”. You should get the import screen in VirtualBox with an “Import” Button. Click on the “Import” button to Import the Virtual Machine.

Starting the Virtual Machine (VM)

1. Once the Import is complete and successful, you should see the VM “TinkerAcademy” in the side panel in VirtualBox.
2. If it says “Powered Off” click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
3. If it says “Running” click on the Show Button (Green Arrow) in the VirtualBox Toolbar. This should display the VM window.
4. Once the VM starts up you will be presented with a login screen. Type in “password” without the quotes. Type it in exactly as indicated and hit “Enter”.
5. Once the login is completed you should see a Desktop with a few icons. The Screen might go fuzzy for a few seconds before displaying the Desktop. *That is ok.*
6. Congratulations. You are now running Linux within your laptop.
7. Double click on the “Firefox” icon in the Sidebar. This should launch Firefox. Verify you have network access. Close “Firefox”

Launching the Virtual Machine in Full Screen

1. Use the VirtualBox menu View->Switch to Fullscreen to switch the VM to fullscreen mode
2. Use the same VirtualBox menu View->Switch to Fullscreen to switch the VM back out of fullscreen mode

Shutting Down the Virtual Machine

1. Click on the red close window button (to the top left on a Mac, top right in Windows).
2. You will prompted with a confirmation message asking if you want to “Power Off” the machine. Click the button to confirm power off.
3. In a few minutes the VM will shut down and you should see the VirtualBox side panel with the “Tinker academy” VM indicating “Powered Off”.

Restarting the Virtual Machine

1. Start VirtualBox
2. Click on the VM “TinkerAcademy” in the VirtualBox side panel.
3. Click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
4. Once the VM startup you will be presented with a login screen.

Right Click in VM on Mac

1. Open System Preferences, Trackpad
2. Enable “Secondary Click”, Toggle the small arrow to the right and select “Click with two fingers”.

Getting Ready to Program

Open StarterPack3.sb

We will be using StarterPack3.sb for this class.
Click on StarterPack3.sb (under Courses, TA-SCR-1, starterpack. starterpack3)
Now either

- Right Click and select “Open with Scratch 2” to open the program in Scratch 2
- Double click to open the program in Scratch 2

Structure of StarterPac3.sb

The Scratch Program in StarterPack3.sb has

1. A Sprite named “Karel”. Karel is a Robot.
2. Sprites named “Marker 0”, “Marker 1”, “Marker 2”, “Marker 3”, “Marker 4” and “Marker 5”
3. The Stage - with a special backdrop of Karel’s city

About Karel’s City

Karel lives in a city. The Center of the City is at (0,0) of the Stage. Karel’s city has roads goir east to west and south to south. The intersection of 2 roads is called a “corner”.

Markers are used to help Karel navigate the City Roads. Markers are numbered from 0 to 5. Each marker has a location using 2 numbers. The first number indicates the “x” position on the Stage, i.e. where it is along the east to west direction. The second number indicates the “y” position on the Stage, i.e. where it is along the south to north direction.

Use Markers to locate the position

Markers are used to help Karel navigate the City Roads. Markers are numbered from 0 to 5.

If you click on the marker, it will “tell” you its location giving its “x” position and its “y” position. Before we start to program, we are going to do is to check that our markers are working correctly. This will also let you know where the markers are on the Stage.
So click on Marker 0, Marker 1, Marker 2 and so until Marker 5. Each Marker will “tell” you its location.

Updates to the Markers

Markers are used to help Karel navigate the City Roads. Markers are numbered from 0 to 5.

In addition to telling you the location, clicking on Marker 1 to Marker 5 will tell you if Karel nee to **go up, go right**, **go down, go left** or **STOP** if Karel is touching **the marker**.

Karel always starts at Marker 0.

