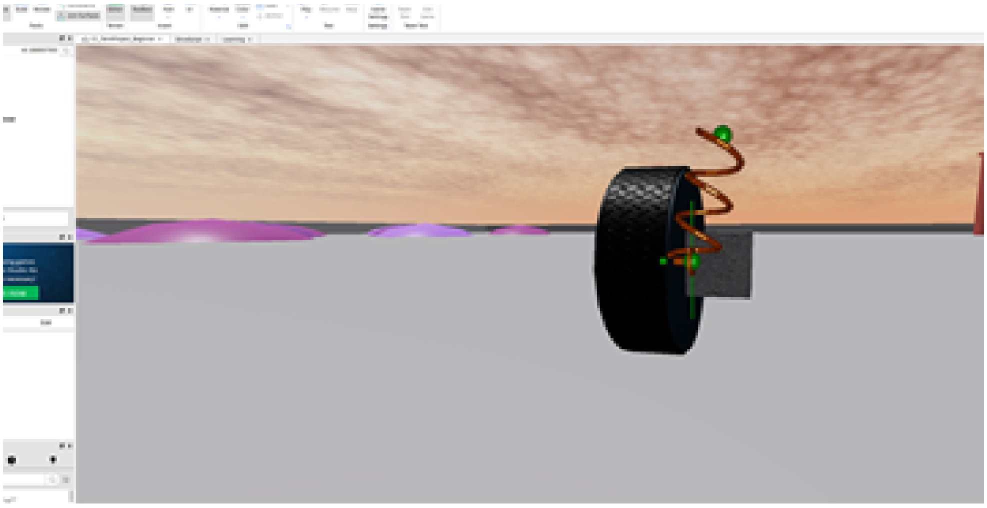
# ROBLOX LESSON 7

TERM PROJECT

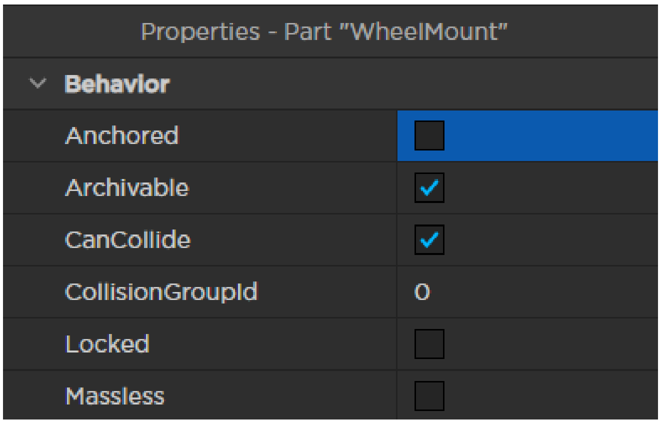
CodeCreate

## TERM PROJECT: RACING GAME

1. Continuing from last week, let's move the wheel module together using the move tool so our wheel is right next to the wheel mount.



2. Let's test our spring's settings by anchoring the spring (tick the checkbox).

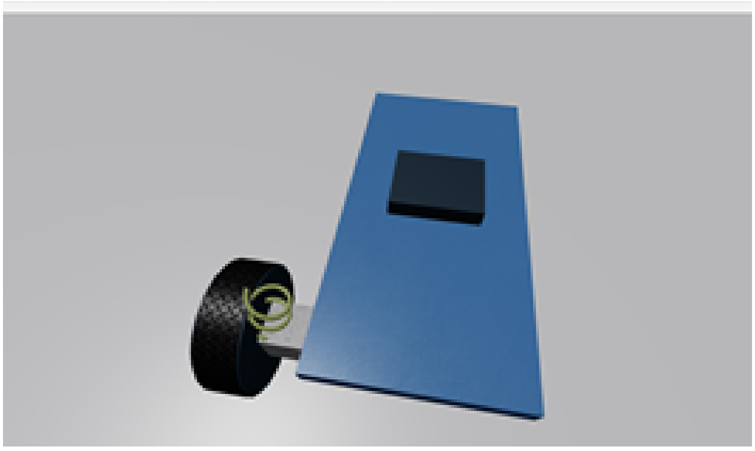


Properties - Part "WheelMount"

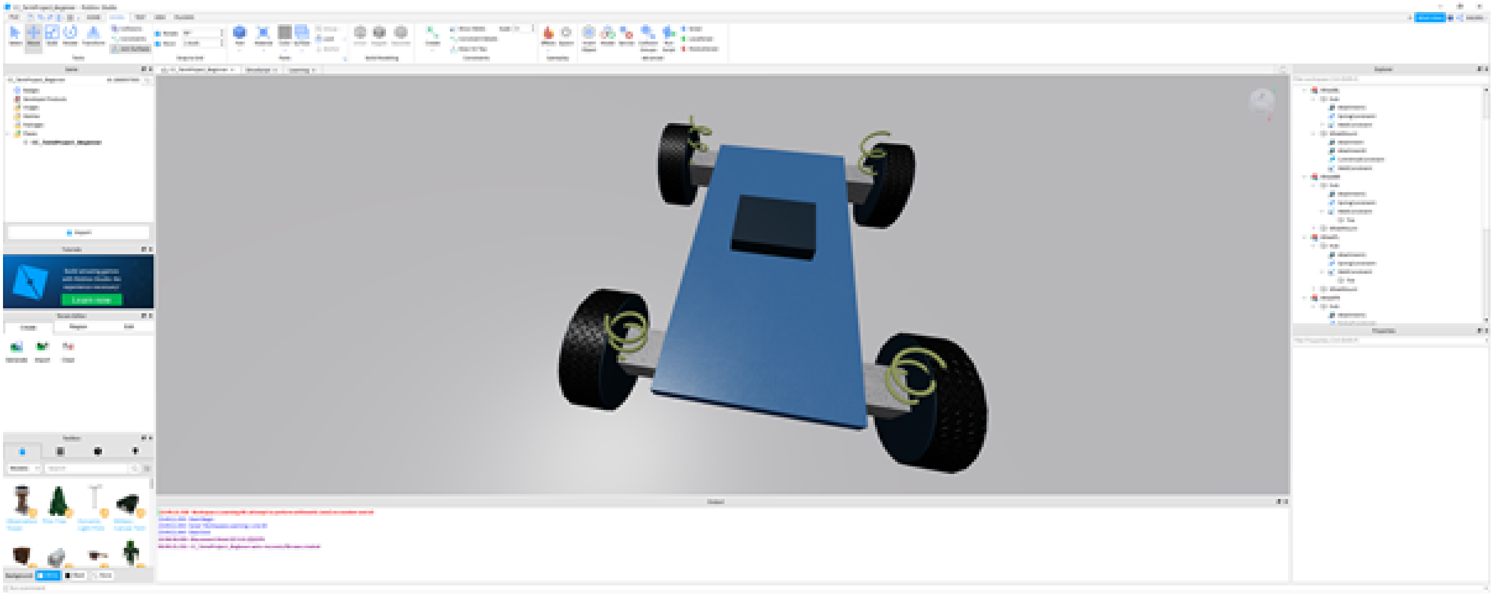
Behavior

| | |
|---|---|
| Anchored | □ |
| Archivable | ☑ |
| CanCollide | ☑ |
| CollisionGroupId | 0 |
| Locked | □ |
| Massless | □ |

3. Once you're satisfied with the "springiness", continue to the next step.Next, we should move the wheels next to the base using the move tool..

