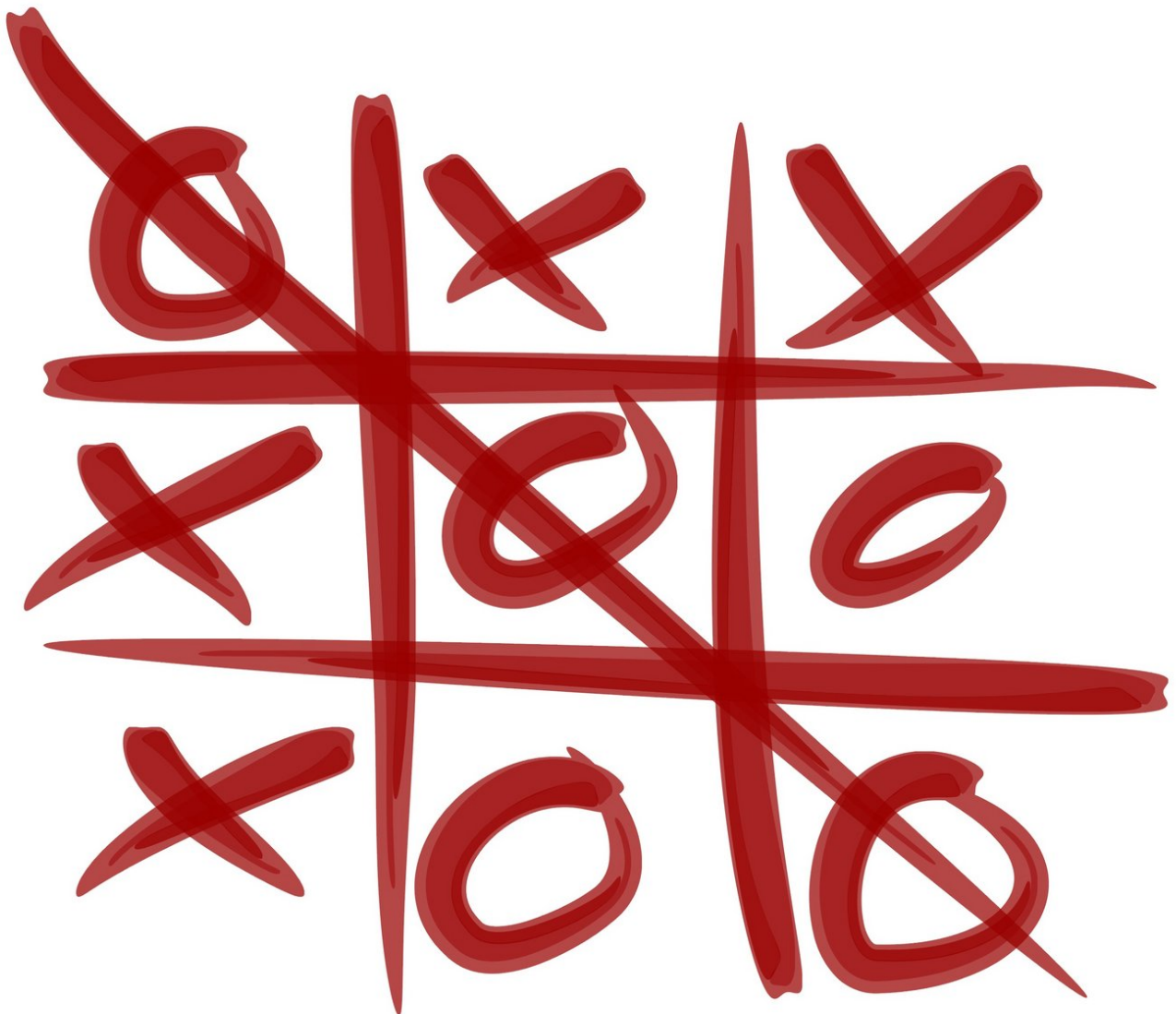


## D - Tic Tac Toe

Let's use our 9 sparkles to build a tic tac toe game!