If Touching Marker	Karel should
Marker 0	go right
Marker 1	go up
Marker 2	go right
Marker 3	go down
Marker 4	go left
Marker 5	**STOP**

Moving a Marker to a different intersection
You can move Markers 1 to 5 to a different intersection

1. Drag the marker to near the intersection
2. Click on the marker. It will move and snap in the place at the closest intersection.

Run the Program

1. Click on the Green Flag
2. Make sure the Markers are at the following locations

Markers 0, 1, 2, and 5 **will be used** for this program

Marker 0	(-150, -100)
Marker 1	(-50, -100)
Marker 2	(-50, 0)
Marker 5	(50, 0)

3. Make sure Karel is at Marker 0
4. If Karel ends up "hiding" behind other markers
    a. Click on Karel Sprite
    b. Looks Palette, Click on "go to front"

What should our Program do?

1. Our program should start when the green flag is clicked.
2. Karel should start at location (0,0) and start moving east to west.
3. On reaching an intersection, Karel should do the following
    a. **Go Right** if Karel is over **Marker 0**
    b. **Go Up** if Karel is over **Marker 1**
    c. **Go Right** if Karel is over **Marker 2**
    d. Go Down if Karel is over **Marker 3**
    e. Go Left if Karel is over **Marker 4**
    f. **STOP** if Karel is over **Marker 5**

Refresher

The Scratch program below has 1 Sprite. The Sprite has 2 Scripts. What will happen when the Green Flag is clicked?

Script Tab

Script 1

when Flag clicked

glide (1) secs to x:(0) y:(0)

Script 2

glide (1) secs to x:(100) y:(100)

	Script 1	Script 2
Green Flag clicked	Script 1 is activated	Script 2 is not activated
	The Sprite glides to (0,0)	

Statements

What is a statement?

In programming, a statement is **an instruction to do something.**

Every block whose **label reads like a command** in a Script is a statement block.

In Handout 2, we covered the goto and glide statements

go to x:(0) y:(0)

glide (1) secs to x:(0) y:(0)

Another statement is the stop (all) statement.

The stop (all) statement stops executing the program. This is similar to clicking on the red circle to stop executing the program.

stop (all)

Expressions

What is an expression?
In programming, an expression is something that returns a value.

There are various types of expressions based on the type of value they return

Some of the various types are

- Boolean expression
- Integer expression
- Text expression

Boolean Expressions

What is an Boolean expression?
Every expression has to return a value. A very special type of expression called the Boolean expression returns only ONE of possible values

- True
- False

Every block **shaped like a diamond** in a Script is a Boolean expression block.

One such Boolean expression block is the touching block.

The Boolean expression block below is part of Karel's Script.

The block will return True if the Karel is touching Marker 1.

The block will return False as soon as Karel stops touching Marker 1.

Sensing Palette

touching Marker 2 ?

Step 1:
Click on Karel Sprite
Click on the Sensing Palette
Drag the touching Marker 0 ? block onto the Stage. Select Marker 0 from the drop down.
Drag the touching Marker 1 ? block onto the Stage. Select Marker 1 from the drop down.
Drag the touching Marker 2 ? block onto the Stage. Select Marker 2 from the drop down.
Drag the touching Marker 3 ? block onto the Stage. Select Marker 3 from the drop down.
Drag the touching Marker 4 ? block onto the Stage. Select Marker 4 from the drop down.
Drag the touching Marker 5 ? block onto the Stage. Select Marker 5 from the drop down.

Other Boolean expression blocks are the less than, equals and greater than blocks.

Operators Palette

3 < 5	Returns True since 3 is less than 5
5 < 3	Returns False since 5 is not less than 3
3 == 5	Returns False since 3 is not equal to 5
5 == 5	Returns True since 5 is equal to 5

Additional Sensing Boolean expression blocks

Sensing Palette

touching color []?	Returns True if the Sprite is touching the specified color []
color [] is touching []?	Returns True if the color [] is touching color [] color [] should be from within the Sprite color [] should be from the background or another Sprite
key <Key> pressed?	Returns True if the the key <Key> is pressed <Key> can be one of up arrow down arrow right arrow

	left arrow space a, b, …, z 0, 1, 2, .. 9
mouse down?	Returns True if the mouse button is clicked within the stage

Integer Expressions

What is an Integer expression?
Every expression has to return a value. Integer expression return integers as values.