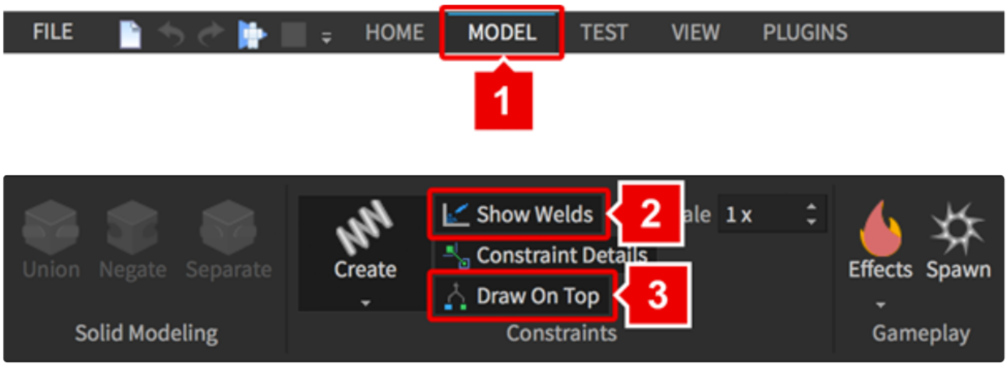


4. Next, we can duplicate this wheel four times to give our four driving wheels.

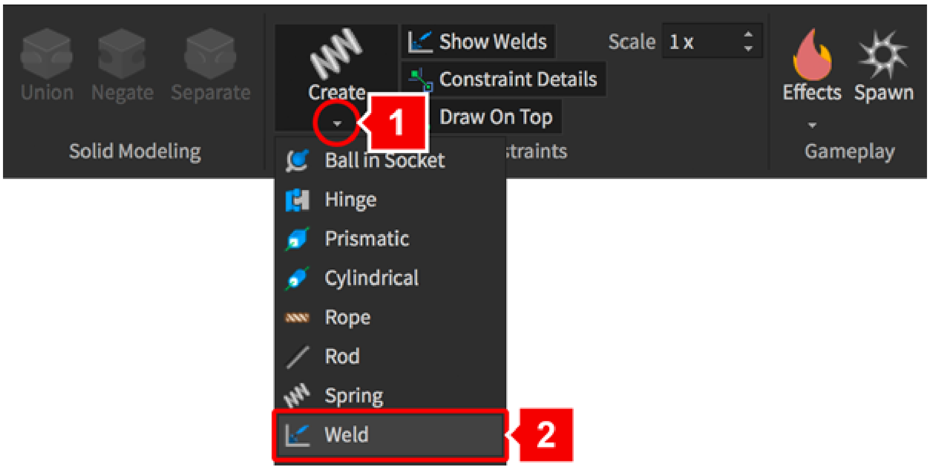


5. Next, let's weld the wheel mounts to the base so they don't fall off when we start driving!

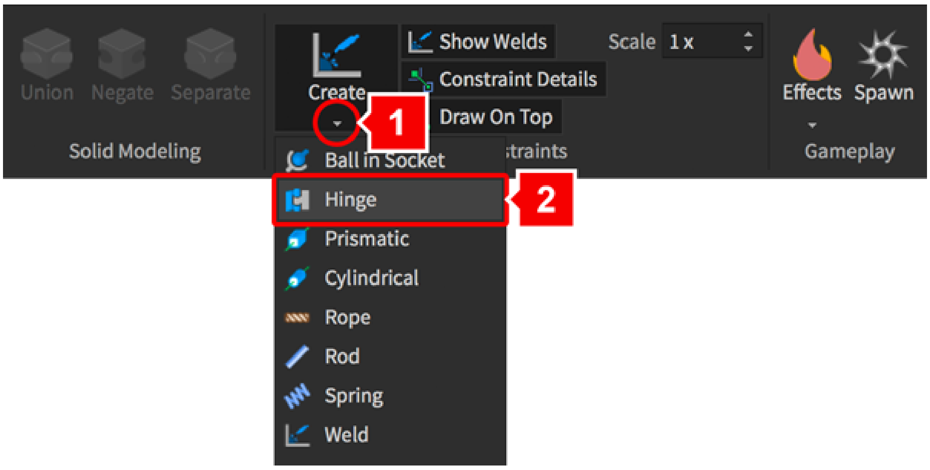
Protip: Before we start, let's show the weld details so we know what we're working with. In the Models tab, check show welds.



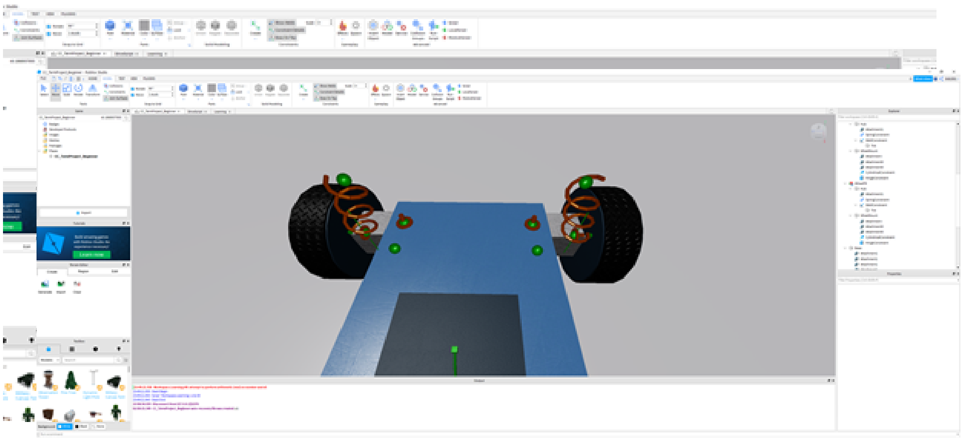
6. Select welds from create. Click the wheel mount, then the blue base. This should create a single weld. Repeat this for both back wheels (wheels further from seat).