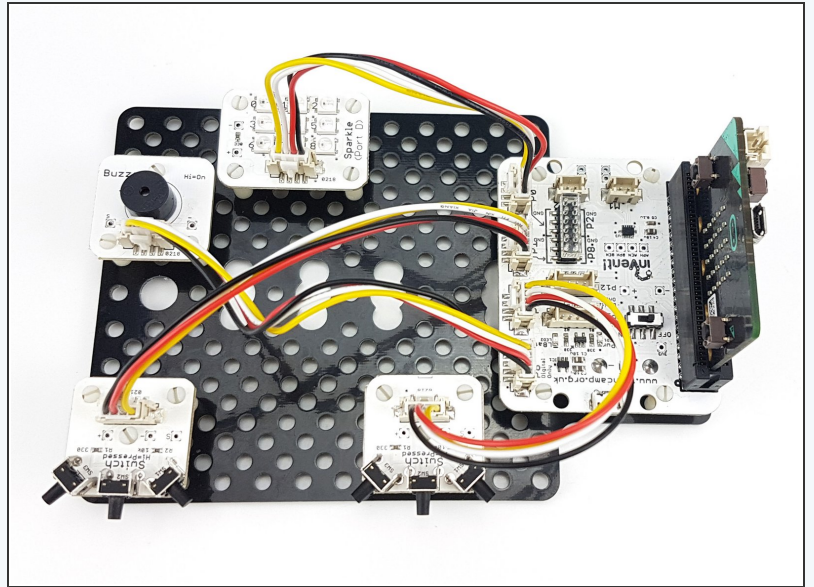
# INTRODUCTION

Let's use our 9 sparkles to build a tic tac toe game!

## Step 1

## Assemble the Robot

- First step - assemble the robot!
- The switches should be plugged into **P1** and **P2**.



## Step 2

### Arrays

- This game is quite complicated to program, and we are going to need to learn some **new programming concepts** to write it!
- First thing - lists (often called arrays in other languages). An list is just a special type of **variable**, which can contain a group of other variables.
- They are really useful when you need to store **lots of separate bits of information**, which would normally require **lots of variables**.
- You can make an array in almost the same way as a variable, except you also need to specify **what things it contains**. In Python, we use **square brackets []** to denote what things are inside the list.
- **Check the picture** for an example! This code makes an array called **numbers**, that contains **3 separate numbers**.

```
8 while(running_time()-p[0]<4000):v=[
9   p[6]=(p[5]+p[3])/2;p[7]=(p[4]+p[2])
10  while(p0()>p[7]and p1()>p[6]):d(0 i
11 def digital_read_line(s): return 1 if (
12 def analog_read_line(s): v=p0()if s==0
13 # Invent! Code End
14 # Start your code below here!
15
16 numbers=[3,4,5]
```

## Step 3

### Make an Array

- Let's make our own list!
- Starting with a **new program**, make a new array called **numbers** - it should contain **5 different numbers**.
- To access the variables (or **items** of the list, we use the **index** of the item. This is just its **position** - for example, the first item has index **0**, the second index **1** and so on.
- To access an item in a list, we use:
  - `listName[index]`
- For example, to access the **3rd** item of the **numbers** list, we would write `numbers[2]` (as the **third item has index 2**).
- Remember how to use the display? Use **`display.show(str(numbers[2]))`** to show the 3rd item of the list on the display. `str()` converts the number into text so we can display it on the screen.
- ❗ You might have noticed we have actually been using lists already with the sparkles!

```
8 while(running_time()-p[0]<4000):v=[
9   p[6]=(p[5]+p[3])/2;p[7]=(p[4]+p[2])
10  while(p0()>p[7]and p1()>p[6]):d(0 i
11 def digital_read_line(s): return 1 if (
12 def analog_read_line(s): v=p0()if s==0
13 # Invent! Code End
14 # Start your code below here!
15
16 numbers=[8,4,6,2,3]
17 display.show(str(numbers[2]))
18
19
20
21
22
23
```

## Step 4

### LED Array

- Let's make an **LED list**, so we can remember what colour we want each sparkle to be for the game without having a variable for each.
- Starting with a new program, **write 2 lines of code to setup the sparkles** so we can use them - look at the previous guide if you can't remember how to do it.
- Underneath, make a list called **LED** with **9 items, all of which are 0**.



## Step 5

### Arrays and For Loops

- Arrays become very powerful when you **combine them with a for loop**.
- The key is to use the **iterable** (i usually) to access the array elements, that way the code can look at **any sequence of items you want, automatically**. Again, this is exactly what we have been doing already to set the colours of all of the sparkles at the same time.
- Add the for loop in the picture into your program - **what do you think it will do?**
- **Run the code** to find out!



**Make sure you understand how this code works before moving on** - ask your teacher if you need help.

```
10 while(p0()>P[7] and p1()>P[8]): a(0,1,0)
11 def digital_read_line(s): return 1 if (
12 def analog_read_line(s): v=p0()if s==0
13 # Invent! Code End
14 # Start your code below here!
15
16 import neopixel
17
18 pixels=neopixel.NeoPixel(pin0,9)
19 leds=[0,0,0,0,0,0,0,0,0]
20
21 for i in range(0,9):
22     display.scroll(str(leds[i]))
23
```