Almost every block **shaped with a rounded edge** in a Script is an Integer expression block.

The only exceptions are the Text expression blocks covered below.

Integer expression block can be arithmetic blocks

Operators Palette

5 + 3	Returns 8 since 5 + 3 = 8
5 - 3	Returns 2 since 5 - 3 = 2
5 * 3	Returns 15 since 5 x 3 = 15 (In programming * represents multiplication)
5 / 3	Returns 1.666666..7 since 5 / 3 = 1.666666..7 (rounded decimal)

The integer expression block below is very useful. It returns the remainder after the division.

Operators Palette

5 mod 3	Returns 2
3 mod 3	Returns 0
2 mod 3	Returns ?

Other integer expression blocks

Motion Palette

x position	Current x position of the Sprite on the Stage. The center is at (0,0)
y position	Current y position of the Sprite on the Stage. The center is at (0,0)
direction	Current direction of the Sprite showing which way its heading. 0 indicates its heading up 90 indicates its heading right 180 indicates its heading down -90 indicates its heading left

Sensing Palette

mouse x	Current x position of the mouse pointer on the Stage. The center is at (0,0)
mouse y	Current y position of the mouse pointer on the Stage. The center is at (0,0)
loudness	Volume (1 to 100) of the sound detected by the computers microphone
timer	Value of the timer (started using the statement block reset timer) in seconds
x position S	x position of Sprite S
y position S	y position of Sprite S
current sec	current second other options are current year current month current date current day of week current hour current minute

Sound Palette

volume	Sound volume for the Sprite
temp	Sound tempo in beats per minute for the Sprite

Text Expressions

What is a Text expression?
Every expression has to return a value. Text expression return some text as values.

Text expression blocks are **shaped with a rounded edge similar to Integer expressions.** However unlike Integer expressions, Text expression blocks return text.

The Text expression blocks are listed below.

Operators Palette

join “hello ” “world”	Returns “hello world” after joining “hello “ and “world”
letter (1) of “world”	Returns “w” since the first letter of the word “world” is “w”

Sensing Palette

answer	Returns keyboard input

They will be covered in more detail later in the course.

Combining Expressions

We Muggles know how to take simple Lego blocks and create a model of a airplane or a Star Wars spacecraft or a house or anything else that you can dream of.

Similarly, we can combine simple expressions and make bigger expressions.

The Boolean expression block below is part of Karel’s Script.

The block will return True if the Karel is touching Marker 1.

The block will return False as soon as Karel stops touching Marker 1.

touching Marker 2 ?

Combining Using Not Expressions

The not Boolean expression block returns True if Karel is not touching Marker 1

The not Boolean expression block returns False if Karel is touching Marker 1

not	touching Marker 2 ?

Combining Using And Expressions

The and Boolean expression block returns True if Karel is touching Marker 1 AND touching Marker 2

The and Boolean expression block returns False if Karel is not touching Marker 1 OR Karel is not touching Marker 2

touching Marker 1 ?	and	touching Marker 2 ?

Combining Using Or Expressions

The or Boolean expression block returns True if Karel is touching Marker 1 OR touching Marker 2

The or Boolean expression block returns False if Karel is not touching Marker 1 AND Karel is not touching Marker 2

touching Marker 1 ?	or	touching Marker 2 ?