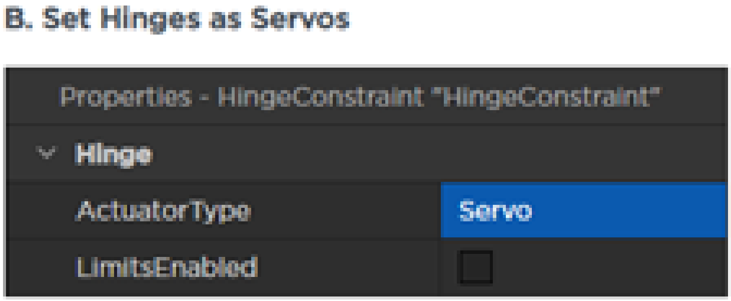
7. Next, we should hinge the front wheels. Let's select the hinge constraint from the model tab.



Then select the wheel mount and the closest attachment point on the blue base. It should look like this:



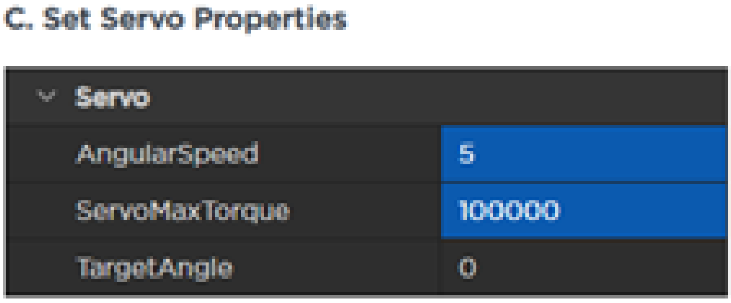
8. We should select both hinges in the Explorer and set their properties. We should change the default properties to the following.



B. Set Hinges as Servos

Properties - HingeConstraint "HingeConstraint"

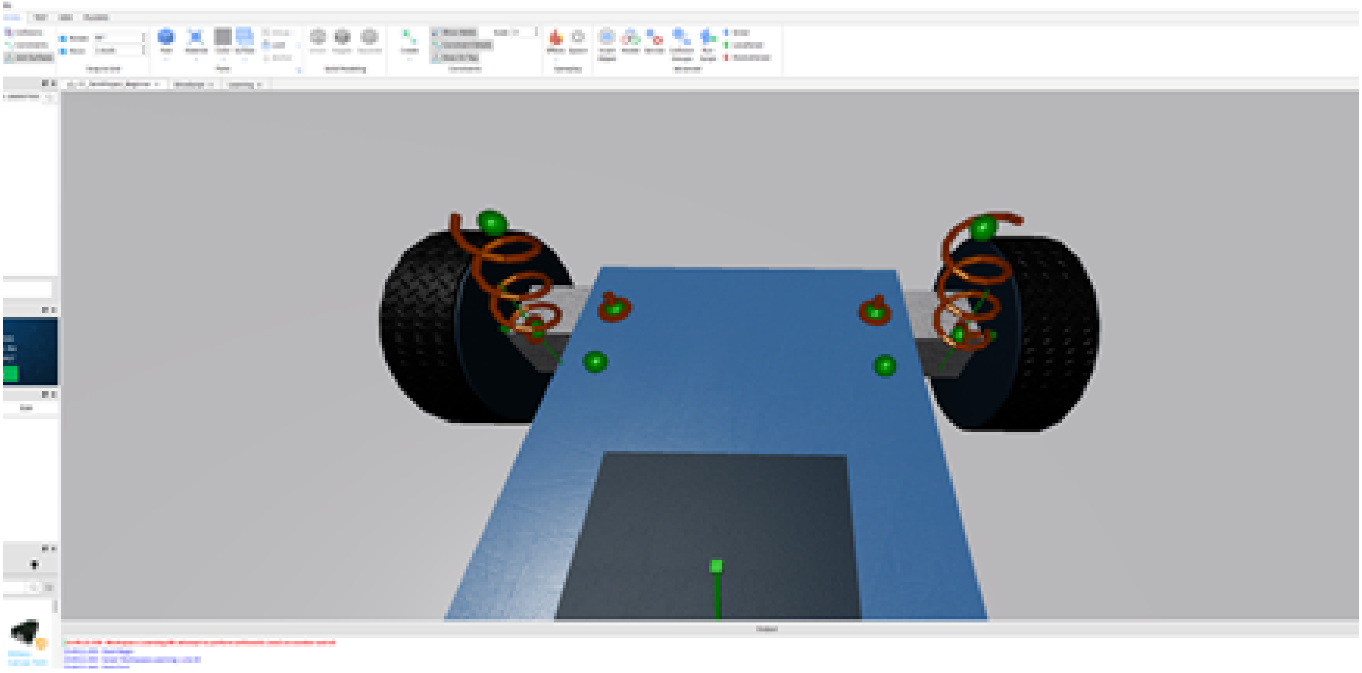
| Hinge | |
|---|---|
| ActuatorType | Servo |
| LimitsEnabled | □ |



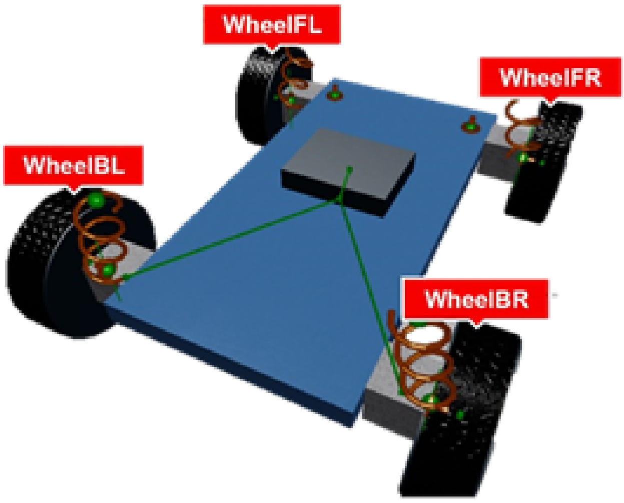
C. Set Servo Properties

| Servo | |
|---|---|
| AngularSpeed | 5 |
| ServoMaxTorque | 100000 |
| TargetAngle | 0 |

9. Finally, let's move the hinge attachment points closer to the base so that the car turns better. Move the attachment point on the wheel mount one click towards the blue base. You should end up with the following result.