## Step 6

### Update the LEDs

- We are going to use this list to **store** the 'states' of the LEDs for the tic tac toe game:
- **Each item represents an LED** - the first item is LED 0, the second LED 1 and so on.
- If the item is **0**, the LED is **off** (not selected by a player)
- **1** means taken by the **red** player
- **2** means taken by the **blue** player
- Using the layout in the picture, **fill in the code in the loop** so that the sparkle colours are updated depending on the values of the items in the list
- **Change some of the items** in the list to 1 or 2 to test it out! (Don't forget, you will need to add a **pixels.show()** line to see anything)
- Notice the if - elif - else setup we have here? This is called an **else if** statement - the else if block is **only checked if the first if is false**. The final else is then only run if **both the if, and else if** don't run (because their conditions are false).

```
14 # Start your code below here:
15
16 import neopixel
17
18 pixels=neopixel.NeoPixel(pin0,9)
19 leds=[0,0,0,0,0,0,0,0,0]
20
21 for i in range(0,9):
22     if leds[i]==1:
23         # Set sparkle i red
24     elif leds[i]==2:
25         # Set sparkle i blue
26     else:
27         # Otherwise must be off
28         # Turn off sparkle i
29
```





## Step 7

### Functions

- We have been using functions since the start, like `sleep()`, `drive_motor()` and so on.
- Now its time to learn how to **make our own!**
- Writing functions allows you to **reuse code** in longer programs, where you might need to do the same thing more than once.
- For example, we are going to want to update the LEDs from the list in **more than one place!**
- We first need to **declare** the function before we can use it- this sets the **name**, what **inputs** it takes, and if it returns any **data**.
- Have a look at the diagram - in Python, the **def** keyword defines a function, and any code we want to run when we call the function is **indented** underneath it.

```
def functionName(input1, input2, input3):  
    # Function code goes here
```

## Step 8

### Make a Function

- Let's make a function that **updates the LEDs**.
- **Move** all your code from the loop that updates the LEDs, into a function that is:
  - Called **updateLeds**
  - Takes **no inputs** (empty brackets)
- Make sure the function is **after** the sparkle setup lines, and the led list.
- Your code should look like the picture - make sure it is **indented** properly!

```
14 # Start your code below here!  
15  
16 import neopixel  
17  
18 pixels=neopixel.NeoPixel(pin0,9)  
19 leds=[0,0,0,0,0,0,0,0,0]  
20  
21 def updateLeds():  
22     for i in range(0,9):  
23         if leds[i]==1:  
24             # Set sparkle i red  
25         elif leds[i]==2:  
26             # Set sparkle i blue  
27         else:  
28             # Otherwise must be off  
29             # Turn off sparkle i
```

## Step 9

### Use the Function

- Now we have **declared** the function, we can use it wherever we like!
- You use (or **call**) a function just like the functions we have been using already - **write its name!**
- **Call** the `updateLeds()` function inside a **while True:** loop, and **test your code** still works like before.

```
15
16 import neopixel
17
18 pixels=neopixel.NeoPixel(pin0,9)
19 leds=[0,0,0,0,0,0,0,0,0]
20
21 def updateLeds():
22     for i in range(0,9):
23         if leds[i]==1:
24             # Set sparkle i red
25         elif leds[i]==2:
26             # Set sparkle i blue
27         else:
28             # Otherwise must be off
29             # Turn off sparkle i
30
31 while True:
32     updateLeds()
```

## Step 10

### Space Select

- To play the game, the user has to select which space they want to take. We will use a **switch** to cycle through the LEDs, so they can **choose where they want to go**.
- We need some **more variables** first - create these underneath the leds array:
  - **turn** - this keeps track of whose turn it is. Initialise it to **1** (red player).
  - **space** - this keeps track of which space is currently selected. Initialise it to **0**.
- Add an **IF statement** to the loop, that checks if the switch in **P1** is pressed. If it is, **increase space by 1**.
- We need to make sure space doesn't go higher than 8, as this is the **last LED!**
- Add another if statement that checks if **space ==9** - if it does, **reset space to 0**.

```
15
16 import neopixel
17
18 pixels=neopixel.NeoPixel(pin0,9)
19 leds=[0,0,0,0,0,0,0,0,0]
20 turn=1
21 space=0
22
23 def updateLeds():
24     for i in range(0,9):
25         if leds[i]==1:
26             pixels[i]=(255,0,0)
27         elif leds[i]==2:
28             pixels[i]=(0,0,255)
29         else:
30             pixels[i]=(0,0,0)
31
32 while True:
33     updateLeds()
34     if pin1.read_digital()==1:
35         # Increase space by 1
36         if space==9:
37             # Set space to 0
```

