TINKER ACADEMY

SCRATCH Computer Programming Adventure (Beginner)

Handout 6: Block Programming (Part I)

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 StarterPack6.sb
We will be using StarterPack6.sb for this class.
Click on StarterPack6.sb (under Courses, TA-SCR-1, starterpack. starterpack6)
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 Our Program

Structure of StarterPack6.sb
The Scratch Program in StarterPack6.sb has

1. A Sprite named “Karel”. Karel is a Robot.
2. Sprites named “Box 0”, “Box 1”, “Box 2”, “Box 3”, “Box 4” and “Box 5”
3. Sprites named “Signal 0”, “Signal 1”, “Signal 2”, “Signal 3”, “Signal 4”, “Signal 5”
4. A Sprite named “Engine”
5. Sprites named “Train 0”, “Train 1”, “Train 2”, “Train 3”, “Train 4”, “Train 5”
6. 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 goin east to west and south to south. The intersection of 2 roads is called a “corner”.

Karel’s City now has a Train and railway tracks.

The “The Karel Express” goes goes around the city loading and unloading freight boxes. The “Karel Express” goes counter clockwise around the city.

The Train has an

- Engine
- 6 Train Coaches numbered from Train 0 to Train 5

The railway system is controlled by 4 signals.

The signals are at the intersection of the railway tracks.

Each signal controls a certain segment of the track.

Use Signals to control the Train

The railway system is controlled by 4 signals.

The signals are at the intersection of the railway tracks.

Each signal controls a certain segment of the track.
Signal 0 controls the track at the bottom.
Signal 1 controls the track on the right.
Signal 2 controls the track on the top.
Signal 3 controls the track on the left.

The signals use the following color codes

Red indicates the train should stop if the train is on the track controlled by the signal
Green indicates that the train should speed up if the train is on the track controlled by the signal.
Yellow indicates that the train should slow down if the train is on the track controlled by the signal

In this Handout...

In this Handout, we will..

- Continue with Block programming
- Be Introduced to Functions

About Our Program

What should our Program do?

1. Our program should start when the green flag is clicked.
2. The Signals control the Train

3. When the Signal turns green, the Train should start if on the signal track
4. When the Signal turns yellow, the Train should slow down if on the signal track
5. When the Signal turns red, the Train should stop if its on the signal track

Remember

Each signal controls a certain segment of the track.
Signal 0 controls the track at the bottom.
Signal 1 controls the track on the right.
Signal 2 controls the track on the top.
Signal 3 controls the track on the left.

Class Activity in this Handout

We will program the following
At each Signal

1. The Engine should slow down
2. Turn counter-clockwise by 90 degrees onto the next track
3. Request a Signal from the signal controlling the next track
4. Update the Engine speed based on the signal.
   If the signal controlling the next track is
   a. green, the engine should speed up if necessary.
   b. yellow, the engine should slow down if necessary
   c. red, the engine should stop if necessary

Steps #1, #2 and #3 and #4 will be completed using Functions.

Introduction to Functions

What is a Function?

Functions are small programs that

- accept some inputs
- carry out a specific task
- may not may not return a value

Functions in SCRATCH are created using the More Blocks Palette.

Functions are created for a Sprite.

The functions we will be creating today are all for the Engine Sprite.

1. Click on the Engine Sprite
2. Click on the More Blocks Palette.
3. The Palette has the Make a Block button.
   The Make a Block button define a new block that
   - accept some inputs
   - carry out a specific task
   - does not return a value

   In other words, the new block is a function.

The new blocks created using the More Blocks Palette act as statement blocks.

Functions for the Engine Sprite

Functions to Turn the Engine

Function	What should this function do?
TURN-AT-SIGNAL0	Turn the Engine at Signal 0
TURN-AT-SIGNAL1	Turn the Engine at Signal 1
TURN-AT-SIGNAL2	Turn the Engine at Signal 2
TURN-AT-SIGNAL3	Turn the Engine at Signal 3

Functions TURN-AT-SIGNAL0, TURN-AT-SIGNAL1 and TURN-AT-SIGNAL2 will be completed in class.

Function TURN-AT-SIGNAL0 will need to be completed as part of Homework6.