10. Don't forget to rename our four wheels to: WheelBL, WheelBR, WheelFR, WheelFL.

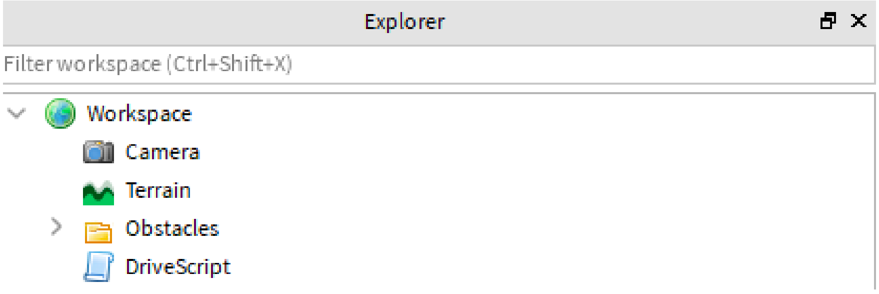


Have fun customising your vehicle!

Try adding colours, shapes, changing springiness or other components we've covered in this course.

# TERM PROJECT: PT 2 CODING IN LUA

1. Navigate to your DriveScript file under the project Workspace.

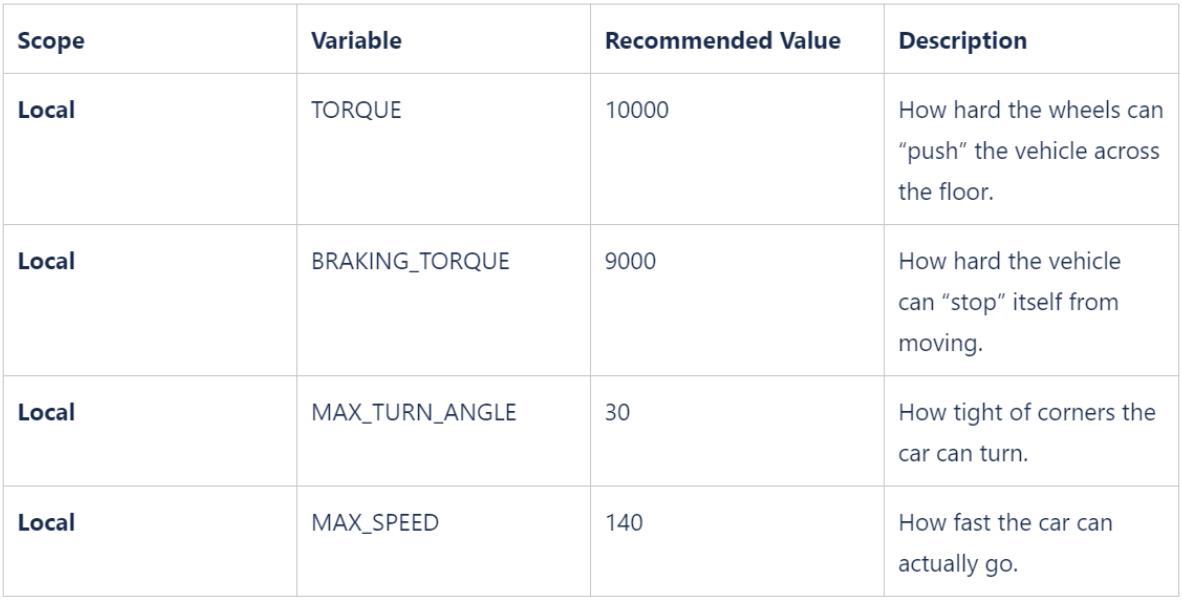


Once this is done, we can enter the template file and start creating variables!

2. In the **TUNING VALUES** section, we should enter the parameters of the vehicle.

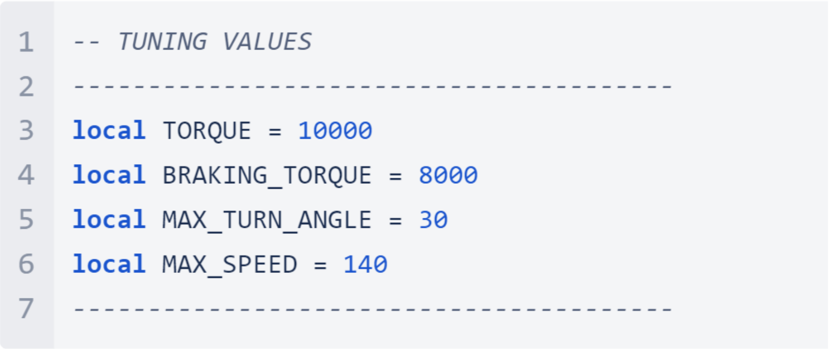
These are the numbers which determine how fast the vehicle can go, how it turns corners and much more!

Start by creating the variables and then setting them to the recommended values in the table below:



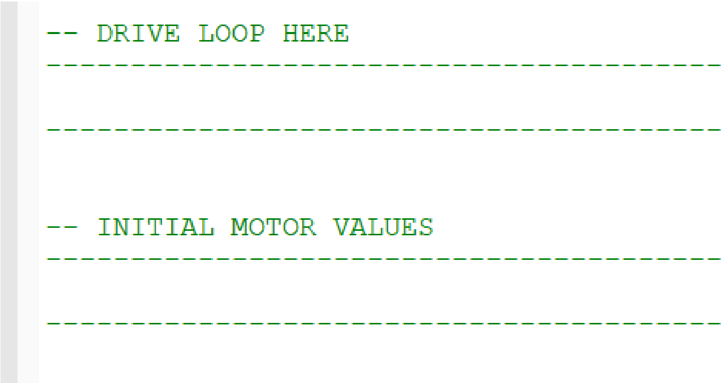
| Scope | Variable | Recommended Value | Description |
|---|---|---|---|
| **Local** | TORQUE | 10000 | How hard the wheels can “push” the vehicle across the floor. |
| **Local** | BRAKING_TORQUE | 9000 | How hard the vehicle can “stop” itself from moving. |
| **Local** | MAX_TURN_ANGLE | 30 | How tight of corners the car can turn. |
| **Local** | MAX_SPEED | 140 | How fast the car can actually go. |

Once that's done, you're code should look something like this:



```lua
-- TUNING VALUES
----------------------------------------
local TORQUE = 10000
local BRAKING_TORQUE = 8000
local MAX_TURN_ANGLE = 30
local MAX_SPEED = 140
----------------------------------------
```