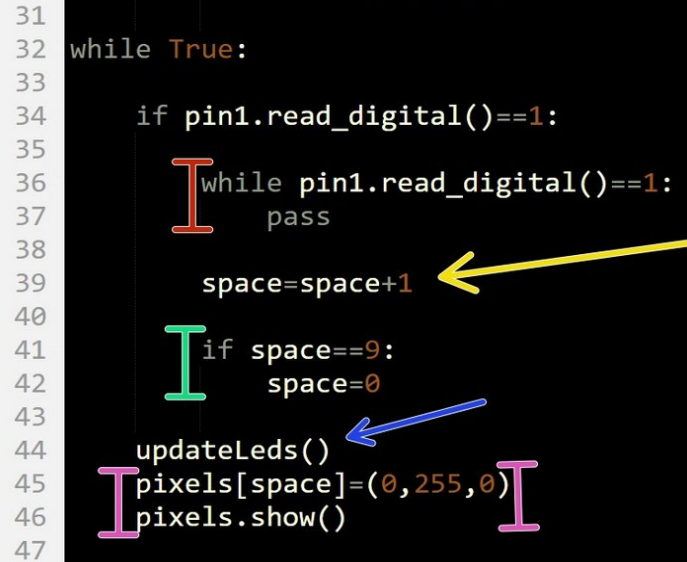


## Step 11

### Select LED

- Final step - we want to turn the currently selected LED **green**, to show the user **which space they have selected**.
- We need to move some things around and update the LEDs in the right places to make this work. Your code in the while True: loop should:
  - Check the switch - if it is pressed, use a **while** loop to wait for it to be released
  - Increase **space** by 1
  - Check if space has reached **9** - if it has, reset it to 0
  - Update the LEDs from the array, by calling **updateLeds()**
  - Change the LED with the same number as **space** to **green**.
- Check the picture for more hints if you need them! Don't forget to add the **pixels.show()** line at the end, otherwise the sparkles won't update.

```
31
32 while True:
33
34     if pin1.read_digital()==1:
35
36         while pin1.read_digital()==1:
37             pass
38
39         space=space+1
40
41         if space==9:
42             space=0
43
44         updateLeds()
45         pixels[space]=(0,255,0)
46         pixels.show()
47
```

A diagram of a code editor window showing Python code. The code is a while loop that checks if a switch is pressed. If pressed, it enters a nested while loop to wait for the switch to be released. Then, it increments a 'space' variable. If 'space' reaches 9, it resets to 0. It then calls 'updateLeds()' and sets the 'space'th pixel to green (0, 255, 0), followed by 'pixels.show()'. Annotations include: a yellow arrow pointing to 'space=space+1', a blue arrow pointing to 'updateLeds()', and pink 'I' characters highlighting 'pixels[space]=(0,255,0)' and 'pixels.show()'.

## Step 12

### Place Counter

- You code should now allow you to cycle the green LED by pressing the switch!
- Now we need to use the **other switch** to place the players counter, and turn an LED red or blue.
- Add an **if statement** at the end of the loop, to check the second switch. If it is pressed, add a **while loop** to wait until it is released.

```
31
32 while True:
33
34     if pin1.read_digital()==1:
35         while pin1.read_digital()==1:
36             pass
37             space=space+1
38             if space==9:
39                 space=0
40
41     updateLeds()
42     pixels[space]=(0,255,0)
43     pixels.show()
44
45     if pin2.read_digital()==1:
46         while pin2.read_digital()==1:
47             pass
48
49
50
```

## Step 13

### Place Counter

- We now need to store the space the player has chosen in the **leds array**.
- Inside the if checking the second switch, **add some code** to:
  - Check if **turn ==1** - if it does, set **leds[space]** to 1 (**red space**), and set **turn =2**
  - Otherwise it must be **blue's turn**, so set **leds[space]** to 2, and set **turn=1**
- **Test your code properly** - you might find a few problems that stop the game working exactly how it should!

```
37         space=space+1
38         if space==9:
39             space=0
40
41     updateLeds()
42     pixels[space]=(0,255,0)
43     pixels.show()
44
45     if pin2.read_digital()==1:
46         while pin2.read_digital()==1:
47             pass
48             if turn==1: # Player 1's turn
49                 # Set leds[space] to 1
50                 # Set turn to 2
51             else: # Must be player 2's turn
52                 # Set leds[space] to 2
53                 # Set turn to 1
54
55
```

## Step 14

### Stop the Cheating!

- You might have found that you can select a space that **the other player has already taken!**
  - For this challenge, add a while loop after space is increased, that **keeps adding 1** to space until the selected space has not already been taken!
  - Make sure you test your program properly and **iron out any bugs.**
- i* Here's a hint - you will need to find a way of stopping this loop going forever when there are no spaces left! You can do this by counting the number of turns taken.....



## Step 15

### Extension Tasks

- Hopefully you should now have a **basic working game!**
- Here are some ideas for improvements you can make:
  - When all of the spaces are used up, flash all the LEDs white and sound the buzzer, then reset the game
  - Hard challenge - write another function that automatically checks if someone has won the game, i.e. got three in a row. This is very difficult!
  - Really really super mega hard challenge - extend the winner checking function so that it checks if it is still possible to win the game at all!