Combining Arithmetic Expressions

y position + 100	y position + 100 (Go Up)
y position - 100	y position - 100 (Go Down)
x position + 100	x position + 100 (Go Right)
x position - 100	x position - 100 (Go Left)

Step 2:
Click on Karel Sprite
Click on Operators Palette
Drag the ( ) + ( ) block onto the Stage
Click on Motion Palette
Drag the x position block into the first slot of the ( ) + ( ) block.
Click on the second slot of the ( ) + ( ) block and type in 100
The block should look like this

(x position) + (100)

Drag another ( ) + ( ) block onto the Stage
Click on Motion Palette
Drag the y position block into the first slot of the ( ) + ( ) block.
Click on the second slot of the ( ) + ( ) block and type in 100
The block should look like this

(y position) + (100)

Drag another ( ) + ( ) block onto the Stage
Click on Motion Palette
Drag the x position block into the first slot of the ( ) + ( ) block.
Click on the second slot of the ( ) + ( ) block and type in 100
The block should look like this

(x position) + (100)

Drag another ( ) - ( ) block onto the Stage
Click on Motion Palette
Drag the y position block into the first slot of the ( ) - ( ) block.
Click on the second slot of the ( ) - ( ) block and type in 100
The block should look like this

(y position) - (100)

Drag another ( ) - ( ) block onto the Stage
Click on Motion Palette

Drag the x position block into the first slot of the ( ) - ( ) block.
Click on the second slot of the ( ) - ( ) block and type in 100
The block should look like this

(x position) - (100)

Condition

What is a Condition?
A condition is something that can either be True or False.

What should our Program do?

1. Our program should start when the green flag is clicked.
2. Karel should start at location (0,0) and start moving east to west.
3. On reaching an intersection, Karel should do the following
    a. If the intersection does not have a marker, Karel should keep moving
    b. If the intersection has a marker then Karel should
        i. **Go Up** if Karel is over **Marker 1**
        ii. **Go Right** if Karel is over **Marker 2**
        iii. **STOP** if Karel is over **Marker 5**

Our program uses conditions.

Condition
If Karel is over Marker 0
If Karel is over Marker 1
If Karel is over Marker 2
If Karel is over Marker 3
If Karel is over Marker 4
If Karel is over Marker 5

Our program should do something different depending on whether the condition is True

Condition	If True, Karel should
If Karel is over Marker 0	Go Right
If Karel is over Marker 1	Go Up
If Karel is over Marker 2	Go Right
If Karel is over Marker 3	Go Down
If Karel is over Marker 4	Go Left
If Karel is over Marker 5	STOP

Conditional Statements

What is a Conditional Statements?
In programming, a statement is **an instruction to do something.**

Conditional statements are one or more instruction to do something **ONLY if the condition is True**

Condition	Conditional Statement
If Karel is over Marker 0	glide 1 secs to x: x position + 100 y: y position
If Karel is over Marker 1	glide 1 secs to x: x position y: y position + 100
If Karel is over Marker 2	glide 1 secs to x: x position + 100 y: y position
If Karel is over Marker 3	glide 1 secs to x: x position y: y position - 100
If Karel is over Marker 4	glide 1 secs to x: x position - 100 y: y position
If Karel is over Marker 5	STOP

If Then Conditional Statement

If Then Conditional Statement
The If Then Conditional Statement is a Block that **contain** one or more blocks.

The contained blocks get executed ONLY if the condition is true

Step 2:
Click on Karel Sprite
Click on the Motion Palette
Drag 5 glide 1 secs to x: () y: () block onto the stage
Update them to

glide 1 secs to x: x position + 100 y: y position
glide 1 secs to x: x position y: y position + 100
glide 1 secs to x: x position + 100 y: y position
glide 1 secs to x: x position y: y position - 100
glide 1 secs to x: x position - 100 y: y position

Step 3:
Click on Karel Sprite
Click on the Control Palette
Drag the if < > then block onto the stage
Drag the touching Marker 0 ? into the < > slot
The block should look like this

if <touching Marker 0 ?> then

Drag the if < > then block onto the stage
Drag the touching Marker 1 ? into the < > slot
The block should look like this

if <touching Marker 1 ?> then

Drag the if < > then block onto the stage
Drag the touching Marker 2 ? into the < > slot
The block should look like this

if <touching Marker 2 ?> then

Drag the if < > then block onto the stage
Drag the touching Marker 3 ? into the < > slot
The block should look like this

if <touching Marker 3 ?> then

Drag the if < > then block onto the stage
Drag the touching Marker 4 ? into the < > slot