4. Under the **DRIVE LOOP** section, create a new Comment called:  **INITIAL MOTOR VALUES**



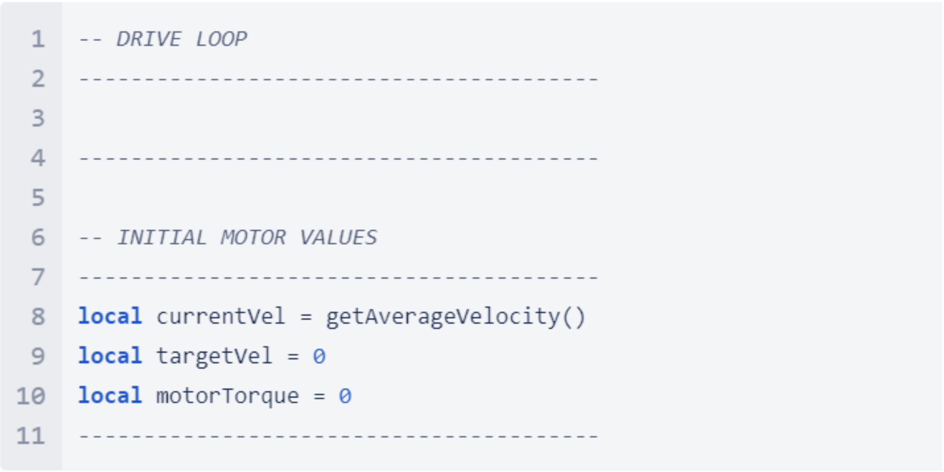
```
-- DRIVE LOOP HERE
-----------------------------------------

-----------------------------------------


-- INITIAL MOTOR VALUES
-----------------------------------------

-----------------------------------------
```

5. Now we want to create a bunch of **local variables** to describe the car's current velocity and motor properties.

Start by creating the variables and then setting them to the recommended values in the table below:

| **Scope** | **Variable** | **Recommended Value** | **Description** |
|---|---|---|---|
| **Local** | currentVel | getAverageVelocity() | This function will be provided to students. An explanation for the parenthesis can include "it tells our program to run a separate piece of code that tells us the average velocity of the car". |
| **Local** | targetVel | 0 | This variable sets the initial target velocity for the motor to 0. |
| **Local** | motorTorque | 0 | This variable sets the initial torque of the motor to 0. |

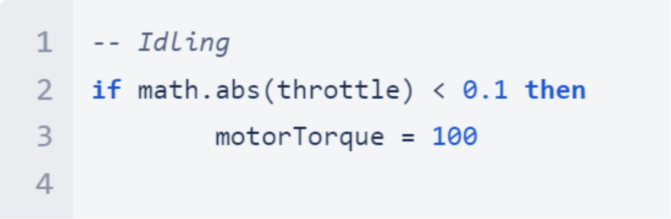
6.Once our variables are added, our program should look like this:



```
-- DRIVE LOOP
----------------------------------------

----------------------------------------


-- INITIAL MOTOR VALUES
----------------------------------------
local currentVel = getAverageVelocity()
local targetVel = 0
local motorTorque = 0
----------------------------------------
```

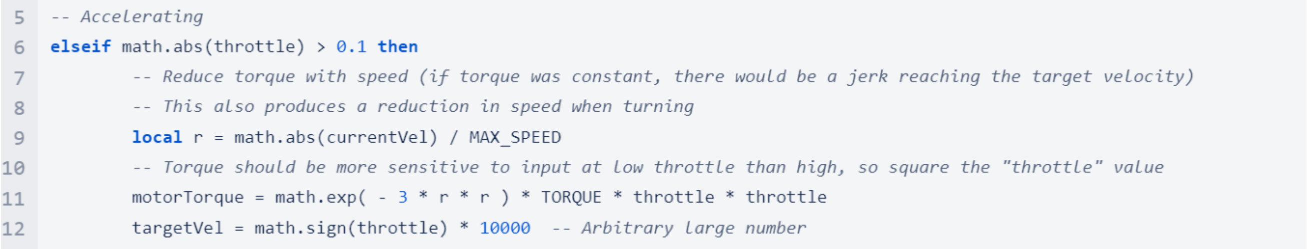
4. Finally, lets add some if/else statements which tell the car whether it should idle, increase speed, or decrease speed.

- **If** the **throttle** (the value which controls whether we want the motors to start or stop) **is less than 0.1:**
  - we should **set the torque** (how hard the wheels push the car across the floor) **to a small number,** namely 100. We will call this "idling".



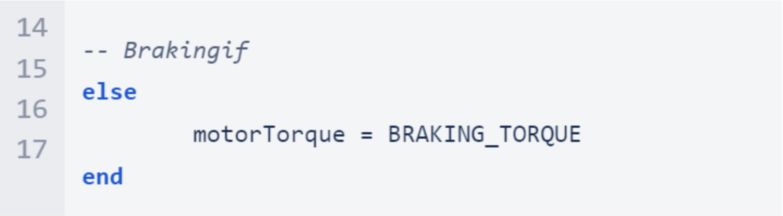
```lua
-- Idling
if math.abs(throttle) < 0.1 then
        motorTorque = 100

```

- **If** the **throttle** is **greater than 0.1:**
  - we should **set the torque** to a **high number** to get the car moving once the target speed is set.
  - **Copy** the **else if** statement from the end of the file in **--ACCELERATION CODE** section



```lua
-- Accelerating
elseif math.abs(throttle) > 0.1 then
        -- Reduce torque with speed (if torque was constant, there would be a jerk reaching the target velocity)
        -- This also produces a reduction in speed when turning
        local r = math.abs(currentVel) / MAX_SPEED
        -- Torque should be more sensitive to input at low throttle than high, so square the "throttle" value
        motorTorque = math.exp( - 3 * r * r ) * TORQUE * throttle * throttle
        targetVel = math.sign(throttle) * 10000  -- Arbitrary large number
```

- Finally to break the car, we set the **motor torque** to **equal** BRAKING_TORQUE in the **else statement**.

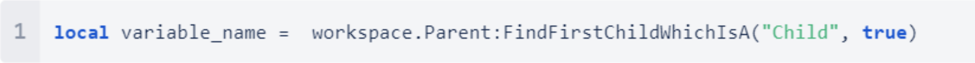


```lua
-- Brakingif
else
        motorTorque = BRAKING_TORQUE
end
```

- Don't forget to **end** your code.

