Basic Drive Commands

<font size =3>Welcome to your first lesson with Zumi! 🎉 Before you begin machine learning with Zumi, she needs help from you to learn how to go forward, reverse, left, and right. Later in this lesson, you will learn how to make a custom remote control. </font>

Import libraries

<font size =3> The first step to running any code with Zumi will be to import libraries. In other words, this cell imports all of Zumi's necessary features, like the drive commands, camera, or screen. If you don't run this cell, the rest of your program won't work! You only need to run the following cell *once* in each lesson, unless you restart the Jupyter Notebook. </font>

In [ ]:

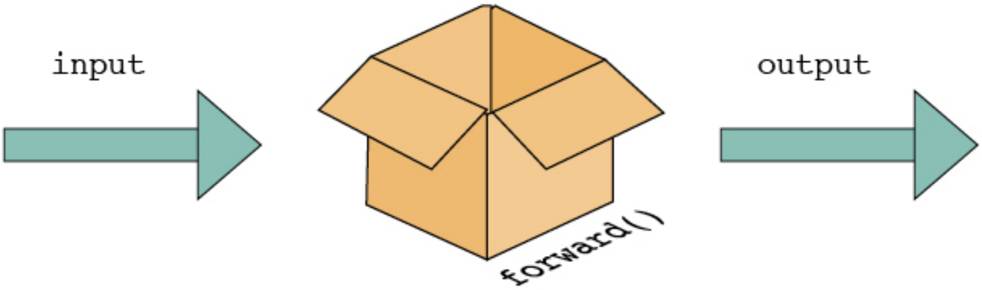
```
from zumi.zumi import Zumi
import time

zumi = Zumi()
```

Drive Commands

<font size =3> Zumi uses functions to drive. In this lesson, we will go more in-depth with what functions are, what they do, and how you can use them to create your remote control. </font>

What are functions?

<font size =3> To make Zumi drive, we need to use some **functions.** Think of functions as packages of code that you can use to make your program more efficient. They can take **inputs** and can have **outputs.**</font>



Zumi functions

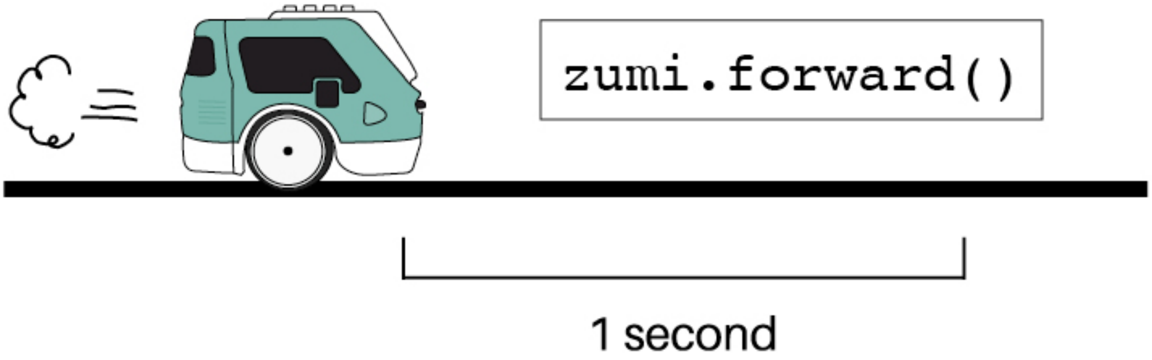
Below is a list of the basic drive functions:

- `forward()`: Drive forward in the direction Zumi is facing at speed 40 for 1 second
- `reverse()`: Reverse in the direction Zumi is facing at speed 40 for 1 second
- `turn_left()`: Pivot 90 degrees to the left
- `turn_right()`: Pivot 90 degrees to the right

</font>

How to call functions

<font size =3> In computer science, calling anything is basically asking it to run. Functions must be called using the object name, which in this case is `zumi`.



The cell below has an example using the `forward()` function. Zumi will drive forward for one second, so make sure you have enough space in its area! </font>

In [ ]:

```
zumi.forward()
```

<font size =3> Now try going in reverse... </font>

In [ ]:

```
# TODO Write code so Zumi reverses for 1 second
```

Let's go over the next two functions. Calling `turn_left()` and `turn_right()` will cause Zumi to turn to the left or turn to the right. Test this code below and then add more commands in any order to see what happens. If you want to have some time between each command, include a `time.sleep(seconds)` to delay the program for the specified number of seconds. Run the code below to see how this works, and then try adding some more commands to the code. </font>

In [ ]:

```
zumi.forward() # Drive forward for 1 second then stop
time.sleep(2) # Wait 2 seconds
zumi.turn_right() # Turn right 90 degrees
# Add more code here!
```

Parameters

<font size =3> At this point, you may want to change the duration, direction, and speed that Zumi drives forward. Some functions will allow you to input **parameters**, which are extra pieces of information that allow you to further customize your function for your needs. Right now `forward()` has a default speed, duration, and direction, but you can alter the parameters to change how fast Zumi drives, as well as for how long in a certain direction.