Function to request a signal update

Function	What should this function do?
REQUEST-SIGNAL	Request the signal for the next track

Function REQUEST-SIGNAL will be completed in class

Function to check and correct Engine Speed

Function	What should this function do?
CHECK-SPEED	Check and correct the engine speed as per the signal

Functions for the Signal Sprite

Function to update the signal

Function	What should this function do?
UPDATE-SIGNAL	Updates the SIGNAL-ACTION and SIGNAL-DIRECTION based on the current costume #

New Blocks

The following new blocks will be used

Block Palette	Block	What does it do?
Sensing Blocks	x position of <Sprite>	Expression Block Returns the value of the current x position of the Sprite
	y position of <Sprite>	Expression Block Returns the value of the current y position of the Sprite
	TRAIN-DIRECTION of <Sprite>	Expression Block Returns the value of the TRAIN-DIRECTION of the Sprite1
Events Blocks	broadcast <message> and wait	Broadcast a message to another Script. This Script with this block will then wait until the ALL Scripts receiving the message completes.

Defining the CHECK-SPEED Function

The first function we will define is CHECK-SPEED.

CHECK-SPEED will do the following

1. It takes no inputs
2. It will compare the Train Direction with the Signal Direction
3. If the Train Direction is the same as the Signal Direction, update the Train Speed

The following code will define a new block called CHECK-SPEED

1. Click on the Engine Sprite
2. Click on More Blocks
3. Click on Make a Block
4. This will popup the Block Editor
5. Type in the name CHECK-SPEED
6. Click OK

A new block called CHECK-SPEED will appear in the Script Editor.

The CHECK-SPEED is a function. As of now, it does nothing. We can add blocks to it to make it do something

1. Click on the Engine Sprite
2. Click the Control Palette
3. Drag the If-Then-Condition Block to the Script Editor
4. Update the If-Then-Condition Block to check for the condition

```
if TRAIN-DIRECTION=SIGNAL-DIRECTION then
```

5. Drag the following blocks which are currently under When I receive SIGNAL-CHANGE into If-Then-Else Condition Block (These blocks were created in the Handout4)

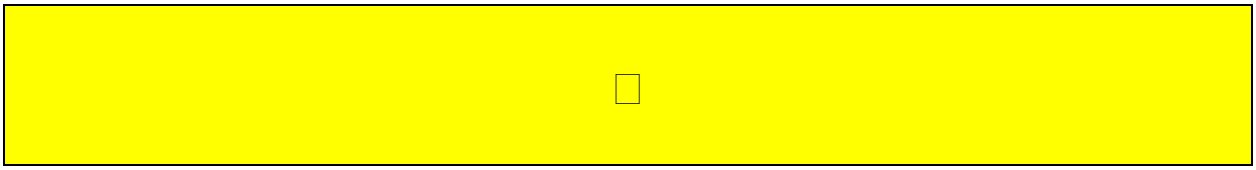
```
if SIGNAL-ACTION=SPEED-UP then
        set TRAIN-SPEED to 40
if SIGNAL-ACTION=SLOW-DOWN then
        set TRAIN-SPEED to 20
```

```
if SIGNAL-ACTION=STOP then
    set TRAIN-SPEED to 0
```

That's it. We have completed the new function CHECK-SPEED. Great!
The CHECK-SPEED will do the following

1. It takes no inputs
2. It will compare the Train Direction with the Signal Direction
3. If the Train Direction is the same as the Signal Direction, update the Train Speed

Save your changes.



Defining the REQUEST-SIGNAL Function

The next function we will define is REQUEST-SIGNAL Function.

REQUEST-SIGNAL will do the following

1. It takes no inputs
2. It will use the following broadcast block to broadcast a SIGNAL-REQUEST message and wait

```
broadcast <message1> and wait
```

This block is different from the broadcast block we saw earlier.
This block

- broadcasts a message similar to the other broadcast block
- however it waits until all receiver scripts have finished executing

In this particular case, our receiver scripts are in the Signal sprites
(Signal 0, Signal 1, Signal 2, Signal 3)

We will add receiver Scripts to listen to the SIGNAL-REQUEST

3. It then executes the statement block CHECK-SPEED block.

The following code will define the new block called REQUEST-SIGNAL

1. Click on the Engine Sprite
2. Click on More Blocks
3. Click on Make a Block
4. This will popup the Block Editor
5. Type in the name REQUEST-SIGNAL
6. Click OK