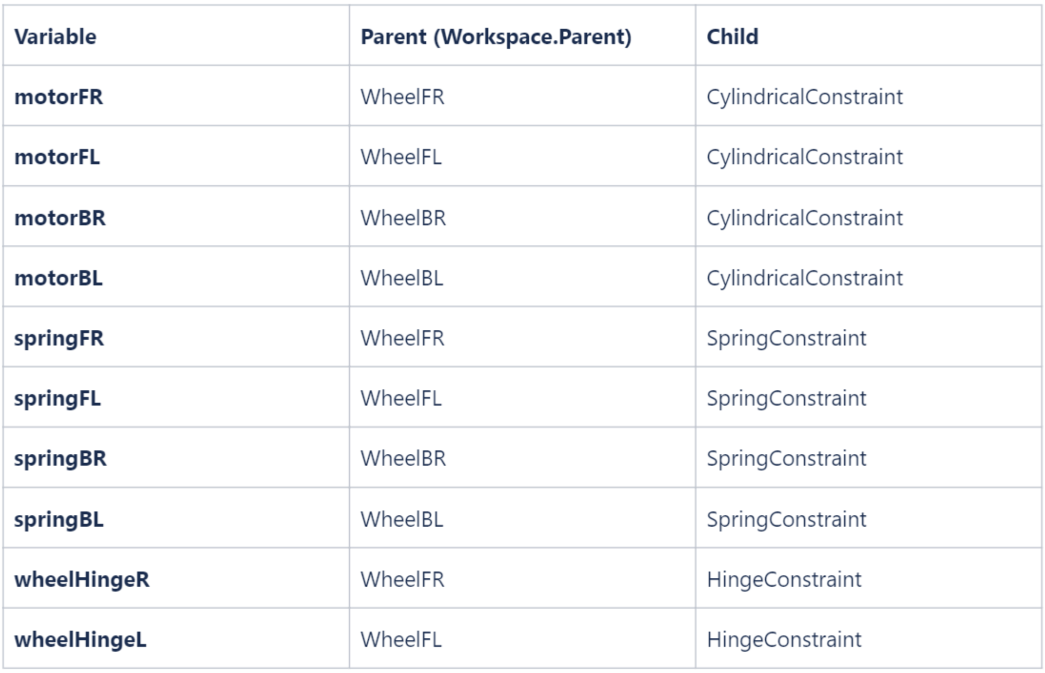
5. Next we want to initialise variables relating to the physical components of our car (Wheels, Motors).

The format for declaring variables in Lua is:



```lua
local variable_name =  workspace.Parent:FindFirstChildWhichIsA("Child", true)
```

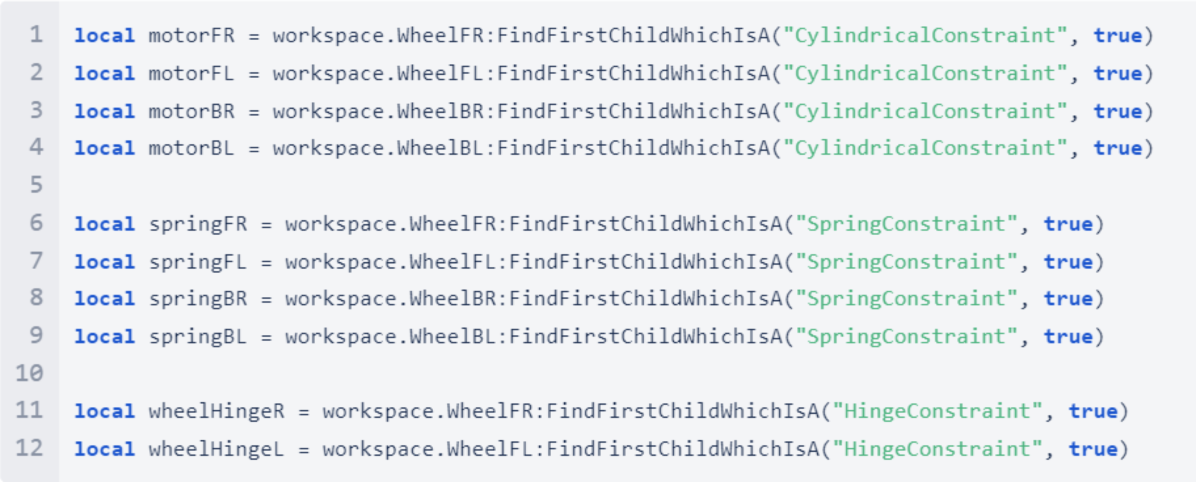
Have a go at declaring and initialising the variables in the table below. You should do this at the very top of your **DriveScript** under the **vehicleSeat variable.**



| Variable | Parent (Workspace.Parent) | Child |
|---|---|---|
| **motorFR** | WheelFR | CylindricalConstraint |
| **motorFL** | WheelFL | CylindricalConstraint |
| **motorBR** | WheelBR | CylindricalConstraint |
| **motorBL** | WheelBL | CylindricalConstraint |
| **springFR** | WheelFR | SpringConstraint |
| **springFL** | WheelFL | SpringConstraint |
| **springBR** | WheelBR | SpringConstraint |
| **springBL** | WheelBL | SpringConstraint |
| **wheelHingeR** | WheelFR | HingeConstraint |
| **wheelHingeL** | WheelFL | HingeConstraint |

6. After declaring all the variables, our code should look like this:

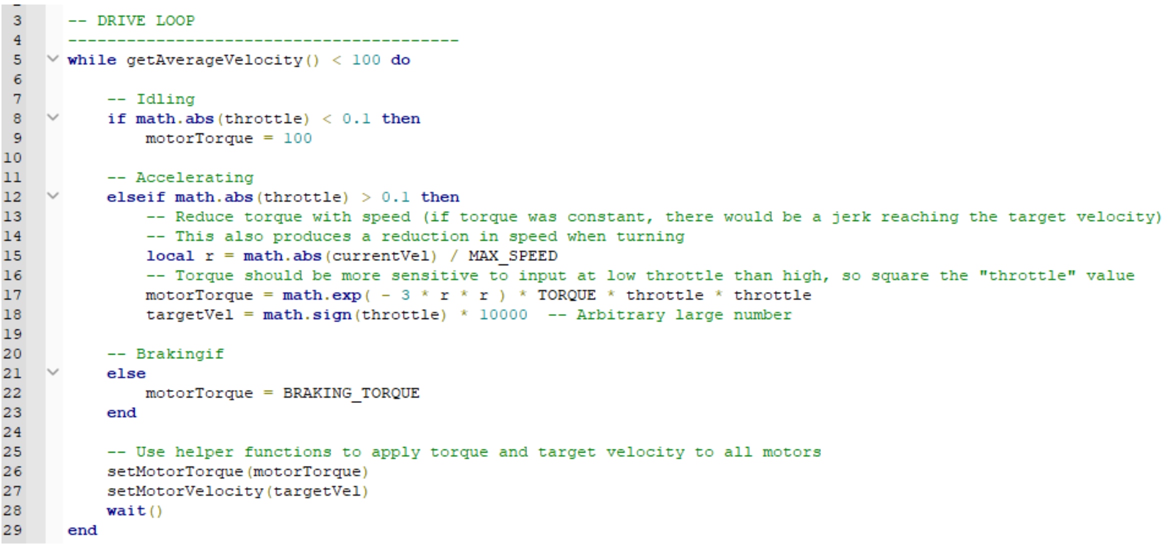
local motorFR = workspace.WheelFR: FindFirstChildWhichIsA("CylindricalConstraint", true)



```lua
local motorFR = workspace.WheelFR:FindFirstChildWhichIsA("CylindricalConstraint", true)
local motorFL = workspace.WheelFL:FindFirstChildWhichIsA("CylindricalConstraint", true)
local motorBR = workspace.WheelBR:FindFirstChildWhichIsA("CylindricalConstraint", true)
local motorBL = workspace.WheelBL:FindFirstChildWhichIsA("CylindricalConstraint", true)

local springFR = workspace.WheelFR:FindFirstChildWhichIsA("SpringConstraint", true)
local springFL = workspace.WheelFL:FindFirstChildWhichIsA("SpringConstraint", true)
local springBR = workspace.WheelBR:FindFirstChildWhichIsA("SpringConstraint", true)
local springBL = workspace.WheelBL:FindFirstChildWhichIsA("SpringConstraint", true)

local wheelHingeR = workspace.WheelFR:FindFirstChildWhichIsA("HingeConstraint", true)
local wheelHingeL = workspace.WheelFL:FindFirstChildWhichIsA("HingeConstraint", true)
```