We’re going to skip changing Zumi’s direction since it requires some more understanding of Zumi's sensors, but you can change the speed and duration by defining them inside of the function call. In the cell below, the code has been modified to reduce the speed to 30 and drive for 2 seconds. Make sure you have enough space! </font>

In [ ]:

```
zumi.forward(speed=30, duration=2)
```

<font size =3> You can do the same for reverse. Change the speed and duration for `reverse()`below: </font>

In [ ]:

```
# TODO Modify reverse() to go at speed 20 for 3 seconds
```

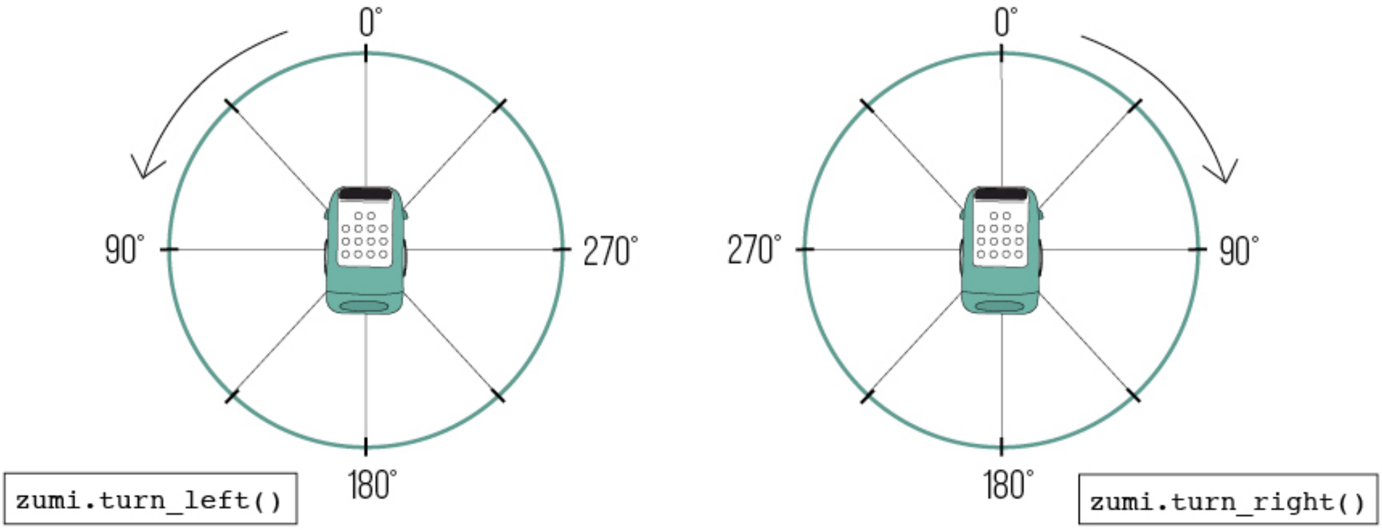
Degrees

The functions `turn_left()` and `turn_right()` also have parameters you can change. The default value is set to 90 degrees, but that value can be changed as well.
The code below will have Zumi turn right 45 degrees instead of 90:</font>

In [ ]:

```
zumi.turn_right(45)
```

Try out the accuracy of the turns by testing different angles. Use this diagram for reference if you need it! Zumi isn't perfect, so the actual angle that Zumi stops at might be off by one or two degrees.



Note: There is another hidden default parameter in `turn_right()` and `turn_left()`. When you call `turn_left(45)`, you are actually calling `turn_left(desired_angle=45, duration=1)`. The duration is what determines how much time Zumi has to complete that turn. One second is enough time for smaller turns, but what if you wanted to turn 135 degrees? You will also have to increase the time that Zumi needs to make the turn. `zumi.turn_left(120, 1.5)`

Since Zumi is turning 120 degrees, you should increase the duration or Zumi will not be able to finish. You may need to adjust the second parameter because each Zumi is unique. Use the cell below to experiment driving and making turns. </font>

In [ ]:

```
# Test some code here
```

Recalibrating

If you find that Zumi isn't going straight, you may need to recalibrate. This may happen if Zumi starts to overheat. Make sure you aren't picking up Zumi and she is resting on a flat surface.

In [ ]:

```
zumi.mpu.calibrate_MPU()
```

Now you know the basics! Use the cell below to test out some more code. For an added challenge, use some materials around you to build a simple obstacle course and write some code for Zumi to go through it without hitting anything. In the next lesson you will learn how to combine the drive commands with if statements to make your own remote control.

In [ ]:

```
# Write some more code here!
```

Extension Activities

Bridge Challenge

Build a bridge out of the object of your choice. However, this object should be flat and relatively wide so Zumi's wheels don't get stuck. Popsicle sticks will work well. Constraints like the degree of incline, height, width, and number of popsicle sticks can be used. After building the bridge, see if a) Zumi is able to cross it and/or b) if the bridge can support Zumi's weight. For an extra challenge, a payload can be added to Zumi.

Calculate Speed

Drive your Zumi for a certain distance or time while measuring the other. Next, use the equation d=rt to find the rate of travel. Afterwards, graph your results either and analyze the results. To incorporate physics, add a payload to Zumi. How does this affect Zumi's rate?