The block should look like this

if <touching Marker 4 ?> then

Step 5:
Click on Karel Sprite

Drag	Into
glide 1 secs to x: x position + 100 y: y position	if <touching Marker 0 ?> then
glide 1 secs to x: x position y: y position + 100	if <touching Marker 1 ?> then
glide 1 secs to x: x position + 100 y: y position	if <touching Marker 2 ?> then
glide 1 secs to x: x position y: y position - 100	if <touching Marker 3 ?> then
glide 1 secs to x: x position - 100 y: y position	if <touching Marker 4 ?> then

If Then Else Conditional Statement

If Then Else Conditional Statement
The If Then Else Conditional Statement is a Block that **contain** 2 sets of blocks.

The first set of blocks get executed ONLY if the condition is true.
The second set of blocks get executed ONLY if the condition is false.

We will cover the If Then Else Conditional Statement in more detail in the next class.

Repeat Until Conditional Statement

Repeat Until Conditional Statement
The Repeat Until Conditional Statement is a Block that **contain** other blocks.

The contained blocks get executed UNTIL the condition is True

Step 6:
Click on Karel Sprite
Click on the Control Palette

Drag the repeat until < > block onto the stage
Drag the touching Marker 5 ? into the < > slot

Drag	Into
if <touching Marker 0 ?> then	repeat until <touching Marker 5 ?>
if <touching Marker 1 ?> then	repeat until <touching Marker 5 ?>
if <touching Marker 2 ?> then	repeat until <touching Marker 5 ?>
if <touching Marker 3 ?> then	repeat until <touching Marker 5 ?>
if <touching Marker 4 ?> then	repeat until <touching Marker 5 ?>

Step 7:
Click on Karel Sprite
Drag the repeat until <touching Marker 5 ?> block under the when Flag clicked below the go to x:(-150) y:(-100) block

That was a LOT we covered!

You made it this far! Awesome!

We will cover **Statements Expressions Control Flow and Variables (Part 2)** in the next class class. For now you need to make sure you have a good conceptual understand of Statements Expressions Control Flow and Variables (Part 1).

Quiz 3: Statements Expressions Control Flow and Variables (Part 1)

Make sure you read this Handout!

Open the Quiz

Make sure you are on the Home WiFi.
Follow the instructions in “Updating the Course” in this Handout.
Open Quiz3.odt under “Courses” “TA-SCR-1” “quiz” “quiz3”

Complete the Quiz

1. Attempt each question. Type in the answers in the “Answer:” box.
2. Save the file using File->Save or Ctrl-S

Submit the Quiz

Make sure you are on the Home WiFi.
Follow the instructions in “Submitting Homework” in this Handout.

Homework 3: Statements Expressions Control Flow and Variables (Part 1)

Make sure you read this Handout!

Overview

In this Homework you will program using Conditional Statements so that Karel can move from Marker 0 to Marker 5

Open the Homework

Follow the instructions in “Updating the Course” in this Handout.
Open Homework3.sb under “Courses” “TA-SCR-1” “homework” “homework3”

- Select “Homework3.sb”
- Right Click, Select Open With Scratch 2 OR
- Double click the

Complete the Homework

You will need to refer to this handout (Handout3). Make sure you read it thoroughly.

1. Go through this Handout. Make sure you understand **Statements Expressions Control Flow and Variables**
2. Open the StarterPack3.sb
3. Your code would be similar to StarterPack3.sb

Submit the Homework

Make sure you are on the Home WiFi.
Follow the instructions in “Submitting Homework” in this Handout.