"Counters" in programming are essential. Counting allows us to keep track of processes in our program (like how many times a sensor has been activated, or a button has been pushed, *or how often someone visits a website*).

Here is how a counter works:

1. Declare an integer "int" variable at the beginning of your code *eg.* int **count;**
2. When you want to increase the value of **count** use the following expression:

count = count +1;

- *The above is NOT an equation. That would be impossible!*
- *In programming the this statement means:*

"set the a *new* value of **count** to: $\Longrightarrow$ the *current* value of **count** +1

Example code:

Look closely at the following code and figure out what each line it is supposed to do, then run the code on your NXT according to the instructions given

```
task main ()
{
wait1Msec(3000);

int count;

count = 0;
nMotorEncoder[WalzlDrive] = 0;

while (true)
{
        if (nMotorEncoder[WalzlDrive] > 360)

        {
        eraseDisplay();
        count = count +1;
        nxtDisplayCenteredBigTextLine(3,"%d",count);
        nMotorEncoder[WalzlDrive] = 0;
        }


}
}
```

Exercise#1

Run this code on Your NXT. Make sure it makes sense to you.

Code was hand written and has not be tested..so it may take some de-bugging

Note: **count= count +1;** is the same as **count++;**

Line Counter Assignment

The Problem:

You have been asked to write a piece of software that will count cracks found in a tunnel after an earthquake. Your Robot will have to go into a tunnel and count an unknown number of cracks on the ground. It's unsafe to send humans.

Specific Objectives:

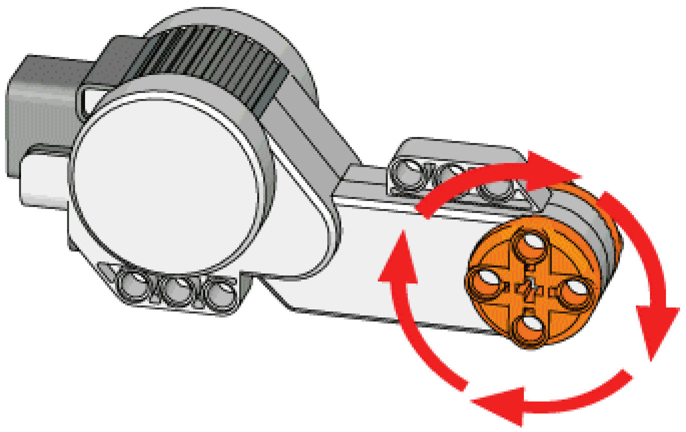
- Write a program that will **count black lines** on the ground as your robot drives forward.
- Your Robot should provide a continually **updating display** of the number of black lines it has counted.
- If the robot hasn't counted any lines in approximately 40cm, the robot will stop moving.
- Once your program begins to run, the program should wait until an **NXT button** of your choice is pressed. (you will have to look up how to use the NXT buttons.

Marking:

You must hand in an **algorithm,** your **code** and **get your line counter checked**
Use your Assessment and Learning Table as a guide

nMotorEncoder

- This is a command that reads or sets the “*position value*” of a motor.
- Each motor in your kit can be turned a number of degrees. The **nMotorEncoder** value is the number of degrees it has been turned.
- **Range is -32768 to 32767**. This means it will "wrap" after about 90 revolutions.

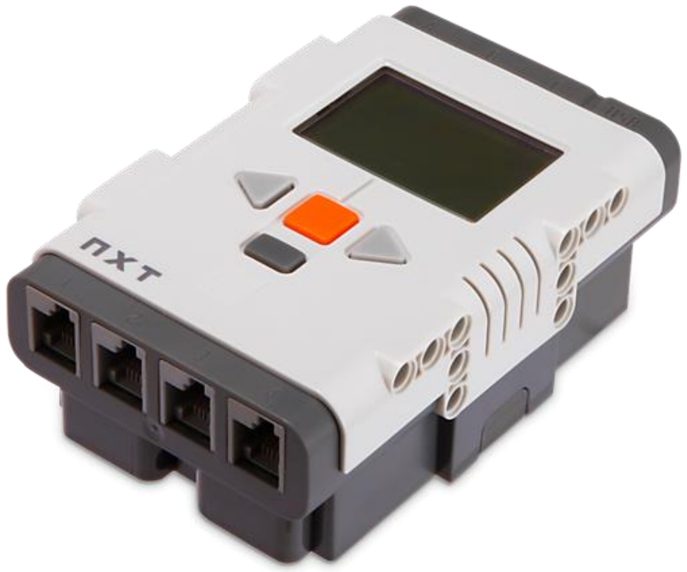


```
nMotorEncoder[motorB] = 0;

// reset the Motor Encoder of Motor B


while(nMotorEncoder[motorB] < 360)

// while the Motor Encoder of Motor B has not yet
reached 360:
```

The NXT motor is a “servo motor” we can control and monitor its position, speed, and direction precisely.

Buttons NXT

There are four buttons on the NXT front panel. The LEFT, ENTER, and RIGHT buttons are on the top row and the EXIT button is on the second row.



Normally, when a program is running, the LEFT and RIGHT buttons are used to toggle between various standard displays. And the EXIT button is used to stop execution of the user program. These characteristics can be overwritten so that button actions can be controlled within a running application program. There are several sample programs (e.g. Centipede, Tetris) included in the ROBOTC distribution that utilize application program control over buttons.

nNxtButtonPressed

Contains the number (0 to 3) of the button that is currently depressed.

-1 indicates no button is currently pressed.
0 = Gray Rectangle button.
1 = Right Arrow button.
2 = Left Arrow button.
3 = Orange Square button.

Note that only one button press can be recognized at a time. This is a limitation in the NXT hardware. It is unable to recognize multiple button presses.

```
if(nNxtButtonPressed == 1) // If the current pressed
button is 1 (the Right Arrow):
{
nxtDisplayCenteredBigTextLine(3, "RIGHT ARROW"); //
Display on line 3, a big, centered, message.
}
```