A new block called REQUEST-SIGNAL will appear in the Script Editor.
The REQUEST-SIGNAL block is a function. As of now, it does nothing. We can add blocks to it to make it do something

1. Click on the Engine Sprite
2. Click on the Events Palette
3. Drag the following block under the REQUEST-SIGNAL block

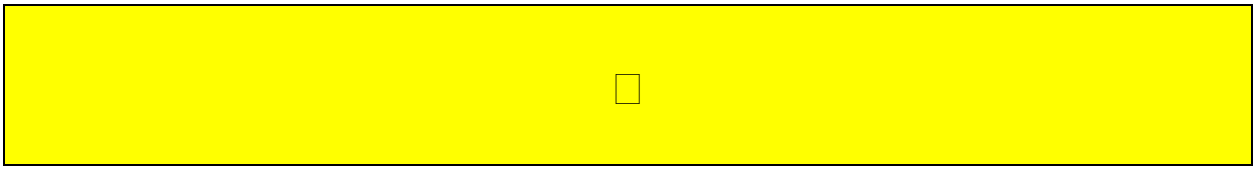
Change the broadcast message to the new message SIGNAL-REQUEST

4. Click on the More Blocks Palette
5. Drag the following block under the previous block

That's it. We have completed the new function REQUEST-SIGNAL. Great!
The REQUEST-SIGNAL will do the following

1. It takes no inputs
2. It will broadcast the SIGNAL-REQUEST message and wait until all receiver scripts have completed execution
3. Execute the CHECK-SPEED block which we defined earlier

Save your changes.



Defining the UPDATE-SIGNAL Function

The next function we will define is UPDATE-SIGNAL Function.
The UPDATE-SIGNAL Function is defined for each of the Signal Sprites

UPDATE-SIGNAL will do the following

1. It takes no inputs
2. It will compare the costume #
3. If the costume # is 1, it will set the SIGNAL-ACTION to STOP
4. If the costume # is 2, it will set the SIGNAL-ACTION to SPEED-UP
5. If the costume # is 3, it will set the SIGNAL-ACTION to SLOW-DOWN
6. Sets the SIGNAL-DIRECTION as per the following table

Signal	Which track does it control?
Signal 0	RIGHT
Signal 1	UP
Signal 2	LEFT
Signal 3	DOWN

The following code will define the new block called UPDATE-SIGNAL for Signal 0

1. Click on the Signal 0 Sprite
2. Click on More Blocks
3. Click on Make a Block
4. This will popup the Block Editor
5. Type in the name UPDATE-SIGNAL
6. Click OK

A new block called UPDATE-SIGNAL will appear in the Script Editor.
The UPDATE-SIGNAL block is a function. As of now, it does nothing. We can add blocks to it to make it do something

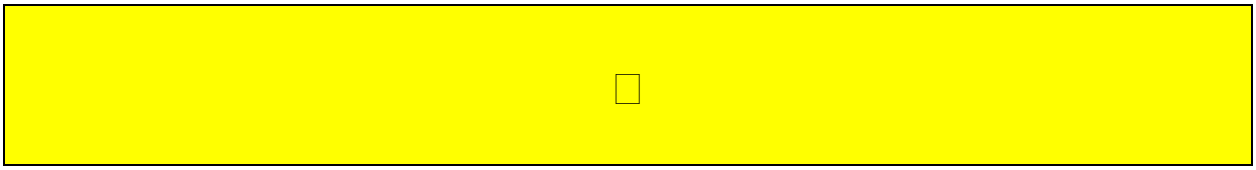
1. Click on the Signal 0 Sprite
2. Drag the 3 If-Then condition blocks for the costume # from the When this sprite clicked block into the UPDATE-SIGNAL block. These blocks update the SIGNAL-ACTION.
3. Drag the Set SIGNAL-DIRECTION block from the When this Sprite clicked block at the end of the UPDATE-SIGNAL block

That's it. We have completed the new function UPDATE-SIGNAL. Great!
The UPDATE-SIGNAL will do the following

1. It takes no inputs
2. It will update the SIGNAL-ACTION and SIGNAL-DIRECTION based on the costume #

Make the corresponding changes for Signal 1, Signal 2, and Signal 3 by following the steps above

Save your changes.