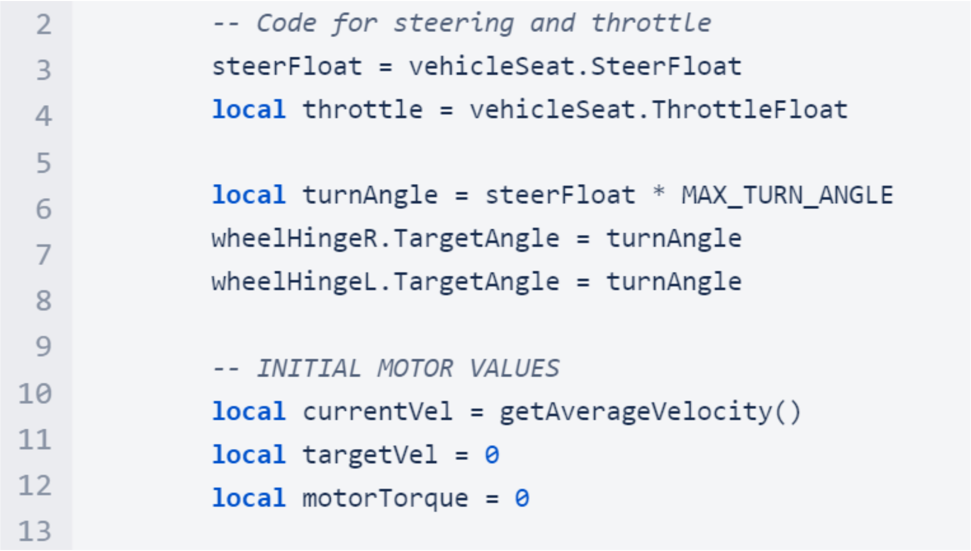
7. Next we will need to implement a while true loop to constantly keep track of the car's speed and whether the car should idle, speed up or slow down.

Place this while true loop around your **if statements** for speeding up/ slowing down:



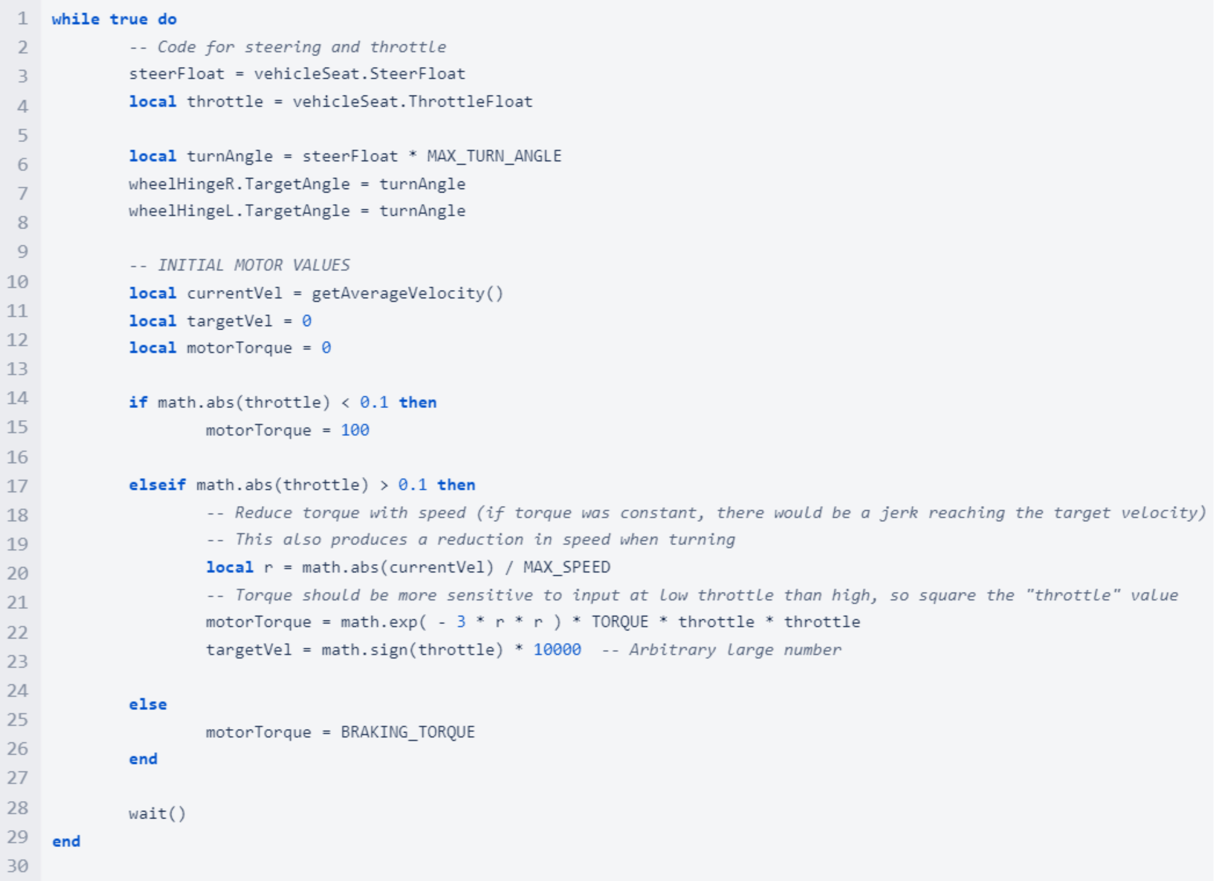
```
-- DRIVE LOOP
----------------------------------------
while getAverageVelocity() < 100 do

    -- Idling
    if math.abs(throttle) < 0.1 then
        motorTorque = 100

    -- Accelerating
    elseif math.abs(throttle) > 0.1 then
        -- Reduce torque with speed (if torque was constant, there would be a jerk reaching the target velocity)
        -- This also produces a reduction in speed when turning
        local r = math.abs(currentVel) / MAX_SPEED
        -- Torque should be more sensitive to input at low throttle than high, so square the "throttle" value
        motorTorque = math.exp( - 3 * r * r ) * TORQUE * throttle * throttle
        targetVel = math.sign(throttle) * 10000  -- Arbitrary large number

    -- Brakingif
    else
        motorTorque = BRAKING_TORQUE
    end

    -- Use helper functions to apply torque and target velocity to all motors
    setMotorTorque(motorTorque)
    setMotorVelocity(targetVel)
    wait()
end
```

8. Copy the code for steering and throttle from the bottom of the script into the while loop.  These variables will allow you to measure user input from the keyboard.



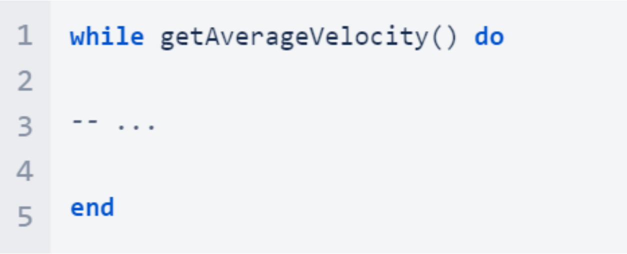
```
-- Code for steering and throttle
steerFloat = vehicleSeat.SteerFloat
local throttle = vehicleSeat.ThrottleFloat

local turnAngle = steerFloat * MAX_TURN_ANGLE
wheelHingeR.TargetAngle = turnAngle
wheelHingeL.TargetAngle = turnAngle

-- INITIAL MOTOR VALUES
local currentVel = getAverageVelocity()
local targetVel = 0
local motorTorque = 0
```

9. Your code should look like this:



```lua
while true do
    -- Code for steering and throttle
    steerFloat = vehicleSeat.SteerFloat
    local throttle = vehicleSeat.ThrottleFloat

    local turnAngle = steerFloat * MAX_TURN_ANGLE
    wheelHingeR.TargetAngle = turnAngle
    wheelHingeL.TargetAngle = turnAngle

    -- INITIAL MOTOR VALUES
    local currentVel = getAverageVelocity()
    local targetVel = 0
    local motorTorque = 0

    if math.abs(throttle) < 0.1 then
        motorTorque = 100

    elseif math.abs(throttle) > 0.1 then
        -- Reduce torque with speed (if torque was constant, there would be a jerk reaching the target velocity)
        -- This also produces a reduction in speed when turning
        local r = math.abs(currentVel) / MAX_SPEED
        -- Torque should be more sensitive to input at low throttle than high, so square the "throttle" value
        motorTorque = math.exp( - 3 * r * r ) * TORQUE * throttle * throttle
        targetVel = math.sign(throttle) * 10000  -- Arbitrary large number

    else
        motorTorque = BRAKING_TORQUE
    end

    wait()
end

```

10. We also wish to add a maximum speed for the car of 120

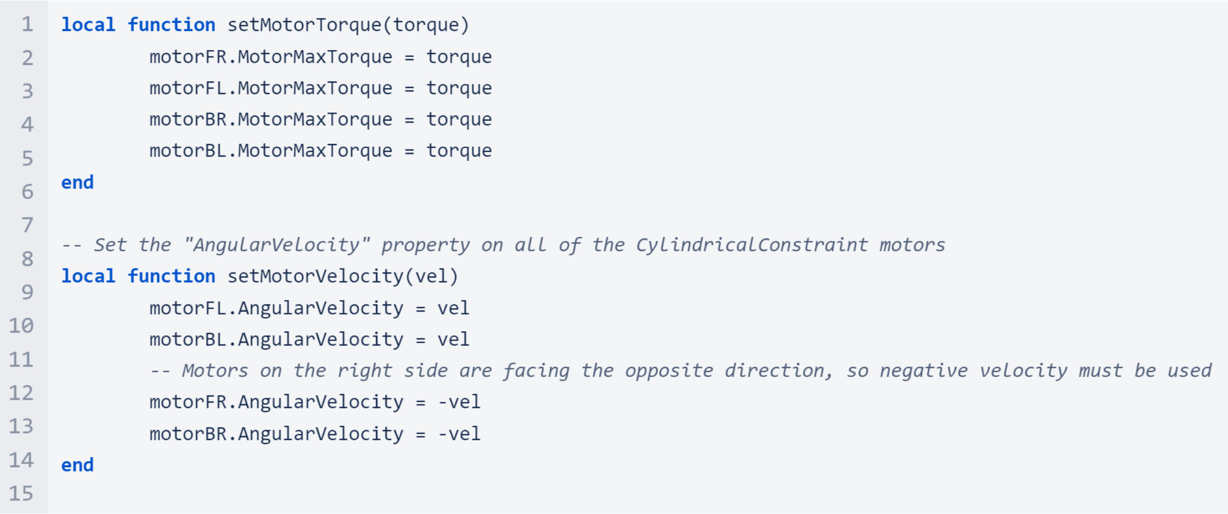


```lua
while getAverageVelocity() do

-- ...

end
```

101 Think about sort of values should be used in when setting each motor's torque and velocity. The two motor properties they will need are:

- motorX.MotorMaxTorque
- motorX.AngularVelocity

After setting the appropriate values, we obtain the code below.



```lua
local function setMotorTorque(torque)
        motorFR.MotorMaxTorque = torque
        motorFL.MotorMaxTorque = torque
        motorBR.MotorMaxTorque = torque
        motorBL.MotorMaxTorque = torque
end

-- Set the "AngularVelocity" property on all of the CylindricalConstraint motors
local function setMotorVelocity(vel)
        motorFL.AngularVelocity = vel
        motorBL.AngularVelocity = vel
        -- Motors on the right side are facing the opposite direction, so negative velocity must be used
        motorFR.AngularVelocity = -vel
        motorBR.AngularVelocity = -vel
end

```

THE END OF CLASS PROJECT!!!