Add the Signal receiver Scripts

We need to add receiver scripts to the Signal sprites (Signal 0, Signal 1, Signal 2, Signal 3)

These receiver scripts should

1. listen to the SIGNAL-REQUEST message
2. Compare the Engine direction to the signal direction (to check if the Engine is on the signal track)
3. Update the Signal by executing the UPDATE-SIGNAL block

The receiver scripts should be added to each of the Signal sprites

1. Click on Signal 0
2. Click on the Events Palette

3. Drag the following block onto the Script Editor

When I receive <message1>

4. Toggle the drop down. Change the message from <message1> to SIGNAL-REQUEST
5. Add the If-Condition-Block to compare the TRAIN-DIRECTION of Engine to the Signal Direction

if TRAIN-DIRECTION of Engine =RIGHT then

The Signal direction for Signals are as follows

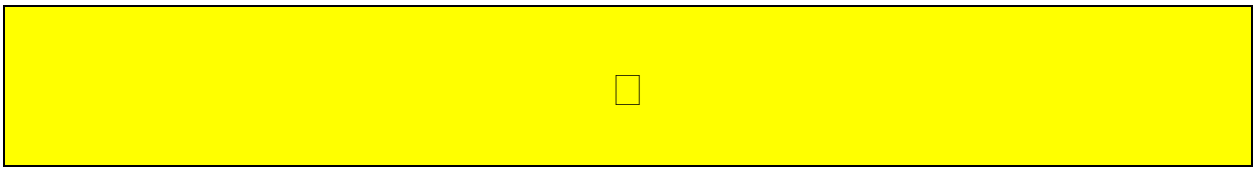
Signal	Which track does it control?
Signal 0	RIGHT
Signal 1	UP
Signal 2	LEFT
Signal 3	DOWN

6. Click on More Blocks
7. Drag the UPDATE-SIGNAL block into the If-Then-Condition block above

That's it. We have completed the receiver script to listen to SIGNAL-REQUEST messages for Signal 0.

Make the corresponding changes for Signal 1, Signal 2, and Signal 3 by following the steps above

Save your changes.



Defining the TURN-AT-SIGNAL Functions

The next function we will define are the TURN-AT-SIGNAL Functions.
The TURN-AT-SIGNAL functions are defined for the Engine Sprite

Function	Which does this do?
TURN-AT-SIGNAL0	Handles the turn at Signal 0
TURN-AT-SIGNAL1	Handles the turn at Signal 1
TURN-AT-SIGNAL2	Handles the turn at Signal 2
TURN-AT-SIGNAL3	Handles the turn at Signal 3

Each TURN-AT-SIGNAL will do the following

1. It takes no inputs
2. Update the x position of the Engine to the x position of the Signal
3. Update the y position of the Engine to the y position of the Signal
4. Point in the correct direction
5. Moves along the track by 18 steps
6. Updates the TRAIN-DIRECTION
7. Sets the X variable to the new x position
8. Sets the Y variable to the new y position
9. Executes the REQUEST-SIGNAL block to request the signal controlling the track to let the Engine know what to update its speed to

The following code will define the new block called TURN-AT-SIGNAL0

1. Click on the Engine Sprite
2. Click on More Blocks
3. Click on Make a Block
4. This will popup the Block Editor
5. Type in the name TURN-AT-SIGNAL0
6. Click OK

A new block called TURN-AT-SIGNAL0 will appear in the Script Editor.
The TURN-AT-SIGNAL0 block is a function. As of now, it does nothing. We can add blocks to it to make it do something

1. Click on the Engine Sprite
2. Click on Motion Palette
3. Drag the set x to block into the TURN-AT-SIGNAL0 block.
4. Click on Sensing Palette
5. Drag the x position of <Sprite1> block. Click on the Sprite 1 toggle and change it to Signal 0
6. Drop the x position of <Sprite1> block above into the set x to block slot.
7. Do the steps 3 to 6 using the set y to block and the y position of Signal 0
8. Click on Motion Palette
9. Drag the point in direction block
10. Toggle the direction to 0 (Up)
11. Click on Motion Palette
12. Drag the change y by block into the TURN-AT-SIGNAL0 block. Update the value to 18 (Going up)
13. Click on the Data Palette
14. Drag the set variable to block into the TURN-AT-SIGNAL0 block.
15. Toggle the variable to TRAIN-DIRECTION
16. Drop the UP variable into the slot
17. Click on the Data Palette
18. Drag the set variable to into the TURN-AT-SIGNAL0 block.
19. Toggle the variable to X
20. Click on the Motion Palette
21. Drag the x position into the slot
22. Do the steps 18 to 21 using the Y variable and the y position
23. Click on the More Blocks Palette
24. Drag the REQUEST-SIGNAL block into the TURN-AT-SIGNAL0 block.

Whew! That was a lot of steps

That's it. We have completed the new function TURN-AT-SIGNAL0. Great!

Test the TURN-AT-SIGNAL Function

Test your changes to TURN-AT-SIGNAL0

1. Click the Green Flag to begin
2. Click on the Signal 0 to toggle
3. The Engine should proceed along the track moving to the RIGHT, turn UP at the Signal 0 and proceed up the Track
4. Click on Signal 1. Signal 1 should turn YELLOW

5. The Engine should slow down
6. Click on Signal 1 again when the Engine is still moving UP. The Signal 1 should turn RED
7. The Engine should stop

Add the other TURN-AT-SIGNAL Functions

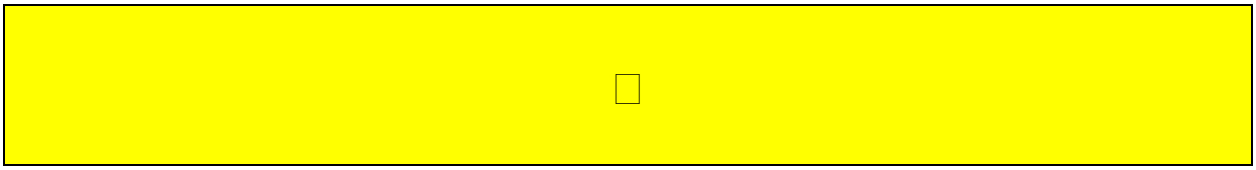
Follow the steps above to define the functions for TURN-AT-SIGNAL1 and TURN-AT-SIGNAL3

Use the table below

Function Name	x position	y position	point	change	TRAIN-DIRECTION
TURN-AT-SIGNAL0	Signal0	Signal0	0 (Up)	y by 18	UP
TURN-AT-SIGNAL1	Signal1	Signal1	-90 (Left)	x by -18	LEFT
TURN-AT-SIGNAL2	Signal2	Signal2	180 (Down)	y by -18	DOWN
TURN-AT-SIGNAL3	Signal3	Signal3	90 (Right)	x by 18	RIGHGT

You will complete TURN-AT-SIGNAL3 as part of your homework

Save your changes.



That was a LOT we covered!

You made it this far! Awesome!

We will cover **Block Programming Part II** in the next class class **building on the concepts we covered in this class**. For now you need to make sure you have a good conceptual understand of **Block Programming Part I**.

Quiz 6: Block Programming (Part I)

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 Quiz6.odt under “Courses” “TA-SCR-1” “quiz” “quiz6”

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 Quiz and Homework” in this Handout.

Homework 6: Block Programming (Part I)

Make sure you read this Handout!

Overview

In this Homework you will use a Variable to keep track of the number of markers Karel has touched while navigating from Marker 0 to Marker 5. Remember that Karel need not touch all markers to navigate from Marker 0 to Marker 5.

Open the Homework

Follow the instructions in “Updating the Course” in this Handout.
Open Homework6.sb under “Courses” “TA-SCR-1” “homework” “homework6”

- Select “Homework6.sb”
- Right Click, Select Open With Scratch 2 OR
- Double click the file

Complete the Homework

You will need to refer to this handout (Handout6). Make sure you read it thoroughly.

Complete TURN-AT-SIGNAL3 function. The function is similar to TURN-AT-SIGNAL0. Refer to section Defining the TURN-AT-SIGNAL Functions for steps to define the function. Use the table in the section Add the other TURN-AT-SIGNAL Functions to see the values for the function

Ensure that all other TURN-AT-SIGNAL functions (TURN-AT-SIGNAL0, TURN-AT-SIGNAL1, TURN-AT-SIGNAL2) are also completed.

Test your TURN-AT-SIGNAL functions

1. Click on the green flag
2. Click Signal 0

3. The Engine should move along the track and turn up at Signal 0
4. It should similarly turn LEFT at Signal 1, DOWN at Signal 2
5. If your function is correct, it should turn RIGHT at Signal 3.

Submit the Homework

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